

# **Agilent PNA Series Network Analyzer**

**Printed Version of PNA Help  
User's and Programming Guide**

**Supports Firmware A.09.22**

**June 3, 2010**



**Agilent Technologies**

---

## Warranty Statement

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AGILENT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. AGILENT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD AGILENT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

---

## DFARS/Restricted Rights Notice

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

---

## Certification

Agilent Technologies, Inc. certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies, Inc. further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

---

## Contacting Agilent

Assistance with test and measurements needs and information on finding a local Agilent office are available on the Web at: <http://www.agilent.com/find/assist>

If you do not have access to the Internet, please contact your Agilent field engineer.

In any correspondence or telephone conversation, refer to the Agilent product by its model number and full serial number. With this information, the Agilent representative can determine whether your product is still within its warranty period.

Printed **Version of PNA Help User's and Programming Guide**

## TABLE OF CONTENTS

---

Whats New	1
Administrative Tasks	
Microsoft EULA	6
PNA User Accounts and Passwords	14
Computer Properties	18
Error-check and Disk Defragmenter	22
Operating System Recovery	23
Wake PNA at Specified Time	24
Windows Considerations	25
Quick Start	
Front Panel Tour	27
Rear Panel Tour (E836x and PNA-L)	36
Rear Panel Tour (PNA-X)	38
Powering the PNA ON and OFF	42
Traces, Channels, and Windows on the PNA	45
Basic Measurement Sequence	52
Frequency Blanking	53
Internal Second Source	56
Connectivity Guide	58
PNA Preferences	59
VNC	64
LXI Compliance	65
Using Help	68
Help About	75
1. Set Up a Measurement	
Preset the PNA	77
Measurement Parameters	85
Measurement Classes	92
Frequency Range	95
Power Level	104
Receiver Leveling	114
Sweep Settings	118
Trigger Setup	129
External Triggering	133

Trigger Model Animation	142
Data Format	143
Scale	150
Preconfigured Measurement Setups	155
Path Configurator	159
Customize Your Analyzer Screen	163
Copy Channels	174
ADC Measurements	177
2. Optimize a Measurement	
Dynamic Range	180
Number of Data Points	184
Phase Accuracy	186
Electrically Long Devices	190
Reflection Accuracy	192
Measurement Stability	195
Noise Reduction Techniques	197
Crosstalk	205
Effects of Accessories	206
Fastest Sweep	207
Multiple State Measurements	209
Fastest Data Transfer	212
Using Macros	213
3. Calibrate a Measurement	
Calibration Overview	217
Calibration Standards	219
Calibration Wizard	223
Select a Calibration	238
Using Cal Sets	242
Error Correction and Interpolation	250
Calibration Thru Methods	254
Accurate Calibrations	260
Validity of a Cal	263
ECal	268
ECal User Characterization	277
TRL Calibration	291
Calibration Preferences	295
Measurement Errors	300
Modify Cal Kits	312



Power Calibration	331
Fixture Compensation	346
Port Extensions	359
Characterize Adaptor Macro	366
Delta Match Calibration	375
4. Analyze Data	
Locate Data Using Markers	378
Math & Memory Operations	391
Equation Editor	397
Use Limits to Test Devices	411
5. Output Data	
Save and Recall Data	416
Drive Mapping	433
Print	435
Programming	441
COM	
Commands	
Objects	
The Analyzer Object Model	442
Application Object	445
AuxTrigger Object	452
BalancedMeasurement Object	454
BalancedStimulus Object	456
BalancedTopology Object	458
CalFactorSegments Collection	460
Calibrator Object	461
CalKit Object	466
CalManager Object	468
CalSet Object	470
CalSets Collection	475
CalStandard Object	476
Capabilities Object	478
Channel Object	481
Channels Collection	488
Converter_Object	490
E5091Testset Collection	493
E5091Testset Object	495
ECalUserCharacterizer Object	497

EmbeddedLO Object	499
EmbeddedLODiagnostic Object	501
ENRFile Object	503
Equation Object	505
ExternalTestsets Collection	507
FIFO Object	509
Fixturing Object	511
FOM Collection	516
FOMRange Object	518
GainCompression Object	521
GainCompressionCal Object	525
Gating Object	527
GuidedCalibration Object	529
HWAuxIO Object	532
HWExternalTestSetIO Object	535
HWMaterialHandlerIO Object	537
IIFConfiguration Object	539
IMixer Interface	541
IMSpectrum Object	545
InterfaceControl Object	547
Limit Test Collection	548
LimitSegment Object	550
Marker Object	552
Measurement Object	555
Measurements Collection	562
NAWindow Object	563
NAWindows Collection	565
NoiseCal Object	566
NoiseFigure Object	568
PathConfiguration Object	571
PathConfigurationManager Object	573
PathElement Object	575
Port Extension Object	577
PowerLossSegment Object	579
PowerLossSegments Collection	581
PowerMeterInterface Object	582
PowerMeterInterfaces Collection	584
PowerSensor Object	586

PowerSensorCalFactorSegment Object	588
PowerSensors Collection	589
Preferences Object	590
PulseGenerator Object	593
ReceiverLevelingConfiguration Object	596
SCPIStringParser Object	598
Segment Object	600
Segments Collection	602
SignalProcessingModuleFour Object	604
SMC Type Object	607
SourcePowerCalibrator Object	610
SweptIMD Object	613
SweptIMDCal Object	617
TestsetControl Object	619
Trace Object	621
Traces Collection	623
Transform Object	624
TriggerSetup Object	626
VMC Type Object	628
Properties	
AcceptTriggerBeforeArmed	631
AcquisitionDirection	632
AcquisitionMode	633
Active Cal Kit	634
Active Channel	635
Active Marker	636
Active Measurement	637
Active NAWindow	638
ActiveXAxisRange	639
ActiveXAxisRange(Conv)	640
ADCCaptureMode	641
ALCLevelingMode	642
Active Trace	644
AllowArbitrarySegments	645
Alternate Sweep	646
AmbientTemperature	647
Application	648
Arrange Windows	649

Attenuator Mode	650
Attenuator	651
AutoIFBWAdjustment	653
AutoOrient	654
AutoPortExtConfig	655
AutoPortExtDCOffset	656
AutoPortExtLoss	657
AutoPortExtSearchStart	658
AutoPortExtSearchStop	659
AutoPortExtState	660
AuxiliaryTriggerCount	661
AuxTriggerScopelsGlobal	662
AverageMode	664
Averaging Count	665
Averaging Factor	666
Averaging ON/OFF	667
AvoidSpurs	668
BalancedMode	669
BalPort1PhaseOffset	670
BalPort1PowerOffset	671
BalPort1StartPhase	672
BalPort1StopPhase	673
BalPort2PhaseOffset	674
BalPort2PowerOffset	675
BalPort2StartPhase	676
BalPort2StopPhase	677
Bandwidth Target	678
Bandwidth Tracking	679
BB_BalPort1Negative	680
BB_BalPort1Positive	681
BB_BalPort2Negative	682
BB_BalPort2Positive	683
BBalMeasurement Property	684
Begin Response	685
Begin Stimulus	686
BroadbandTuningSpan	687
Bucket Number	688
C0	689

C1	690
C2	691
C3	692
Cal Factor	693
Cal Type (applied)	694
Calibration Name	696
Calibration Port	697
CalibrationTypeID	698
CalibrationFrequencies	700
Cal KitType	701
CalKitType (FCA)	702
CalMethod	704
CalMethod (imd)	705
Cal Power	706
Center	707
Center (Meas)	708
Center Frequency	709
Channel Number	710
CharacterizationNumber	711
CharacterizeMixerOnly	712
CharFileName	713
CharMixerReverse	714
CitiContents	715
CitiFormat	716
CmnModeZConvPortImag	717
CmnModeZConvPortReal	718
CmnModeZConvPortZ0	719
CmnModeZConvState	720
CompatibleCalKits	721
CompositeNormalizationMode	722
CompositeNormalizedCSOPower	723
CompositeNormalizedCTBPower	724
CompressionAlgorithm	725
CompressionBackoff	726
CompressionDeltaX	727
CompressionDeltaY	728
CompressionInterpolation	729
CompressionLevel	730

CompressionLevel (mkr)	731
CompressionPin	732
CompressionPout	733
ConnectorType	734
ConnectorType ECal	735
ControlLines	736
Count	738
Couple Ports	740
CoupleChannelParams	741
Coupled	742
Coupled Markers	743
CoupledParameters - Gate	744
CoupledParameters - Transform	745
CoupleTonePower	746
CSONumDistortionProducts	747
CSOOffset	748
CTBOffset	749
CTBXMODNumCarriers	750
CustomChannelConfiguration	751
CW Frequency	752
Data	753
DataCount	754
DataInCompactForm	755
Delay	756
Delay pulse	757
Delay_trigger	758
DelayIncrement	759
DeltaFrequency	760
DeltaFrequencyStart	761
DeltaFrequencyStop	762
DeltaMarker	763
Description	764
DescriptiveText	765
DeviceInputPort (FCA)	766
DeviceInputPort	767
DeviceOutputPort (FCA)	768
DeviceLinearPowerLevel	769
DeviceOutputPort	770

DiffPortMatch_C	771
DiffPortMatch_G	772
DiffPortMatch_L	773
DiffPortMatch_R	774
DiffPortMatchMode	775
DiffPortMatchState	776
DiffPortMatchUserFilename	777
DiffZConvPortImag	778
DiffZConvPortReal	779
DiffZConvPortZ0	780
DiffZConvState	781
Display Format	782
DisplayAutomationErrors	784
DisplayGlobalPassFail	785
DisplayRange	786
Distance	787
DistanceMarkerMode	788
DistanceMarkerUnit	789
Divisor	790
Do1PortEcal	791
Do2PortEcal	792
Domain	793
DUTTopology	794
Dwell Time	795
ECALCharacterization (smc)	796
ECALCharacterization (vmc)	797
ECALCharacterizationEx	798
ECaIID	799
ECALIsolation	800
ECALModuleNumberList	801
EcalOrientation	802
EcalOrientation1Port	804
EcalOrientation2Port	805
ECALPortMapEx	807
ElecDelay Medium	809
ElecDistanceDelay	810
ElecDistanceDelayUnit	811
Electrical Delay	812

Element	813
Elements	814
Embed4PortA	815
Embed4PortB	816
Embed4PortC	817
Embed4PortD	818
Embed4PortList	819
Embed4PortNetworkFilename	821
Embed4PortNetworkMode	822
Embed4PortState	823
Embed4PortTopology	824
Enable	825
Enabled	826
EnableLOPowerCal	827
EnableSourceUnleveledEvents	828
EndOfSweepOperation	829
ENRFile	830
ENRID	831
ENRSN	832
Error Correction	833
ErrorCorrection(Channel)	834
External ALC	835
ExternalTriggerConnectionBehavior	836
ExternalTriggerDelay	838
F1Frequency	839
F2Frequency	840
FastCWPointCount	841
FastMode	842
Filter BW	843
Filter CF	844
Filter Loss	845
Filter Q	846
FilterErrors	847
FilterMode	849
FirmwareMajorRevision	850
FirmwareMinorRevision	851
FirmwareSeries	852
FixturingState	853



FootSwitch	854
Footswitch Mode	855
Format (Marker)	856
Format	782
FormatUnit	858
Frequency	859
FrequencyCenter	860
FrequencyCenterCenter	861
FrequencyCenterspan	862
FrequencyCenterStart	863
FrequencyCenterStop	864
FrequencyList	865
Frequency Offset Divisor	866
Frequency Offset Frequency	867
Frequency Offset Multiplier	868
Frequency Offset Override To CW	869
Frequency Offset State	870
FrequencySpan	871
Gate Shape	872
Gate Type	873
GPIBAddress	874
GPIB Mode	875
GPIBPortCount	876
HandshakeEnable	877
HighestOrderProduct	878
ID	879
IDString	880
IF Bandwidth Option	881
IF Bandwidth	882
IFFilterSampleCount	883
IFFilterSamplePeriod	884
IFFilterSamplePeriodList	885
IFFilterSamplePeriodMode	886
IFFilterSource Property	887
IFGainLevel	888
IFGainMode	889
IFGateEnable	890
IFFrequency	891

IFFrequencyMode	892
IFSideband	893
IFSideband (conv)	894
IFSourcePath	895
IFStartFrequency	896
IFStopFrequency	897
ImpedanceStates	898
Impulse Width	899
IMToneIFBandwidth	900
Include2ndOrderProduct	901
IndexState	902
Input A	903
Input B	904
Input C	905
InputDenominator	906
InputFixedFrequency	907
InputIsGreaterThanLO	908
InputNumerator	909
InputPower	910
InputRangeMode	911
InputStartFrequency	912
InputStartPower	913
InputStopFrequency	914
InputStopPower	915
InternalTestsetPortCount	916
Interpolate Correction	917
Interpolated	918
InterpolateNormalization	919
Interrupt	920
IsContinuous	921
IsECALModuleFoundEx	922
IsFrequencyOffsetPresent	923
IsHold	924
IsMarkerOn	925
IsolationAveragingIncrement	926
IsOn	927
IsReceiverStepAttenuatorPresent	928
IsReferenceBypassSwitchPresent	929

IsSPParameter	930
IterationNumber	931
IterationsTolerance	932
Kaiser Beta	933
L0	934
L1	935
L2	936
L3	937
Label	938
Label Testset	939
LANConfiguration	940
LastModified	941
LevelingIFBW	942
LimitTestFailed	943
Limit Line Begin Stimulus	686
Limit Line End Stimulus	944
Limit Line Begin Response	685
Limit Line End Response	945
Limit Type	946
Line Display	947
LoadCharFromFile	948
LoadPort	949
LocalLockoutState	950
Locator	951
LODeltaFound	952
LODenominator	953
LOFixedFrequency	954
LOFrequencyDelta	955
OBS_LogMagnitudeOffset	956
LOName	957
LONumerator	959
LOPower	960
LORangeMode	961
Loss	962
Loss (sourceCal)	963
LOStage	964
LOStartFrequency	965
LOStartPower	966

LOStopFrequency	967
LOStopPower	968
LXIDeviceIDState	969
MagnitudeOffset	970
MagnitudeSlopeOffset	971
MainToneIFBandwidth	972
Marker Annotation	973
Marker Bucket Number	688
Marker Format (all)	974
Marker Format (indiv)	856
Marker Interpolate(all)	976
Marker Interpolate (indiv)	918
Marker Number	977
Marker Position	978
Marker Readout	979
Marker ReadoutSize	980
Marker State	981
Marker Type	982
Marker X-axis Value	983
Marker Y-axis Value	984
Maximum Frequency	986
MaximumFrequency (capabilities)	987
MaximumFrequency (sourceCal)	988
MaximumIFFilterSampleCount	989
MaximumIFFrequency	990
MaximumIterationsPerPoint	991
MaximumNumberOfChannels	992
MaximumNumberOfTracesPerWindow	993
MaximumNumberOfWindows	994
MaximumNumberOfPoints	995
MaximumReceiverStepAttenuator	996
MaximumSourceALCPower	997
MaximumSourceStepAttenuator	998
MaxPreciseTuningIterations	999
MaxProduct	1000
Mean	1001
MeasurementClass	1002
Medium	1003

Minimum Frequency	1004
MinimumFrequency (capabilities)	1005
MinimumFrequency (sourceCal)	1006
MinimumIFFilterSampleCount	1007
MinimumIFFrequency Property	1008
MinimumNumberOfPoints	1009
MinimumReceiverStepAttenuator	1010
MinimumSourceALCPower	1011
Mode	1012
Mode-iTMSA	1013
Multiplier	1014
Name (Calset)	1015
Name (CalKit object)	1016
Name config	1017
Name element	1018
Name FOMRange	1019
Name (meas)	1020
Name (trace)	1021
NetworkFilename	1022
NetworkMode	1023
NoiseAverageFactor	1024
NoiseAverageState	1025
NoiseBandwidth	1026
NoiseGain	1027
NoiseSourceCalKitType	1028
NoiseSourceCold	1029
NoiseSourceConnectorType	1030
NoiseSourceState	1031
NoiseTuner	1032
NoiseTunerIn	1033
NoiseTunerOut	1034
NominalIncidentPowerState	1035
NormalizePoint Property	1036
Number (meas)	1037
Number	1038
NumberOfFrequencyPoints	1039
Number of Points	1040
Number of Points (Meas)	1042

NumberOfPorts	1043
NumberOfPorts(Testset)	1044
NumberOfPowerPoints	1045
NumberOfSweeps	1046
Offset	1047
OffsetReceiverAttenuator	1048
OffsetSourceAttenuator	1049
OmitIsolation	1050
OneReadoutPerTrace	1051
Options	1052
OrientECALModule	1053
OutputFixedFrequency	1055
OutputPort	1056
OutputPorts calset	1058
OutputPorts	1059
OutputRangeMode	1060
OutputSideband (conv)	1061
OutputSideband	1062
OutputStartFrequency	1063
OutputStopFrequency	1064
Parameter	1065
Parameter_elo	1066
Parent	1067
PassFailLogic	1069
PassFailMode	1070
PassFailPolicy	1071
PassFailScope	1072
PassFailStatus	1073
Path Property	1074
PathCalMethod	1075
PathThruMethod	1077
Peak Excursion	1079
Peak Threshold	1080
PeakTo Peak	1081
Period	1082
Phase Offset	1083
PhaseAsFixture	1084
PhaseSwpAsFixture	1085

PhaseSwpState	1086
PointAveragingState	1087
PointSweepState	1088
Port 1	1089
Port 2	1090
Port 3	1091
Port1NoiseTunerSwitchPresetsToExternal	1092
Port2PdeembedCktModel	1093
Port2PdeembedState	1094
PortArbzImag	1095
PortArbzReal	1096
PortArbzState	1097
PortArbzZ0	1098
PortCatalog	1099
PortCLogic	1100
PortCMode	1101
PortCoupleToSystemMedia	1102
PortCoupleToSystemVelocity	1103
PortDelay	1104
PortDescription	1105
PortDistance	1106
PortDistanceUnit	1107
PortExtState	1108
PortExtUse1	1109
PortExtUse2	1110
PortFreq1	1111
PortFreq2	1112
Port Label	1113
PortLogic	1114
PortLoss1	1115
PortLoss2	1116
PortLossDC	1117
PortMatching_C	1118
PortMatching_G	1119
PortMatching_L	1120
PortMatching_R	1121
PortMatchingCktModel	1122
PortMatchingState	1123

PortMedium	1124
PortMode	1125
PortsNeedingDeltaMatch	1126
PortVelocityFactor	1127
PortWGCutoffFreq	1128
Power Slope	1129
Power Acquisition Device	1130
PowerAsFixture	1131
PowerLevel	1132
PowerMax	1133
Power Meter Channel	1134
Power Meter GPIBAddress	1135
PowerMin	1136
PowerOffset	1137
PowerOnDuringRetraceMode	1138
PowerSensorCalKitType	1139
PowerSensorConnectorType	1140
PowerStep	1141
PowerSweepRetracePowerMode	1142
PreciseTuningTolerance	1143
PreferInternalTriggerOnChannelSingle	1144
PreferInternalTriggerOnUnguidedCal	1146
RangeCount	1147
rangeNumber	1148
R1 Input Path	1149
Readings Per Point	1150
ReadingsTolerance	1151
ReadyForTriggerPolarity	1152
ReadyForTriggerState	1153
Receiver Attenuator	1154
ReceiverCount	1155
ReceiverStepAttenuatorStepSize	1156
Receive Port	1157
ReduceIFBandwidth	1158
ReferenceCalFactor	1159
Reference Marker State	1160
ReferenceReceiver	1161
Reference Level	1162



Reference Position	1163
RemoteCalStoragePreference	1164
ResolutionBW	1165
ReverseLinearPowerLevel	1166
SafeMode	1167
SafeSweepCoarsePowerAdjustment	1168
SafeSweepEnable	1169
SafeSweepFinePowerAdjustment	1170
SafeSweepFineThreshold	1171
SB_BalPortNegative	1172
SB_BalPortPositive	1173
SB_SEPort	1174
SBalMeasurement	1175
Scope	1176
Search Function	1177
SearchFailures	1178
SecurityLevel	1179
Segment Number	1180
SelectPort	1181
Show Statistics	1182
ShowProperties	1183
SICL	1184
SICLAddress	1185
Simultaneous2PortAcquisition	1186
SmartSweepMaximumIterations	1187
SmartSweepSettlingTime	1188
SmartSweepShowIterations	1189
SmartSweepTolerance	1190
Smoothing Aperture	1191
Smoothing ON/OFF	1192
SnPFormat	1193
Sound On Fail	1194
SourceCount	1195
Source Port	1196
SourcePortCount	1197
SourcePortMode	1198
SourcePortNames	1199
Source	1201

SourcePowerCalPowerOffset	1202
Source Power Correction	1203
Source Power Option	1204
Source Power State	1205
Span	1206
Span (Meas)	1207
SpectrumCenterFrequency	1208
SpectrumSpanFrequency	1209
SpectrumStartFrequency	1210
SpectrumStopFrequency	1211
SSB_BalPortNegative	1212
SSB_BalPortPositive	1213
SSB_SEPort1	1214
SSB_SEPort2	1215
SSBMeasurement	1216
Stage1Coefficients	1217
Stage1Frequency	1218
Stage1MaximumCoefficient	1219
Stage1MaximumCoefficientCount	1220
Stage1MaximumCoefficientSum	1221
Stage1MinimumCoefficientCount	1222
Stage2Coefficients	1223
Stage2MaximumCoefficient	1224
Stage2MaximumCoefficientCount	1225
Stage2MaximumCoefficientSum	1226
Stage2MinimumCoefficientCount	1227
Stage3FilterType Property	1228
Stage3FilterTypes Property	1229
Stage3Parameter Property	1230
Stage3ParameterMaximum Property	1231
Stage3ParameterMinimum Property	1232
Stage3Parameters Property	1233
Standard Deviation	1234
Standard For Class	1235
Start Frequency_CS	1237
Start Frequency	1238
Start Power	1240
Start	1241

Start (Meas)	1242
StartPowerEx	1243
State	1244
State rx	1246
State pulse	1247
Statistics Range	1248
StatusAsString	1249
Step Rise Time	1250
StepData	1251
StepTitle	1252
StimulusValues	1253
Stop Frequency	1254
Stop Frequency_CS	1256
Stop Power	1257
Stop	1258
Stop (Meas)	1259
StopPowerEx	1260
strPort2Pdeembed_S2PFile	1261
strPortMatch_S2PFile	1262
SweepEndMode	1263
SweepHoldOff	1264
SweepOrder	1265
SweepSpeedMode	1266
Sweep Generation Mode	1267
Sweep Time	1268
SweepTimeOption	1269
SweepType (imd)	1270
SweepType (ims)	1272
Sweep Type	1273
System Impedance Z0	1275
SystemName	1276
Target Value	1277
Test Port Power	1278
TestSetType	1280
Text	1281
ThruCalMethod (FCA)	1282
ThruCalMethod	1283
ThruPortList	1284

Title	1286
Title State	1287
Tolerance	1288
TonePower	1289
TonePowerStart	1290
TonePowerStop	1291
TotalIterations	1292
TotalNumberOfPoints	1293
Touchscreen	1294
Trace Math	1295
TraceMax	1296
TraceTitle	1297
TraceTitleState	1298
Tracking	1299
TrackingChannel	1300
TrackingEnable	1301
TrackingManualStepEnable	1302
TrackingStepIndex	1303
Transform Mode	1012
Trigger Delay	1304
TriggerInPolarity	1305
TriggerInType	1307
TriggerOutDuration	1309
TriggerOutInterval	1310
TriggerOutPolarity	1311
TriggerOutPosition	1312
TriggerOutputEnabled	1313
TuningIFBW	1314
TuningMode	1315
TuningSweepInterval	1316
Trigger Mode	1317
Trigger Signal	1318
Trigger Type	1319
Type (calstd)	1320
Type_ts	1321
TZImag Property	1322
TZReal Property	1323
UnusedChannelNumbers	1324

USBPowerMeterCatalog	1325
UseCalWindow	1326
UsedChannelNumbers	1327
Use Power Loss Segments	1328
Use Power Sensor Frequency Limits	1329
User Range	1330
User Range Max	1331
User Range Min	1332
UserDescriptionOfPNA	1333
UserName	1334
UserPresetEnable	1335
Valid	1336
ValidConnectorType	1337
Value element	1338
Values	1339
Velocity Factor	1340
View	1341
Visible	1342
WGCutoffFreq	1343
Width	1344
Window Number	1345
Window State	1346
XAxisAnnotation	1347
XAxis Point Spacing	1348
XAxisStart	1349
XAxisStop	1350
YAxisAnnotation	1351
YScale	1352
Z0	1353
Methods	
Abort	1354
AbortPowerAcquisition	1355
Acquire Cal Standard	1356
Acquire Cal Standard2	1358
AcquireCalConfidenceCheckECALEX	1360
AcquirePowerReadingsEx	1361
AcquireStep	1363
Activate	1364

Activate Marker	1365
Activate Window	1366
Add (channels)	1367
Add (measurement)	1368
Add (naWindows)	1371
Add (PowerLossSegment)	1372
Add (PowerSensorCalFactorSegment)	1373
Add (segments)	1374
Add Testset	1375
Allow All Events	1376
AllowChannelToSweepDuringCalAcquisition	1377
Allow Event Category	1379
Allow Event Message	1380
Allow Event Severity	1381
Apply	1382
ApplyDeltaMatchFromCalSet	1383
ApplyPowerCorrectionValuesEx	1384
ApplySourcePowerCorrectionTo	1385
AssignSourceToRole	1386
AutoPortExtMeasure	1387
AutoPortExtReset	1388
Autoscale	1389
Averaging Restart	1390
Build Hybrid Kit	1391
Calculate Error Coefficients	1392
Calculate	1393
Change Parameter	1395
CheckPower	1399
Clear	1400
Clear fifo	1401
Close CalSet	1402
ComputeErrorTerms	1403
ConfigEnhancedNB2	1404
ConfigEnhancedNBIFAtten	1406
ConfigNarrowBand3	1407
ConfigurationFile	1409
Configurations	1410
Configure	1411

Continuous Sweep	1412
Copy	1413
CopyToChannel	1415
Create SParameterEX	1416
CreateCalSet	1418
CreateCustomCal	1419
CreateCustomCalEx	1420
CreateCustomMeasurementEx	1422
CustomCalConfiguration Method	1427
Create Measurement	1428
DataToMemory	1432
Delete	1433
Delete Marker	1434
Delete All Markers	1435
DeleteCalSet	1436
DeleteConfiguration	1437
Delete ShortCut	1438
Delta Marker	763
Disallow All Events	1439
DiscardChanges	1440
DisplayNAWindowDuringCalAcquisition	1441
DisplayOnlyCalWindowDuringCalAcquisition	1442
Do Print	1443
DoECAL1PortEx	1444
DoECAL2PortEx	1445
DoneCalConfidenceCheckECAL	1447
DoReceiverPowerCal	1448
EnumerateCalSets	1450
Execute	1451
Execute Shortcut	1452
GenerateGlobalDeltaMatchSequence	1453
GenerateErrorTerms	1454
GenerateSteps	1455
GetAllSegments	1456
Get AuxIO	1458
Get Cal Standard	1459
Get CalManager	1460
Get CalSetByGUID	1461

Get CalSetCatalog	1462
Get CalSetUsageInfo	1463
Get Cal Types	1464
Get Complex	1465
GetConverter	1467
Get DataByString	1468
Get Data	1470
Get ECALModuleInfoEx	1473
GetEcalUserCharacterizer	1474
Get ENRData	1475
Get ErrorCorrection	1476
Get Error Term	1477
Get Error Term2	1479
Get Error Term By String	1481
Get Error Term Complex	1483
Get Error Term Complex2	1485
Get Error Term Complex By String	1487
Get Error Term List	1489
Get Error Term List2	1491
Get ExtendedCalInterface	1492
Get ExternalTestSetIO	1493
Get Filter Statistics	1494
Get Guid	1495
get InputVoltageEX	1496
Get Input1	1498
GetIPConfigurationStruct	1499
Get IsolationPaths	1501
Get MaterialHandlerIO	1502
Get NAComplex	1503
Get NumberOfGroups	1505
Get Output	1506
Get Output Voltage	1507
Get OutputVoltage Mode	1508
Get Paired Data	1509
Get Port	1511
Get PortC Data	1512
Get PortNumber	1513
GetRaw2DData	1514



GetRaw2DDataIm	1516
GetRaw2DDataRe	1518
GetRxLevelingConfiguration	1520
GetSourceByRole	1521
GetSourceRoles	1522
Get Reference Marker	1523
Get Required Eterm Names	1524
Get Scalar	1525
Get Shortcut	1527
Get SnPData	1528
Get SnpDataWithSpecifiedPorts	1530
Get SourcePowerCalDataEx	1532
Get SourcePowerCalDataScalarEx	1534
Get Standard	1536
Get Standard By String	1538
Get Standard Complex	1539
Get Standard Complex By String	1541
Get StandardsList	1542
Get Standard List2	1544
Get StandardsForClass	1545
Get StepDescription	1547
Get SupportedALCModes	1548
Get Test Result	1549
Get Trace Statistics	1550
Get X-Axis Values	1551
Get XAxisValues (Meas)	1552
Get X-axis Values Variant	1553
Has CalType	1554
Hold	1556
Hold (All Chans)	1557
Initialize	1558
Initialize ECal	1559
Item	1560
LANConfigurationInitialize	1562
LaunchCalWizard	1563
Launch Dialog	1564
LaunchPowerMeterSettingsDialog	1566
Load Configuration	1567

LoadENRFile	1568
LoadFile	1569
Manual Trigger	1570
MessageText	1571
Next IF Bandwidth	1572
Number of Groups	1573
Open CalSet	1574
Parse	1576
Preset (app and chan)	1577
Previous IF Bandwidth	1578
Print To File	1579
Put Complex	1580
Put Data Complex	1582
PutENRData	1584
Put ErrorTerm	1585
Put ErrorTerm2	1587
Put Error Term By String	1588
Put ErrorTerm Complex	1589
Put ErrorTerm Complex2	1591
Put Error Term Complex By String	1593
Put Formatted Scalar Data	1594
Put NAComplex	1596
Put Output	1598
Put Output Voltage	1599
Put Output Voltage Mode	1600
Put Port	1601
Put PortCData	1603
Put Scalar	1604
Put Shortcut	1606
Put SourcePowerCalDataEx	1607
Put SourcePowerCalDataScalarEx	1609
Put Standard	1611
Put Standard By String	1613
Put Standard Complex	1614
Put Standard Complex By String	1616
Quit	1617
Read Data	1618
Read Raw	1619

Recall	1621
Recall Kits	1622
Remove	1623
Reset	1625
ResetLOFrequency	1626
ResetTuningParameters	1627
Restore Cal Kit Defaults	1628
Restore Cal Kit Defaults All	1629
Restore Defaults	1630
Resume	1631
Save	1632
Save (CalSet)	1634
Save CalSets	1635
SaveCitiDataData	1636
SaveCitiFormattedData	1637
SaveENRFile	1638
Save File	1639
SaveToECal	1640
Save Kits	1641
SearchCompressionPoint	1642
Search Filter Bandwidth	1643
Search Max	1644
Search Min	1645
Search Next Peak	1646
Search Peak Left	1647
Search Peak Right	1648
Search Target	1649
Search Target Left	1650
Search Target Right	1651
SelectCalSet	1652
Set All Segments	1653
Set BBPorts	1655
Set Cal Info	1656
SetCallInfoEx	1658
Set Center	1660
Set CW	1661
SetCWFreq	1662
SetDutPorts	1663

Set Electrical Delay	1664
Set FailOnOverRange	1665
Set IsolationPaths	1666
SetIPConfiguration	1668
Set PowerAcquisitionDevice	1670
Set Frequency LowPass	1671
SetPortMap	1672
Set Reference Level	1674
Set SBPorts	1675
Set SSBPorts	1676
SetupMeasurementsForStep	1677
Set StandardsForClass	1678
Set Start	1680
Set Stop	1681
Show Marker Readout	1682
Show Status Bar	1683
Show Stimulus	1684
Show Table	1685
Show Title Bars	1686
Show Toolbar	1687
Single	1688
Store	1690
StoreConfiguration	1691
StringToNACalClass	1692
StringtoNAErrorTerm2	1694
SweepOnlyCalChannelDuringCalAcquisition	1695
TestsetCatalog	1696
UserPreset	1697
UserPresetLoadFile	1698
UserPresetSaveState	1699
Write Data	1700
Write Raw	1701
WriteSnpFileWithSpecifiedPorts	1703
Events	
OnCalEvent	1705
OnChannelEvent	1707
OnDisplayEvent	1709
OnHardwareEvent	1711

OnMeasurementEvent	1713
OnSCPIEvent	1715
OnSystemEvent	1717
OnUserEvent	1719
Examples	
C Example	1720
CalSet_Examples	1725
Getting Trace Data from the Analyzer	1727
Perform a Guided Cal using COM	1730
Perform a Source Power Cal	1733
Perform an Unguided Cal using COM	1736
Perform an Unknown Thru or TRL Cal	1739
Perform Global Delta Match Cal	1741
Perform a Guided Cal with CSharp	1742
Perform an ECal	1746
Perform an ECal User Characterization	1748
Writing Cal Set Data using COM	1753
Upload a Source Power Cal	1755
Upload Segment Table	1757
Create and Cal an SMC Measurement	1761
Create and Cal a VMC Measurement	1763
Create an SMC Fixed Output Meas	1766
Create a Pulsed Measurement	1768
Create a Balanced Measurement	1770
Create a PMAR Device and Measurement	1774
Create a Wideband Pulsed Measurement using the PNA-X	1776
Create an IM Spectrum Measurement	1778
Create an iTMSA Measurement	1780
Create_and_Cal_a_Gain_Compression_Measurement	1781
Create and Cal a NoiseFigure Measurement	1786
Create and Cal an IMD Measurement	1789
Perform an ECAL Confidence Check	1793
ENR File Management Example	1795
Events with C	1797
FOM Examples	1799
Limit Line Testing Example with COM	1801
Modify Display Colors	1802
E5091Testset Control	1803

Errors and the SCPIStringParser Object	1804
External Testset Control	1806
PathConfiguration Example	1808
PNA-X Create a Pulsed Measurement	1810
Setup Basic Measurements	1814
Setup Compression Marker	1816
Setup FastCW and FIFO	1817
Setup Receiver Leving	1819
Show Custom Window during Calibration	1821
Events Example	1823
Upload_Segment_Table_in_C	1824
Using C#	1831

## Concepts

Configure for COM-DCOM Programming	1833
COM Fundamentals	1838
Getting a Handle to an Object	1842
Collections in the Analyzer	1845
COM Data Types	1846
PNA Automation Interfaces	1848
Working with the Analyzer's Events	1850
Read and Write Calibration Data using COM	1854
C and the COM Interface	1856
Using .NET	1859

## SCPI

### Commands

SCPI Command Tree	1861
Common Commands	1864
Abort	1867
Calculate	
Correction	1868
Custom	1876
Data	1881
Equation	1889
Filter	1892
Format	1898
FSimulator	1901
Function	1903
GCData	1908

Limit	1913
Marker	1921
Math	1941
Mixer	1943
Normalize	1944
Offset	1947
Parameter	1950
RData	1959
Smoothing	1961
Transform	1964
Control	1973
Display	1985
Format	2005
Hardcopy	2008
Initiate	2021
Memory	2024
Output	2037
Route	2039
Sense	
Average	2040
Bandwidth	2043
Class	2045
Correction	2046
Cal Kit	2074
Cal Stds	2087
Cal Sets	2109
Extensions	2124
Guided Cal	2138
IMD	2163
Session	2169
SMC	2175
VMC	2187
Couple	2198
FOM	2200
FOMSegment	2210
Frequency	2221
Gain Compression	2225
IF	2241

IF (PNA-X)	2247
IMD	2256
IMS	2276
Mixer	2289
MixerEmbedLO	2312
Multiplexer	2326
Noise	2337
Offset	2348
Path	2352
Power	2357
Pulse	2358
Roscillator	2365
Segment	2366
Sweep	2379
XAxis	2390
Source	2391
Source Correction	2415
Status	2433
System	2449
Trigger	2475
Examples	
Catalog Measurements	2490
Create an FOM Measurement	2491
Create an S-Parameter Measurement	2494
Create a Balanced Measurement	2495
Channels, Windows, and Measurements using GPIB	2500
Setup Sweep Parameters	2502
Setup the Display	2503
Triggering the PNA	2505
GPIB using Visual C	2511
Guided 2-Port or 4-Port Cal	2515
Guided 2-Port Comprehensive Cal	2518
Guided ECal	2524
Guided Mechanical Cal	2526
Guided 1-Port on Port 2	2528
Guided TRL Calibration	2530
Guided Unknown Thru or TRL Cal	2532
Global Delta Match Cal	2534



Unguided ECAL	2535
Unguided 2-Port Mech Cal	2537
Unguided 1-Port Cal on Port 2	2539
Unguided 2-Port Cal on a 4-Port PNA	2541
Unguided_Cal_on_Multiple_Channels	2550
ECAL Confidence Check	2554
Source and Receiver Power Cal	2557
Upload a Source Power Cal	2566
Perform a Sliding Load Cal	2570
Load Eterms during Cal Sequence	2571
Create New Cal Kit	2572
Modify a Calibration Kit	2578
Create and Cal a VMC Measurement	2580
Create and Cal an SMC Measurement	2583
Create an SMC Fixed Output Measurement	2586
Create and Cal a GCA Measurement	2589
Create and Cal a Noise Figure Measurement	2595
Show Custom Window during Calibration	2598
Getting and Putting Data	2601
Getting and Putting Data (definite block transfers)	2603
Control an External Test Set	2606
Transfer Data using GPIB	2608
Establish a VISA Session	2610
Status Reporting	2612
Create a Custom Power Meter Driver	2614
GPIB Pass Through	2618
PNA as Controller and Talker/Listener	2619
Send SCPI Commands using a Socket Client	2621
Concepts	
GP-IB Fundamentals	2629
The Rules and Syntax of SCPI Commands	2634
How to Configure for GPIB, SCPI, and SICL	2638
Getting Data from the Analyzer	2641
Understanding Command Synchronization	2645
Calibrating the PNA Using SCPI	2650
The PNA as a USB Device	2656
Reading the Analyzer's Status Registers	2658
Configure for VISA and SICL	2662

VEE Examples	
VEE Pro Runtime	2665
Basic Control VEE	2666
ECal with Confidence Check using VEE	2668
Data Access Map	
DataMapSet	2669
Rear Panel IO Connectors	
Interface Control	2670
Auxiliary IO connector	2676
External TestSet IO Connector	2682
Material Handler IO Connector	2688
Pulse IO (PNA-X)	2701
Power IO (PNA-X)	2703
Programming Guide	441
New Programming Commands	2706
COM versus SCPI	2728
Remotely Specifying a Source Port	2730
Shut Down the PNA Remotely	2732
LXI Compliance	65
Using Macros	213
Applications	
Frequency Converter Application (Opts 082 / 083)	
Known Issues	2733
Overview	2734
Using the Frequency Converter Application	2736
Calibrations	2746
Configure a Mixer	2768
Configure an External LO Source	2777
How to make a VMC Fixed Output Measurement	2786
How to make an SMC Fixed Output Measurement	2795
Embedded LO (Opt 084)	2803
Characterize Adaptor Macro	366
SMC with a Booster Amp	2809
Frequency Offset (Opt 080)	
Frequency Offset (Option 080)	2812
Frequency Converting Device Measurements	2822
Frequency Offset Calibration	2823
Conversion Loss	2825

Conversion Compression	2829
Isolation	2831
Harmonic Distortion	2834
Return Loss and VSWR	2836
Antenna Features	2838
Gain Compression Application (Opt 086)	2842
Gain Compression Cal	2867
iTMSA (Opt 460)	2871
Noise Figure (Opt 029)	2880
Noise Figure Cal	2900
Swept IMD and IM Spectrum (Opt 087)	
Swept IMD and IM Spectrum Concepts	2907
Swept IMD	2917
IM Spectrum	2930
Swept IMDx for Converters	2939
IMx Spectrum for Converters	2955
Time Domain (Opt 010)	2966
Integrated Pulse Application (Opt 008)	2982
Pulsed Application (Opt H08)	2990
WideBand Pulsed App	3004
Networking the PNA	
Drive Mapping	433
Connecting the PNA to a PC	3005
Easy vs Secure Configuration	3008
Changing Network Client	3009
Product Support	
Troubleshoot the PNA	3010
List of Error Messages	3014
About Error Messages	3076
Accessories	3080
USB to GPIB Adapter	3087
Firmware Update	3090
PNA Configurations and Options	3096
Option Enable	3107
Instrument Calibration	3111
Other Resources	3113
SCPI Errors	3114
Technical Support	3125

Diagnostic Tools and Adjustments	
3.8 GHz Frequency Adjust	3130
10 MHz Reference Adjust	3132
Display Test	3134
LO Power Adjust	3135
Offset LO Power Adjustment	3137
Operators Check	3138
Option H11 Verification	3141
Phase-Lock IF Gain Adjust	3144
System Verification	3146
Source Adjustment	3157
Receiver Cal	3160
Receiver Display	3163
IF Access	
IF Access User Interface Settings	3164
External Test Head Configuration	3171
PNA-X IF Path Configuration	3164
N5264A Measurement Receiver	3180
System Settings	
Configure an External Device	2777
Configure a Power Meter As Receiver	3184
E5091 Test Set Control	3190
Error Messages	3076
External Testset Control	3194
Display Colors	3206
Frequency Blanking	53
Interface Control	2670
Mechanical Devices	3210
PNA Preferences	59
Power Limit and Power Offset	3213
System Impedance	3216
Synchronize an External PSG Source	3218
Rear Panel IO Connectors	
Interface Control	2670
Auxiliary IO connector	2676
External TestSet IO Connector	2682
Material Handler IO Connector	2688
Pulse IO (PNA-X)	2701

Power IO (PNA-X)	2703
Tutorials	
App Notes	3222
Network Analyzer Basics	3224
Connector Care	3225
ESD Protection	3236
Measurements	
Absolute Output Power	3237
AM-PM Conversion	3239
Amplifier Measurements	3244
Antenna Measurements	3247
Balanced Measurements	3250
Complex Impedance	3256
Comparing the PNA Delay Functions	3259
Deviation from Linear Phase	3261
External Source Control	3218
Gain and Flatness	3264
Gain Compression	3267
Group Delay	3272
High-Gain Amplifier Measurements	3279
High-Power Measurements using a PNA	3281
High Power Measurements using a PNA-X	3282
Phase Measurements	3284
Reverse Isolation	3287
Reflection Measurements	3290
Time Domain Measurements	2966
Specifications	3294
Glossary	3295

---

## What's New in PNA Code Version A.09.22

---

- [Use any PNA-X Ports with Noise Figure Opt 028.](#)

### [See New 9.22 Programming Commands](#)

This version includes features from all versions listed below.

To check your PNA code version, click **Help**, then **About Network Analyzer**

---

## What's New in PNA Code Version A.09.20

### Application Enhancements

- [Integrated Pulse Measurements](#) (Opt 008)
- [Noise Figure using Standard PNA Receiver](#) (Opt 028)
- [Noise Figure on N5244A/45A](#) (Opt H29)
- [Edge and Level Trigger in Pulse](#)
- [Exclude SC12 Sweep for SMC](#) (Opt 082/083)
- [Include Phase with SMC](#) (Opt 082/083)
- [Fixturing in Apps](#)
- [Max Output Power for GCA](#) (Opt 086)

### Standard Measurements - available on all Models / Options

- [Mechanical Device conflicts cause Channel Block \(NOT Channel Hold\)](#)
- [PSAT Marker and Power Normal Operating Point Marker](#)
- [Group Delay Aperture Setting](#)
- [Active Background Display Color](#)
- [Solid or Dotted Grid Lines](#)
- [Point Sweep on PNA "C" Models](#)
- [Fixture Power Compensation](#)
- [Sweep Delay](#)
- [Uncertainty equations using RSS Computations](#)
- [Preset Power Preference Setting](#)

- [Use Last Receiver Leveling Correction for SPC](#)
  - Data Save Enhancements
    - [Recall .SNP files to view as trace](#)
    - ["Save Data As" Dialog](#)
    - [Save Balanced Data as SNP files](#)
  - Characterize Adaptor Macro Rev. A.02.10
    - [Reverse S2P](#)
    - [Load the PNA Power Loss Table from an existing S2P file](#)
- 

#### **What's New in PNA Code Version A.09.10**

- [Noise Figure on Converters \(NFX\)](#)
  - [AgileUpdate for Customer Releases](#)
  - [New Help, About Capabilities](#)
- 

#### **What's New in PNA Code Version A.09.00**

- [External Device Configuration](#)
- [Power Meter as Receiver](#) (PMAR)
- [Power Offsets and Limits](#)
- [Display and Print Colors](#)
- [Scale Coupling](#)
- [Mechanical Device Settings](#)
- [Device side USB](#)
- [Increased Number of Channels to 200](#)
- [ECal User Chars Saved to PNA Disk Memory](#)

#### **Application Enhancements**

- [GCA Compression Analysis](#)

- [GCA Compression from Saturation](#)
  - [Receiver Leveling on IMD, FCA, and GCA](#)
  - [Embedded LO on SMC and IMDx](#)
  - [Limited Port Mapping on IMD](#)
  - [Point Averaging on FCA and IMD](#)
  - [SMC Measurements with mmWave Modules](#)
  - [mmWave Measurements with no Test Set](#)
- 

#### **What's New in PNA Code Version A.08.60**

- [New 40 GHz and 50 GHz PNA-X Models](#)
- 

#### **What's New in PNA Code Version A.08.55**

- [IMDx \(Swept and Spectrum\) for Converters](#)
  - [ADC Measurements in a Gain Compression channel.](#)
  - [New 13.5 GHz PNA-X Model](#)
- 

#### **What's New in PNA Code Version A.08.50**

- [Fast Antenna Features for the PNA-X](#)
- [Receiver Leveling](#)
- [Phase Sweep](#) in iTMSA
- [Up to 25 User Macros](#)
- [Gain Compression Marker](#)
- [Port Extensions enhancements](#)
- [Electrical Delay enhancements](#)
- [Extra Security enhancement](#)
- Save [\\*.CSV](#) and [\\*.MDF](#) File Types
- Noise Figure App enhancements:



- [Scalar Noise Figure measurement](#)
  - [Incident Noise Power parameters](#)
  - MM Module enhancements:
    - [Power Level Control](#)
    - Supports [iTMSA](#)
  - [Max point count to 32,001](#)
  - [Faster Power Sweeps](#)
- 

#### **What's New in PNA Code Version A.08.35**

- [New N5264A model](#)
- 

#### **What's New in PNA Code Version A.08.33**

- [IMD Application](#) (Opt 087)
  - [Fast Sweep Mode](#)
  - [New Equation Editor functions](#)
  - [20001 Segments in Power Loss Table](#)
  - [Data Format Units](#)
  - [Marker=>CW Freq Function](#)
  - [Up to 12 User Characterizations](#)
  - [Characterize Adaptor Macro 2.0](#)
- 

#### **What's New in PNA Code Version A.08.20**

- [iTMSA](#)
- [User Preferences dialog](#)
- [Uncoupled Power Sweep](#)
- [Equation Editor Import Functions](#)

- [Wider Traces Preference](#)
  - [24 Traces per Window](#)
  - [LXI Compliance](#)
  - [GCA Enhancements](#)
  - [FCA - selectable ports](#)
  - [Support for N5261A and N5262A MM test sets](#)
  - [cXL Code Translation Software - Quick Link](#)
- 

#### **What's New in PNA Code Version A.08.00**

- [Noise Figure Application \(Opt 029\)](#)
  - [Gain Compression Application \(Opt 086\)](#)
  - ['Sweep' Trigger Mode](#)
  - [Custom Cal Window settings](#) (remote only)
  - [New Equation Editor Functions](#)
  - [Minimum Number of Points = 1](#)
- 

Last modified:

9/28/06    Cross-browser

END-USER LICENSE AGREEMENT FOR MICROSOFT SOFTWARE  
MICROSOFT WINDOWS XP PROFESSIONAL EDITION SERVICE PACK 3

**IMPORTANT—READ CAREFULLY:** This End-User License Agreement (“EULA”) is a legal agreement between you (either an individual or a single entity) and Microsoft Corporation or one of its affiliates (“Microsoft”) for the Microsoft software that accompanies this EULA, which includes computer software and may include associated media, printed materials, “online” or electronic documentation, and Internet-based services (“Software”). An amendment or addendum to this EULA may accompany the Software. **YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE. IF YOU DO NOT AGREE, DO NOT INSTALL, COPY, OR USE THE SOFTWARE; YOU MAY RETURN IT TO YOUR PLACE OF PURCHASE FOR A FULL REFUND, IF APPLICABLE.**

**1. GRANT OF LICENSE.** Microsoft grants you the following rights provided that you comply with all terms and conditions of this EULA:

**1.1 Installation and use.** You may install, use, access, display and run one copy of the Software on a single computer, such as a workstation, terminal or other device (“Workstation Computer”). The Software may not be used by more than two (2) processors at any one time on any single Workstation Computer.

**1.2 Mandatory Activation.** The license rights granted under this EULA are limited to the first thirty (30) days after you first install the Software unless you supply information required to activate your licensed copy in the manner described during the setup sequence of the Software. You can activate the Software through the use of the Internet or telephone; toll charges may apply. You may also need to reactivate the Software if you modify your computer hardware or alter the Software. There are technological measures in this Software that are designed to prevent unlicensed use of the Software. Microsoft will use those measures to confirm you have a legally licensed copy of the Software. If you are not using a licensed copy of the Software, you are not allowed to install the Software or future Software updates. Microsoft will not collect any personally identifiable information from your Workstation Computer during this process.

**1.3 Device Connections.** You may permit a maximum of ten (10) computers or other electronic devices (each a “Device”) to connect to the Workstation Computer to utilize one or more of the following services of the Software: File Services, Print Services, Internet Information Services, Internet Connection Sharing and telephony services. The ten connection maximum includes any indirect connections made through “multiplexing” or other software or hardware which pools or aggregates connections. This ten connection maximum does not apply to other uses of the Software, such as synchronizing data between a Device and the Workstation Computer, provided only one user uses, accesses, displays or runs the Software at any one time. This Section 1.3 does not grant you rights to access a Workstation Computer Session from any Device. A “Session” means any use of the Software that enables functionality similar to that available to an end user who is interacting with the Workstation Computer through any combination of input, output and display peripherals.

**1.4 Remote Desktop/Remote Assistance/NetMeeting.** The Software contains Remote Desktop, Remote Assistance, and NetMeeting technologies that enable the Software or applications installed on the Workstation Computer (sometimes referred to as a host device) to be accessed remotely from other Devices. You may use the Software’s Remote Desktop feature (or other software which provides similar functionality for a similar purpose) to access a Workstation Computer Session from any Device provided you acquire a separate Software license for that Device. As an exception to this rule, the person who is the single primary user of the Workstation Computer may access a Workstation Computer Session from any Device without acquiring an additional Software license for that Device. When you are using Remote Assistance or NetMeeting (or other software which provides similar functionality for a similar purpose) you may share a Session with other users without any limit on the number of Device connections and without acquiring additional licenses for the Software. For Microsoft and non-Microsoft applications, you should consult the license agreement accompanying the applicable software or contact the applicable licensor to determine whether use of the software with Remote Desktop, Remote Assistance, or NetMeeting is permitted without an additional license.

**1.5 Storage/Network Use.** You may also store or install a copy of the Software on a storage device, such as a

network server, used only to install or run the Software on your other Workstation Computers over an internal network; however, you must acquire and dedicate an additional license for each separate Workstation Computer on or from which the Software is installed, used, accessed, displayed or run. Except as otherwise permitted by the NetMeeting and Remote Assistance features described above, a license for the Software may not be shared or used concurrently on different Workstation Computers.

**2. AUTOMATIC INTERNET-BASED SERVICES.** The Software features described below are enabled by default to connect via the Internet to Microsoft computer systems automatically, without separate notice to you. You consent to the operation of these features, unless you choose to switch them off or not use them. Microsoft does not obtain personal information through any of these features. For more information about these features, please see your Software documentation, the Microsoft online support site, or the privacy statement at <http://go.microsoft.com/fwlink/?LinkId=25243>.

**2.1 Windows Update Features.** If you connect hardware to your Workstation Computer, it may not have the drivers needed to communicate with that hardware. The Software's update feature can obtain the correct drivers from Microsoft and install them on your device. You can switch this update feature off.

**2.2 Web Content Features.** Under the Software's default configuration, if you are connected to the Internet, several features of the Software are enabled by default to retrieve content from Microsoft computer systems and display it to you. When you activate such a feature, it uses standard Internet protocols, which transmit the type of operating system, browser and language code of your Workstation Computer to the Microsoft computer system so that the content can be viewed properly from your Workstation Computer. These features only operate when you activate them, and you may choose to switch them off or not use them. Examples of these features include Windows Catalog, Search Assistant, and the Headlines and Search features of Help and Support Center.

**2.3 Digital Certificates.** The Software uses digital certificates based on the x.509 standard. These digital certificates confirm the identity of Internet users sending x.509 standard encrypted information. The software retrieves certificates and updates certificate revocation lists. These security features operate only when you use the Internet.

**2.4 Auto Root Update.** The Auto Root Update feature updates the list of trusted certificate authorities. You can switch off the Auto Root Update feature.

**2.5 Windows Media Player.** Some features of Windows Media Player automatically contact Microsoft computer systems if you use Windows Media Player or specific features of it: features that (A) check for new codecs if your Workstation Computer does not have the correct ones for content you attempt to play (this feature may be switched off), and (B) check for new versions of Windows Media Player (this feature will operate only when you are using Windows Media Player).

**2.6 Windows Media Digital Rights Management.** Content providers are using the digital rights management technology for Windows Media contained in this Software ("WM-DRM") to protect the integrity of their content ("Secure Content") so that their intellectual property, including copyright, in such content is not misappropriated. Portions of this Software and third party applications such as media players use WM-DRM to play Secure Content ("WM-DRM Software"). If the WM-DRM Software's security has been compromised, owners of Secure Content ("Secure Content Owners") may request that Microsoft revoke the WM-DRM Software's right to copy, display and/or play Secure Content. Revocation does not alter the WM-DRM Software's ability to play unprotected content. A list of revoked WM-DRM Software is sent to your Workstation Computer whenever you download a license for Secure Content from the Internet. Microsoft may, in conjunction with such license, also download revocation lists onto your Workstation Computer on behalf of Secure Content Owners. Secure Content Owners may also require you to upgrade some of the WMDRM components in this Software ("WM-DRM Upgrades") before accessing their content. When you attempt to play such content, WM-DRM Software built by Microsoft will notify you that a WM-DRM Upgrade is required and then ask for your consent before the WM-DRM Upgrade is downloaded. WM-DRM Software built by third parties may do the same. If you decline the upgrade, you will not be able to access content that requires the WM-DRM Upgrade; however, you will still be able to access unprotected content and Secure Content that does not require the upgrade. WM-DRM features that access the Internet, such as acquiring new licenses and/or performing a required WM-DRM Upgrade, can be switched off. When these features are switched off, you will still be able to play Secure Content if you have a valid license for such content already stored on your

Workstation Computer.

3. **RESERVATION OF RIGHTS AND OWNERSHIP.** Microsoft reserves all rights not expressly granted to you in this EULA. The Software is protected by copyright and other intellectual property laws and treaties. Microsoft or its suppliers own the title, copyright, and other intellectual property rights in the Software. The Software is licensed, not sold.

4. **LIMITATIONS ON REVERSE ENGINEERING, DECOMPILE, AND DISASSEMBLY.** You may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

5. **NO RENTAL/COMMERCIAL HOSTING.** You may not rent, lease, lend or provide commercial hosting services with the Software.

6. **LINKS TO THIRD PARTY SITES.** You may link to third party sites through the use of the Software. The third party sites are not under the control of Microsoft, and Microsoft is not responsible for the contents of any third party sites, any links contained in third party sites, or any changes or updates to third party sites. Microsoft is not responsible for webcasting or any other form of transmission received from any third party sites. Microsoft is providing these links to third party sites to you only as a convenience, and the inclusion of any link does not imply an endorsement by Microsoft of the third party site.

7. **ADDITIONAL SOFTWARE/SERVICES.** This EULA applies to updates, supplements, add-on components, product support services, or Internet-based services components, of the Software that you may obtain from Microsoft after the date you obtain your initial copy of the Software, unless you accept updated terms or another agreement governs. Microsoft reserves the right to discontinue any Internet-based services provided to you or made available to you through the use of the Software.

8. **UPGRADES.** To use Software identified as an upgrade, you must first be licensed for the software identified by Microsoft as eligible for the upgrade. After upgrading, you may no longer use the software that formed the basis for your upgrade eligibility.

9. **NOT FOR RESALE SOFTWARE.** Software identified as "Not For Resale" or "NFR," may not be sold or otherwise transferred for value, or used for any purpose other than demonstration, test or evaluation.

10. **ACADEMIC EDITION SOFTWARE.** To use Software identified as "Academic Edition" or "AE," you must be a "Qualified Educational User." For qualification-related questions, please contact the Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399 or the Microsoft subsidiary serving your country.

11. **NOTICES REGARDING THE MPEG-4 VISUAL STANDARD.** The Software includes MPEG-4 visual decoding technology. This technology is a format for data compression of video information. For this technology, MPEG LA, L.L.C. requires this notice: USE OF THIS PRODUCT IN ANY MANNER THAT COMPLIES WITH THE MPEG-4 VISUAL STANDARD IS PROHIBITED, EXCEPT FOR USE DIRECTLY RELATED TO (A) DATA OR INFORMATION (i) GENERATED BY AND OBTAINED WITHOUT CHARGE FROM A CONSUMER NOT THEREBY ENGAGED IN A BUSINESS ENTERPRISE, AND (ii) FOR PERSONAL USE ONLY; AND (B) OTHER USES SPECIFICALLY AND SEPARATELY LICENSED BY MPEG LA, L.L.C. If you have questions regarding this notice, please contact MPEG LA, L.L.C., 250 Steele Street, Suite 300, Denver, Colorado 80206; Telephone 303 331.1880; FAX 303 331.1879; <http://www.mpegla.com>.

12. **EXPORT RESTRICTIONS.** You acknowledge that the Software is subject to U.S. export jurisdiction. You agree to comply with all applicable international and national laws that apply to the Software, including the U.S. Export Administration Regulations, as well as end-user, enduse, and destination restrictions issued by U.S. and other governments. For additional information see <http://www.microsoft.com/exporting>.

13. **END USER PROOF OF LICENSE.** If you acquired the Software on a compact disc or other media, a genuine Microsoft "Proof of License" label with a genuine copy of the software identifies a licensed copy of the Software. To be valid, the label must appear on Microsoft software packaging. If you receive the label separately, it is invalid. You should keep the packaging that has the label on it to prove that you are licensed to use the Software.

14. **SOFTWARE TRANSFER.** Internal. You may move the Software to a different Workstation Computer. After the

transfer, you must completely remove the Software from the former Workstation Computer. Transfer to Third Party. The initial user of the Software may make a one-time permanent transfer of this EULA and Software to another end user, provided the initial user retains no copies of the Software. This transfer must include the Software and the Proof of License label. The transfer may not be an indirect transfer, such as a consignment. Prior to the transfer, the end user receiving the Software must agree to all the EULA terms.

15. **TERMINATION.** Without prejudice to any other rights, Microsoft may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the Software and all of its component parts.

16. **NOTICE REGARDING SECURITY.** To help protect against breaches of security and malicious software, periodically back up your data and system information, use security features such as firewalls, and install and use security updates.

17. **LIMITED WARRANTY FOR SOFTWARE ACQUIRED IN THE US AND CANADA.** Microsoft warrants that the Software will perform substantially in accordance with the accompanying materials for a period of ninety (90) days from the date of receipt. If an implied warranty or condition is created by your state/jurisdiction and federal or state/provincial law prohibits disclaimer of it, you also have an implied warranty or condition, **BUT ONLY AS TO DEFECTS DISCOVERED DURING THE PERIOD OF THIS LIMITED WARRANTY (NINETY DAYS).** AS TO ANY DEFECTS DISCOVERED AFTER THE NINETY-DAY PERIOD, THERE IS NO WARRANTY OR CONDITION OF ANY KIND. Some states/jurisdictions do not allow limitations on how long an implied warranty or condition lasts, so the above limitation may not apply to you. Any supplements or updates to the Software, including without limitation, any (if any) service packs or hot fixes provided to you after the expiration of the ninety day Limited Warranty period are not covered by any warranty or condition, express, implied or statutory. **LIMITATION ON REMEDIES; NO CONSEQUENTIAL OR OTHER DAMAGES.** Your exclusive remedy for any breach of this Limited Warranty is as set forth below. Except for any refund elected by Microsoft, **YOU ARE NOT ENTITLED TO ANY DAMAGES, INCLUDING BUT NOT LIMITED TO CONSEQUENTIAL DAMAGES,** if the Software does not meet Microsoft's Limited Warranty, and, to the maximum extent allowed by applicable law, even if any remedy fails of its essential purpose. The terms of Section 19 ("Exclusion of Incidental, Consequential and Certain Other Damages") are also incorporated into this Limited Warranty. Some states/jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. This Limited Warranty gives you specific legal rights. You may have other rights which vary from state/jurisdiction to state/jurisdiction. **YOUR EXCLUSIVE REMEDY.** Microsoft's and its suppliers' entire liability and your exclusive remedy for any breach of this Limited Warranty or for any other breach of this EULA or for any other liability relating to the Software shall be, at Microsoft's option from time to time exercised subject to applicable law, (a) return of the amount paid (if any) for the Software, or (b) repair or replacement of the Software, that does not meet this Limited Warranty and that is returned to Microsoft with a copy of your receipt. You will receive the remedy elected by Microsoft without charge, except that you are responsible for any expenses you may incur (e.g. cost of shipping the Software to Microsoft). This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication, abnormal use or a virus. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer, and Microsoft will use commercially reasonable efforts to provide your remedy within a commercially reasonable time of your compliance with Microsoft's warranty remedy procedures. Outside the United States or Canada, neither these remedies nor any product support services offered by Microsoft are available without proof of purchase from an authorized international source. To exercise your remedy, contact: Microsoft, Attn. Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399, or the Microsoft subsidiary serving your country.

18. **DISCLAIMER OF WARRANTIES.** The Limited Warranty that appears above is the only express warranty made to you and is provided in lieu of any other express warranties or similar obligations (if any) created by any advertising, documentation, packaging, or other communications. Except for the Limited Warranty and to the maximum extent permitted by applicable law, Microsoft and its suppliers provide the Software and support services (if any) **AS IS AND WITH ALL FAULTS**, and hereby disclaim all other warranties and conditions, whether express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of reliability or availability, of accuracy or completeness of



responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the Software, and the provision of or failure to provide support or other services, information, software, and related content through the Software or otherwise arising out of the use of the Software. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CORRESPONDENCE TO DESCRIPTION OR NONINFRINGEMENT WITH REGARD TO THE SOFTWARE.

19. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, PUNITIVE, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT OR OTHER SERVICES, INFORMATION, SOFTWARE, AND RELATED CONTENT THROUGH THE SOFTWARE OR OTHERWISE ARISING OUT OF THE USE OF THE SOFTWARE, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), MISREPRESENTATION, STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

20. LIMITATION OF LIABILITY AND REMEDIES. Notwithstanding any damages that you might incur for any reason whatsoever (including, without limitation, all damages referenced herein and all direct or general damages in contract or anything else), the entire liability of Microsoft and any of its suppliers under any provision of this EULA and your exclusive remedy hereunder (except for any remedy of repair or replacement elected by Microsoft with respect to any breach of the Limited Warranty) shall be limited to the greater of the actual damages you incur in reasonable reliance on the Software up to the amount actually paid by you for the Software or US\$5.00. The foregoing limitations, exclusions and disclaimers (including Sections 17, 18, and 19) shall apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

21. U.S. GOVERNMENT LICENSE RIGHTS. All Software provided to the U.S. Government pursuant to solicitations issued on or after December 1, 1995 is provided with the commercial license rights and restrictions described elsewhere herein. All Software provided to the U.S. Government pursuant to solicitations issued prior to December 1, 1995 is provided with "Restricted Rights" as provided for in FAR, 48 CFR 52.227-14 (JUNE 1987) or DFAR, 48 CFR 252.227-7013 (OCT 1988), as applicable.

22. APPLICABLE LAW. If you acquired this Software in the United States, this EULA is governed by the laws of the State of Washington. If you acquired this Software in Canada, unless expressly prohibited by local law, this EULA is governed by the laws in force in the Province of Ontario, Canada; and, in respect of any dispute which may arise hereunder, you consent to the jurisdiction of the federal and provincial courts sitting in Toronto, Ontario. If you acquired this Software in the European Union, Iceland, Norway, or Switzerland, then local law applies. If you acquired this Software in any other country, then local law may apply.

23. ENTIRE AGREEMENT; SEVERABILITY. This EULA (including any addendum or amendment to this EULA which is included with the Software) is the entire agreement between you and Microsoft relating to the Software and the support services (if any) and they supersede all prior or contemporaneous oral or written communications, proposals and representations with respect to the Software or any other subject matter covered by this EULA. To the extent the terms of any Microsoft policies or programs for support services conflict with the terms of this EULA, the terms of this EULA shall control. If any provision of this EULA is held to be void, invalid, unenforceable or illegal, the other provisions shall continue in full force and effect.

Si vous avez acquis votre produit Microsoft au CANADA, la garantie limitée suivante vous concerne :

#### GARANTIE LIMITÉE

Microsoft garantit que le Logiciel fonctionnera conformément aux documents inclus pendant une période de 90

jours suivant la date de réception. Si une garantie ou condition implicite est créée par votre État ou votre territoire et qu'une loi fédérale ou provinciale ou État en interdit le déni, vous jouissez également d'une garantie ou condition implicite, MAIS UNIQUEMENT POUR LES DÉFAUTS DÉCOUVERTS DURANT LA PÉRIODE DE LA PRÉSENTE GARANTIE LIMITÉE (QUATRE-VINGT-DIX JOURS). IL N'Y A AUCUNE GARANTIE OU CONDITION DE QUELQUE NATURE QUE CE SOIT QUANT AUX DÉFAUTS DÉCOUVERTS APRÈS CETTE PÉRIODE DE QUATRE-VINGT-DIX JOURS. Certains États ou territoires ne permettent pas de limiter la durée d'une garantie ou condition implicite de sorte que la limitation ci-dessus peut ne pas s'appliquer à vous. Tous les suppléments ou toutes les mises à jour relatifs au Logiciel, notamment, les ensembles de services ou les réparations à chaud (le cas échéant) qui vous sont fournis après l'expiration de la période de quatre-vingt-dix jours de la garantie limitée ne sont pas couverts par quelque garantie ou condition que ce soit, expresse ou implicite. LIMITATION DES RECOURS; ABSENCE DE DOMMAGES INDIRECTS OU AUTRES. Votre recours exclusif pour toute violation de la présente garantie limitée est décrit ci-après. Sauf pour tout remboursement au choix de Microsoft, si le Logiciel ne respecte pas la garantie limitée de Microsoft et, dans la mesure maximale permise par les lois applicables, même si tout recours n'atteint pas son but essentiel, VOUS N'AVEZ DROIT À AUCUNS DOMMAGES, NOTAMMENT DES DOMMAGES INDIRECTS. Les modalités de la clause «Exclusion des dommages accessoires, indirects et de certains autres dommages » sont également intégrées à la présente garantie limitée. Certains États ou territoires ne permettent pas l'exclusion ou la limitation des dommages indirects ou accessoires de sorte que la limitation ou l'exclusion ci-dessus peut ne pas s'appliquer à vous. La présente garantie limitée vous donne des droits légaux spécifiques. Vous pouvez avoir d'autres droits qui peuvent varier d'un territoire ou d'un État à un autre. VOTRE RECOURS EXCLUSIF. L'obligation intégrale de Microsoft et de ses fournisseurs et votre recours exclusif seront, selon le choix de Microsoft de temps à autre sous réserve de toute loi applicable, a) le remboursement du prix payé, le cas échéant, pour le Logiciel ou b) la réparation ou le remplacement du Logiciel qui ne respecte pas la présente garantie limitée et qui est retourné à Microsoft avec une copie de votre reçu. Vous recevrez la compensation choisie par Microsoft, sans frais, sauf que vous êtes responsable des dépenses que vous pourriez engager (p. ex., les frais d'envoi du Logiciel à Microsoft). La présente garantie limitée est nulle si la défectuosité du Logiciel est causée par un accident, un usage abusif, une mauvaise application, un usage anormal ou un virus. Tout Logiciel de remplacement sera garanti pour le reste de la période de garantie initiale ou pendant trente (30) jours, selon la plus longue entre ces deux périodes. À l'extérieur des États-Unis ou du Canada, ces recours ou l'un quelconque des services de soutien technique offerts par Microsoft ne sont pas disponibles sans preuve d'achat d'une source internationale autorisée. Pour exercer votre recours, vous devez communiquer avec Microsoft et vous adresser au Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399, ou à la filiale de Microsoft de votre pays.

DÉNI DE GARANTIES. La garantie limitée mentionnée ci-dessus constitue la seule garantie expresse qui vous est donnée et remplace toutes autres garanties expresses (s'il en est) mentionnées dans un document ou sur un emballage. Sauf en ce qui a trait à la garantie limitée et dans la mesure maximale permise par les lois applicables, le Logiciel et les services de soutien technique (le cas échéant) sont fournis TELS QUELS ET AVEC TOUS LES DÉFAUTS par Microsoft et ses fournisseurs, lesquels par les présentes dénie toutes autres garanties et conditions expresses, implicites ou en vertu de la loi, notamment (le cas échéant) les garanties, devoirs ou conditions implicites de qualité marchande, d'adaptation à un usage particulier, d'exactitude ou d'exhaustivité des réponses, des résultats, des efforts déployés selon les règles de l'art, d'absence de virus et de négligence, le tout à l'égard du Logiciel et de la prestation des services de soutien technique ou de l'omission d'une telle prestation.

PAR AILLEURS, IL N'Y A AUCUNE GARANTIE OU CONDITION QUANT AU TITRE DE PROPRIÉTÉ, À LA JOUISSANCE OU LA POSSESSION PAISIBLE, À LA CONCORDANCE À UNE DESCRIPTION NI QUANT À UNE ABSENCE DE CONTREFAÇON CONCERNANT LE LOGICIEL. EXCLUSION DES DOMMAGES ACCESSOIRES, INDIRECTS ET DE CERTAINS AUTRES DOMMAGES. DANS LA MESURE MAXIMALE PERMISE PAR LES LOIS APPLICABLES, EN AUCUN CAS MICROSOFT OU SES FOURNISSEURS NE SERONT RESPONSABLES DES DOMMAGES SPÉCIAUX, CONSÉCUTIFS, ACCESSOIRES OU INDIRECTS DE QUELQUE NATURE QUE CE SOIT (NOTAMMENT, LES DOMMAGES À L'ÉGARD DU MANQUE À GAGNER OU DE LA DIVULGATION DE RENSEIGNEMENTS CONFIDENTIELS OU AUTRES, DE LA PERTE D'EXPLOITATION, DE BLESSURES CORPORELLES, DE LA VIOLATION DE LA VIE PRIVÉE, DE L'OMISSION DE REMPLIR TOUT DEVOIR, Y COMPRIS D'AGIR DE BONNE FOI OU D'EXERCER UN SOIN RAISONNABLE, DE LA NÉGLIGENCE ET DE



TOUTE AUTRE PERTE PÉCUNIAIRE OU AUTRE PERTE DE QUELQUE NATURE QUE CE SOIT) SE RAPPORTANT DE QUELQUE MANIÈRE QUE CE SOIT À L'UTILISATION DU LOGICIEL OU À L'INCAPACITÉ DE S'EN SERVIR, À LA PRESTATION OU À L'OMISSION D'UNE TELLE PRESTATION DE SERVICES DE SOUTIEN TECHNIQUE OU AUTREMENT AUX TERMES DE TOUTE DISPOSITION DU PRÉSENT EULA OU RELATIVEMENT À UNE TELLE DISPOSITION, MÊME EN CAS DE FAUTE, DE DÉLIT CIVIL (Y COMPRIS LA NÉGLIGENCE), DE RESPONSABILITÉ STRICTE, DE VIOLATION DE CONTRAT OU DE VIOLATION DE GARANTIE DE MICROSOFT OU DE TOUT FOURNISSEUR ET MÊME SI MICROSOFT OU TOUT FOURNISSEUR A ÉTÉ AVISÉ DE LA POSSIBILITÉ DE TELS DOMMAGES. LIMITATION DE RESPONSABILITÉ ET RECOURS. Malgré les dommages que vous puissiez subir pour quelque motif que ce soit (notamment, tous les dommages susmentionnés et tous les dommages directs ou généraux), l'obligation intégrale de Microsoft et de l'un ou l'autre de ses fournisseurs aux termes de toute disposition du présent EULA et votre recours exclusif à l'égard de tout ce qui précède (sauf en ce qui concerne tout recours de réparation ou de remplacement choisi par Microsoft à l'égard de tout manquement à la garantie limitée) se limite au plus élevé entre les montants suivants : le montant que vous avez réellement payé pour le Logiciel ou 5,00 \$US. Les limites, exclusions et dénis qui précèdent (y compris les clauses cidessus), s'appliquent dans la mesure maximale permise par les lois applicables, même si tout recours n'atteint pas son but essentiel. La présente Convention est régie par les lois de la province d'Ontario, Canada. Chacune des parties à la présente reconnaît irrévocablement la compétence des tribunaux de la province d'Ontario et consent à instituer tout litige qui pourrait découler de la présente auprès des tribunaux situés dans le district judiciaire de York, province d'Ontario. Au cas où vous auriez des questions concernant cette licence ou que vous désiriez vous mettre en rapport avec Microsoft pour quelque raison que ce soit, veuillez contacter la succursale Microsoft desservant votre pays, dont l'adresse est fournie dans ce produit, ou écrivez à : Microsoft Sales Information Center, One Microsoft Way, Redmond, Washington 98052-6399.

The following MICROSOFT GUARANTEE applies to you if you acquired this Software in any other country:

Statutory rights not affected - The following guarantee is not restricted to any territory and does not affect any statutory rights that you may have from your reseller or from Microsoft if you acquired the Software directly from Microsoft. If you acquired the Software or any support services in Australia, New Zealand or Malaysia, please see the "Consumer rights" section below.

The guarantee - The Software is designed and offered as a general-purpose software, not for any user's particular purpose. You accept that no Software is error free and you are strongly advised to back-up your files regularly. Provided that you have a valid license, Microsoft guarantees that a) for a period of 90 days from the date of receipt of your license to use the Software or the shortest period permitted by applicable law it will perform substantially in accordance with the written materials that accompany the Software; and b) any support services provided by Microsoft shall be substantially as described in applicable written materials provided to you by Microsoft and Microsoft support engineers will use reasonable efforts, care and skill to solve any problem issues. In the event that the Software fails to comply with this guarantee, Microsoft will either (a) repair or replace the Software or (b) return the price you paid. This guarantee is void if failure of the Software results from accident, abuse or misapplication. Any replacement Software will be guaranteed for the remainder of the original guarantee period or 30 days, whichever period is longer. You agree that the above guarantee is your sole guarantee in relation to the Software and any support services.

Exclusion of All Other Terms - To the maximum extent permitted by applicable law and subject to the guarantee above, Microsoft disclaims all warranties, conditions and other terms, either express or implied (whether by statute, common law, collaterally or otherwise) including but not limited to implied warranties of satisfactory quality and fitness for particular purpose with respect to the Software and the written materials that accompany the Software. Any implied warranties that cannot be excluded are limited to 90 days or to the shortest period permitted by applicable law, whichever is greater.

Limitation of Liability - To the maximum extent permitted by applicable law and except as provided in the Microsoft Guarantee, Microsoft and its suppliers shall not be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of business information or other pecuniary loss) arising out of the use or inability to use the Software, even if Microsoft has been advised of the possibility of such damages. In any case Microsoft's entire liability under any provision of this Agreement shall be limited to the

amount actually paid by you for the Software. These limitations do not apply to any liabilities that cannot be excluded or limited by applicable laws.

Consumer rights - Consumers in Australia, New Zealand or Malaysia may have the benefit of certain rights and remedies by reason of the Trade Practices Act and similar state and territory laws in Australia, the Consumer Guarantees Act in New Zealand and the Consumer Protection Act in Malaysia in respect of which liability cannot lawfully be modified or excluded. If you acquired the Software in New Zealand for the purposes of a business, you confirm that the Consumer Guarantees Act does not apply. If you acquired the Software in Australia and if Microsoft breaches a condition or warranty implied under any law which cannot lawfully be modified or excluded by this agreement then, to the extent permitted by law, Microsoft's liability is limited, at Microsoft's option, to: (i) in the case of the Software: a) repairing or replacing the Software; or b) the cost of such repair or replacement; and (ii) in the case of support services: a) re-supply of the services; or b) the cost of having the services supplied again.

Should you have any questions concerning this EULA, or if you desire to contact Microsoft for any reason, please use the address information enclosed in this Software to contact the Microsoft subsidiary serving your country or visit Microsoft on the World Wide Web at <http://www.microsoft.com>.

## PNA User Accounts and Passwords

---

**Important:** When the PNA power is switched on, it AUTOMATICALLY logs into Windows using the default user name and password. You do NOT need to log on. This gives anyone full access to the analyzer. The following steps can be taken to increase security of your PNA.

- Require users to logon when the PNA computer is turned ON - [Learn how to enable this feature](#)
- Setup individual accounts on the PNA with varying level of access - Learn how to [Add or Change User Accounts and Passwords](#)

[Please read about Anti-virus protection for your PNA](#)

### Existing User Accounts

The following user accounts already exist on new PNAs.

- **Default User Account** The Default User Account is created by Windows and cannot be deleted. We recommend you change the password and, if desired, the user name. DO NOT FORGET YOUR NEW PASSWORD. If you choose to [require users to logon when the PNA is turned ON](#), you will not be able to start your PNA without it.
  - Beginning in April 2004, PNAs were shipped from the factory with the default:
    - User name (not case sensitive): **PNA-Admin**
    - Password (case sensitive): **agilent**
  - Before April 2004, PNAs were shipped from the factory with the default:
    - User name: **Administrator**
    - Password: either **tsunami** or left blank.
- **Agilent Account** This Administrator account is created by Agilent for service purposes. Each PNA has a unique password for this account. Although allowed by Windows, please do not delete this account.
- **Guest Account** This account allows anyone to type in any name, without password, and gain limited access to the PNA files. This account is created by Windows and cannot be deleted. It can be renamed. This account is turned OFF when the PNA is shipped.

### Notes

- When connecting the PNA to the Internet, do NOT setup an Administrator account without a password. Although allowed by Windows, Internet viruses look for, and exploit, this condition.
- You can create as many user accounts as you like.

- The user name is not case sensitive. The password IS case sensitive.
- The PNA local policies are set so that, if logon is required, you must retype the user name (and password) every time. Do not change the local policies on the PNA.

## How to Require Users to Logon when the PNA Computer is turned ON.

[How do I know which Operating System I have?](#)

Windows 2000	Windows XP
On the Windows taskbar, click <b>Start</b> , then <b>Settings</b> , then <b>Control Panel</b>	On the Windows taskbar, click <b>Start</b> , then <b>Run</b>
Double click <b>Users and Passwords</b>	Type <b>control userpasswords2</b> then click <b>OK</b>
Check <b>Users must enter a user name and password to use this computer.</b>	Check <b>Users must enter a user name and password to use this computer.</b>

To turn this function OFF, perform the same procedure, but clear the checkbox. The account that is selected when the checkbox is cleared is the account that is automatically logged on when the PNA is turned ON.

## Add or Change User Accounts and Passwords

If the analyzer is in a secure environment, you can setup PNA users by name and grant various levels of access. This is particularly important when the PNA is remotely controlled or accessed over LAN.



You can designate a person as the administrator and then configure the PNA to allow others to use it with reduced permissions. That is, other people can be signed on to use the analyzer but they will not have the ability to perform all of the administrative functions that you can as the administrator.

**Note:** To connect and install a new ECal module, you must be logged on with an Administrator account. A user account with limited access will NOT allow an ECal module driver to be loaded. [Learn more.](#)

## How to add or change a user account and password

[How do I know which Operating System I have?](#)

Windows 2000	Windows XP
In the analyzer <b>System</b> menu, point to <b>Configure</b> , and click <b>Control Panel</b> .	Click Start, then point to Settings, then click Control Panel
In the <b>Control Panel</b> window, scroll down and select the <b>Users and Passwords</b> application.	Click <b>User Accounts</b>
On the <b>Users</b> tab, if the <b>Add</b> button appears dimmed, select the <b>Users must enter a user name and password to use this computer</b> check box near the top of the window.	Follow the prompts to: <ul style="list-style-type: none"> <li>• Change an account</li> <li>• Create a new account (see below)</li> </ul>

	<ul style="list-style-type: none"> <li>Change the way users log on or off (with password) <b>CAUTION:</b> Although allowed by Windows, do NOT allow an Administrator account without a password. Internet viruses look for, and exploit, this condition.</li> </ul>
Click <b>Add</b> to enter the information for yourself or for another user.	
In the <b>User name</b> box, enter a user name for the user. In the <b>Full name</b> box, enter the full name of the user.	
In the <b>Description</b> box, enter a description for the user. Then, click <b>Next</b> .	
In the <b>Password</b> box, have the user type a password. Have the user retype the password in the <b>Confirm password</b> box. Then, click <b>Next</b> .	
<p>Select the level of access that you wish to grant this user.</p> <p><b>Note:</b> Standard users and restricted users are <b>NOT</b> able to switch GPIB modes and install firmware.</p> <p>There are several other levels of security that you may grant in the <b>Other</b> list. A description of each of these other levels is displayed beneath the <b>Other</b> box when it is selected. Then, click <b>Finish</b>.</p>	
In the <b>Users for this computer</b> box, validate the user name and security level group of the user.	
If you want this user to be able to use the network analyzer without entering their password each use, clear the <b>Users must enter a user name and password to use this computer</b> check box. Click <b>OK</b> .	

**Note:**

A user account with limited access will NOT allow an ECal module driver to be loaded. [Learn more.](#)

However, limited access users **ARE** ABLE to switch GPIB modes and install firmware.

When the <b>Automatically Log On</b> window is displayed, have the new user type their password in the <b>Password</b> box and have them retype the password in the <b>Confirm Password</b> box.	
Click <b>OK</b> to complete this user addition.	
In the <b>File</b> menu, click <b>Close</b> to close the Control Panel.	

---

Last Modified:

8-Apr-2008    Added Logon notes

## PNA Computer Properties

---

The PNA uses a personal computer and a Windows operating system. The following are common tasks that you may need to perform on the PNA computer.

- [View or change Full Computer Name](#)
- [Check IP Address](#)
- [Check the amount of RAM](#)
- [Check CPU Speed](#)
- [Set Time and Date](#)
- [Internal and External Speakers](#)

### Other Administrative Task Topics

#### View or change Full Computer Name

Your PNA has a unique computer name that identifies it on a network. To view or change the computer name, you must first [minimize the PNA](#) application.

[How do I know which Operating System I have?](#)

Windows 2000	Windows XP
On the desktop, right-click <b>My Computer</b>	On the desktop, right-click <b>My computer</b> Icon
Click <b>Properties</b>	Click <b>Properties</b>
Click the <b>Network Identification</b> tab at the top of the dialog box	Click the <b>Computer Name</b> tab at the top of the dialog box
Click <b>Properties</b>	Click <b>Change</b> next to "..rename this computer.." message
Type your new <b>Computer Name</b>	Type your new <b>Computer Name</b>

**Note:** To add your computer to a domain, or to set up the networking configuration, contact your company's I.T. department. This setup is custom for each company.

To restore the PNA application, click **PNA Analyzer** in the task bar at the bottom of the screen.

#### Check IP Address

If your PNA is connected to a LAN, you can view the IP address and other networking information.

1. [Minimize the PNA](#) application
2. Click **Start**, then **Run**
3. Type **cmd**, then click **OK**
4. At a DOS prompt, type **ipconfig /all**

## Check the amount of RAM

Random Access Memory (RAM) is the amount of working memory in your computer. The PNA application can require up to 512 MB of RAM depending on the settings you use. If your PNA is operating slowly when you have more than four windows open or if you routinely use more than 1601 data points, you may need to upgrade to 512 MB.

The amount of RAM in your PNA may limit your ability to upgrade firmware. See [http://na.tm.agilent.com/pna/firmware/PNA\\_support\\_matrix.doc](http://na.tm.agilent.com/pna/firmware/PNA_support_matrix.doc)

To view the amount of PNA RAM, you must first [minimize the PNA](#) application.

### [How do I know which Operating System I have?](#)

Windows 2000	Windows XP
On the desktop, right-click <b>My Computer</b>	On the desktop, right-click <b>My computer</b> Icon
Click <b>Properties</b>	Click <b>Properties</b>
Click the <b>General</b> tab at the top of the dialog box	Click the <b>General</b> tab at the top of the dialog box
The amount of RAM appears at the bottom of the window.	The amount of RAM appears at the bottom of the window.

To restore the PNA application, click **PNA Analyzer** in the task bar at the bottom of the screen.

## Check CPU Speed

The speed of the PNA processor (CPU) is a factor in determining how quickly the PNA processes data. Also, the CPU speed in your PNA may limit your ability to upgrade firmware. See [http://na.tm.agilent.com/pna/firmware/PNA\\_support\\_matrix.doc](http://na.tm.agilent.com/pna/firmware/PNA_support_matrix.doc).

You can see which CPU is in your PNA by comparing your PNA rear-panel with the images at <http://na.tm.agilent.com/pna/cputype.html>.

Or, beginning with PNA Rev. A.09.10, on the PNA click **Help**, then **About Network Analyzer**. [Learn more.](#)

Or, you can do the following to check your PNA CPU speed:

1. [Minimize the PNA](#) application.
2. Then do the following:



### [How do I know which Operating System I have?](#)

Windows 2000	Windows XP
On the desktop, right-click <b>My Computer</b>	On the desktop, right-click <b>My computer</b> Icon
Click <b>Manage</b>	Click <b>Properties</b>
Open <b>System Tools</b> folder, then click <b>System Information</b> .	Click the <b>General</b> tab at the top of the dialog box
Click <b>System Summary</b> .	The CPU speed appears near the bottom of the window
After refreshing, the CPU speed appears at the end of the <b>Processor</b> entry.	

To restore the PNA application, click **PNA Analyzer** in the task bar at the bottom of the screen.

See [PNA configurations](#) to learn if you can upgrade your PNA CPU.

## Set Time and Date

Both Windows 2000 and XP

To set the time and date on your PNA, you must first [minimize the PNA](#) application.

1. Move the cursor to the lower corner of the screen
2. When the taskbar appears, double-click on the displayed time. This opens the **Date/Time Properties** dialog box.
3. Change the date, time, and time zone as appropriate.

To restore the PNA application, click **PNA Analyzer** in the task bar at the bottom of the screen.

## Internal and External Speakers

There is an internal sound card and speaker in the PNA. However, the output volume is limited and there is no audio output jack. For higher volume, use a powered USB speaker.

### Turn OFF | ON Speaker

There are occasion when you might want to turn the speaker OFF, such as if the PNA is generating errors. [Learn more about errors.](#)

1. To turn ON or OFF the PNA speaker, you must first [minimize the PNA](#) application.
2. Then click **Start, Control Panel**, then **Sounds and Audio Devices**.
3. Under **Device Volume**, check **Mute**.

---

Last Modified:

30-Nov-2009	Added Help, About note.
26-Nov-2008	Added external speaker note
12-Mar-2008	Added link to support site
20-Sep-2007	Added speaker OFF

## Run Error Check and Disk Defragmenter

---

When the PNA is shutdown unexpectedly or power is removed without first shutting down, large amounts of Hard Disk Drive space is rendered unusable. If shutdown in this manner enough times, the PNA could become unstable and no longer work.

This Hard Disk Drive space can be recovered by first running Windows **Error-checking** to find and correct errors on the disk, and then the **Disk Defragmenter** to recover Hard Disk Drive space.. These programs should be run routinely, about every 1 to 4 weeks, depending on how often the PNA is unexpectedly shutdown.

To learn more about Disk Defragmenter, see the Windows Help file.

Follow this procedure to run these programs:

Windows 2000	Windows XP
On the desktop, double-click <b>My Computer</b>	On the desktop, double-click <b>My Computer</b>
Select <b>Local Disk (C:)</b>	Select <b>System OS</b>
Click <b>File</b> , then <b>Properties</b>	Click <b>File</b> , then <b>Properties</b>
Click the <b>Tools</b> tab	Click the <b>Tools</b> tab

### Error-checking

- Click **Check Now**.
- Check **Automatically fix file system errors**.
- Click **Start**.
- Click **Yes** to run disk check on next restart.
- Manually [restart the PNA](#). The disk check will run before Windows restarts.

Approximately every six months, check the second box in addition to the first box. The error-checking process takes much longer, but performs a more complete check.

### Defragmentation

- Click **Defragment Now...**
- Click **Defragment** to begin the defragment process.
- Click **Close** when defragmentation is complete.

## Recovering from PNA Hard Drive Problems

---

The leading cause of PNA failures is problems with the PNA Hard Disk Drive (HDD). These problems are usually preventable (see [Preventing PNA HDD Problems](#)), and in many cases, recoverable. The following could save you weeks of downtime and the cost of replacing your PNA HDD.

This document is now on the Agilent PNA Support Website: <http://na.tm.agilent.com/pna/>. When at this webpage, click the **Hard Drive Recovery** link.

If your PNA does experience a Hard Disk Drive Problem, you will not be able to access this Help file, but you may be able to access the Internet from another computer.

## Wake PNA at Specified Time

---

Depending on the CPU that is used in your PNA-X, you may be able to start your PNA-X at a specified time every day. This feature allows the PNA-X to be shut down at night (saving energy), then wake the PNA-X at a specified time to allow sufficient warm-up time in the morning before being used.

To use this feature, the PNA-X must have a 2000 MHz CPU board. [See how to determine the CPU board.](#)

### How to configure the PNA-X to wake at a specified time:

1. Connect a USB keyboard to the PNA-X.
2. [Restart](#) the PNA-X.
3. During power-up, press F2 on the keyboard at the moment it indicates to do so on the bottom of the PNA-X screen.
4. In the BIOS, go to **Configuration** then **Power Control Configuration**. Set **Resume On Time:** to **ON**, and set the **Resume Time:** to the desired wake-up time.
5. Press **F10** to Save, Exit, and restart the PNA-X.
6. At the end of the day, Shut Down (not hibernate or standby) the PNA-X.
7. The PNA-X will wake-up at the specified time.

### Note:

A utility can also be set to provide a Wake-on-LAN (WOL) call. This is outside the scope of PNA Help. Search the internet for **WOL utility**.

To configure the PNA-X for WOL, follow the above procedure for accessing the BIOS.

In the BIOS:

1. Go to **Configuration** then **LAN Configuration**. Ensure that **Power Saving when S5** to **Disabled** (default setting).
2. Go to **Configuration** then **Power Control Configuration**. Set **PME Wake From S5** to **Enabled**.
3. Press **F10** to Save, Exit.

---

Last Modified:

18-May-2010    MX New topic

## Microsoft Windows® XP / 2000 Considerations

---

In this topic:

- [Microsoft Windows on the PNA](#)
  - [Using USB](#)
  - [Plug & Play Stability and Security](#)
  - [LAN Connections](#)
  - [Single and Double Click option](#)
  - [Windows XP Theme](#)
  - [Printing](#)
- 

### Microsoft Windows on the PNA

- Beginning in April 2004, the PNA is shipped from the factory with a modified version of Microsoft Windows XP operating system. Previously, the PNA was shipped with Windows 2000. The PNA application performs identically using these two operating systems.
- Beginning in Dec. 2005 with PNA Rev 6.0, firmware cannot be upgraded on PNA models that use Microsoft Windows 2000. For more information, see the [PNA support website](#).

### To determine which Operating System is installed on your PNA:

1. [Minimize the PNA application](#)
2. On the PNA desktop, click **Start**.
3. Along the side of the Start menu appears one of the following:
  - **Windows 2000 Professional**
  - **Windows XP Professional**

### VERY IMPORTANT Protect your hard drive!

The leading cause of PNA failures is problems with the PNA Hard Disk Drive (HDD). These problems are usually preventable, and in many cases, recoverable. [Learn more about protecting your PNA](#).

### Using USB

The PNA has at least two USB ports for connecting devices: one on the front panel and at least one on the [rear](#)

[panel](#). The main advantages of USB are “hot” connects and disconnects and fast data transfer speeds. Electronic Calibration modules are now available with USB connections.

The first time you plug a device into a USB port there is some wait time. Windows reports it is identifying the hardware, then searching for the correct driver, then installing the driver (if it was found).

Connecting that same device back into that same port later is quick and easy, but if you move the device to a different USB port, you will have to wait through the hardware ID and driver search again.

[Learn about USB limitations.](#)

**Note:** Certain USB devices (such as ECAL modules) require you be logged on with [Administrator privileges](#) the first time you plug them into the PNA. This must be done for each serial number. Click **Next** to choose the default settings when installing new USB devices.

## Plug & Play Stability and Security

Plug & Play capabilities is similar to Win 95 and 98. It provides both a stable and secure operating environment. You may notice also that it greatly reduces the number of required reboots.

## LAN Connections

Windows supports DHCP and fixed IP addressing. Also, “Hot” connect and disconnect of the LAN cable, as well as a visual indicator of LAN status in system tray area, makes LAN connections more intuitive. In addition, the Hardware Wizard helps users with system hardware configuration.

## Single and Double Click option

By default, Windows allows a single-click method of launching icons. To revert to double-clicking, click **Start**, then **Settings**, then **Control Panel**, then click **Mouse**. In the Mouse Properties dialog, select **Double-click to open an item**. Then click **OK**.

## Windows XP Theme

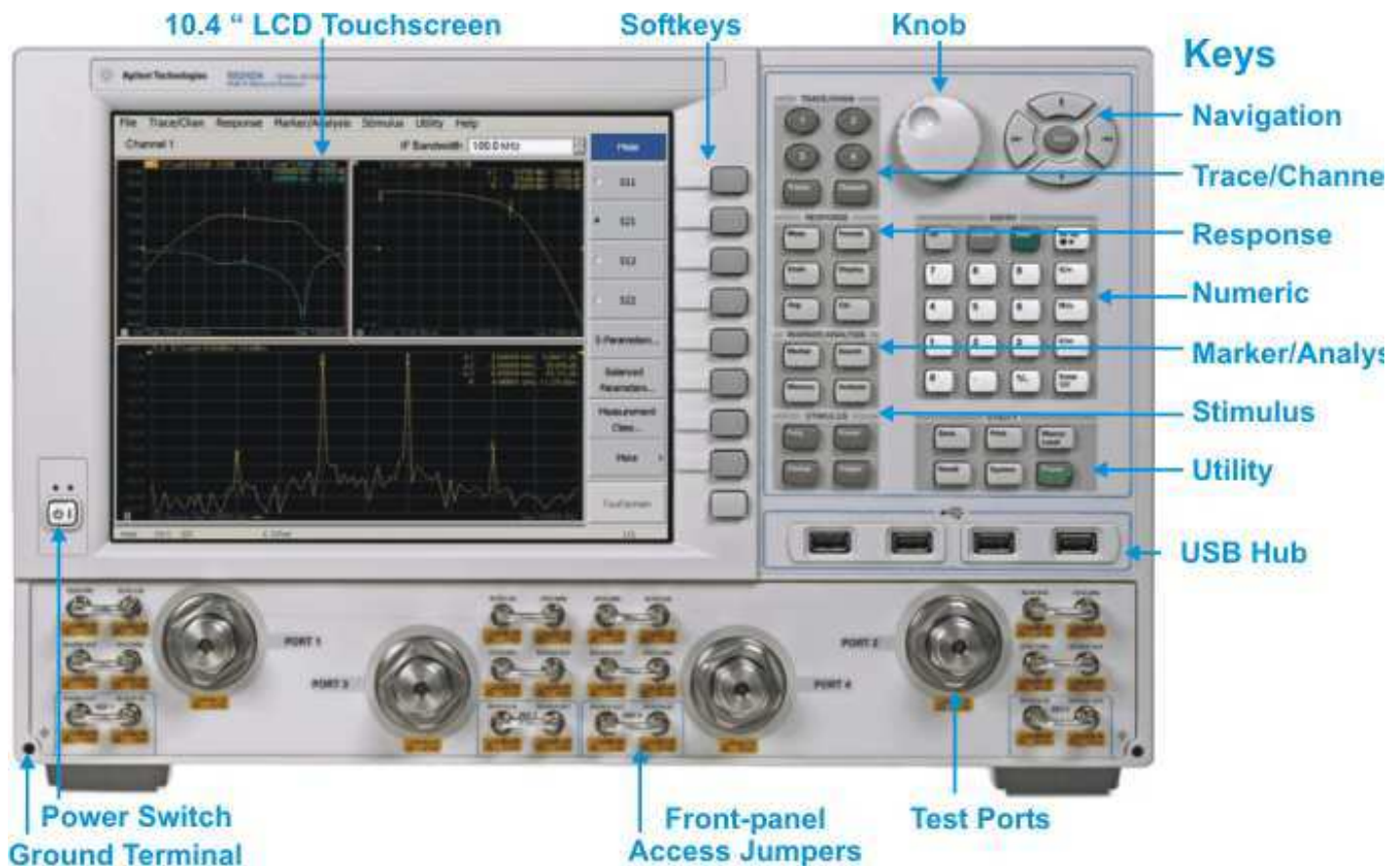
The PNA application is designed for, and best viewed in, **Windows Classic** theme. To change the theme from Windows XP to Windows Classic,

1. [Minimize the PNA application](#).
2. Right-click on the Desktop, then click **Properties**.
3. On the Theme tab, under **Theme** select **Windows Classic**.

## Printing

Adding a printer should be done outside of the PNA application. [Learn more.](#)

## PNA-X and 'C' Model Front-Panel Tour



### See Also

- 'C' Models
- PNA-X Display area
- PNA-X Models / Options
- PNA-X Rear-Panel Tour

### Familiar Hardkey layout, similar to Agilent 8720 and 8753 Network Analyzers

Back to the familiar layout, significantly different from legacy PNA models. Most measurement settings are made from the Stimulus Block and the Response Block.

### Fully functional Hardkey/Softkey selections consistent with Menu (mouse) selections

Access ALL PNA settings from the front panel using hardkey/softkeys or from the Menu using a mouse. Both methods are consistent; learn the menu structure once, and it applies to both methods of UI navigation.



Power Switch

Used for choosing between power-on ( | ) and standby ( O ) state.  
Learn to power ON and OFF the PNA.

Test Ports

The PNA-X is available with 2 or 4 test ports.  
See PNA-X specs for more informations about the Test port connectors and Input damage levels.

Front panel Access Jumpers

These connectors provide direct access to the PNA source and receivers. This allows you to make a wide variety of measurements and improve dynamic range.  
See the PNA-X front panel jumpers specifications

Port 1	Port 3	Port 4	Port 2
			

USB Hub

This USB hub contains four USB ports to power your PNA peripherals. There are also four USB ports on the rear panel.  
**Limitation:** The total power consumption for all eight USB ports is limited to 4.0 amps. If this limit is exceeded, all USB ports are disabled until a device is removed and power consumption falls below the limit. When first connected, Agilent ECal modules 8509x and N4431 draw significantly more current than other modules.

Ground terminal

Connect a banana-type plug to this terminal for grounding to the PNA chassis.

No probe power

Probe power is NOT provided with the PNA-X.

## Hardkeys

### TRACE/CHANKeys

Manages the Traces and Channels on the PNA display.

Hard Key	Invokes these Softkeys
1, 2, 3, 4	Makes the corresponding trace active.
Traces	Invokes the Traces softkey menu which allows you to <b>create</b> a new trace, <b>select</b> a trace, <b>delete</b> a trace, or <b>maximize</b> the trace.
Channels	Invokes the Channels softkey menu which allows you to manage channels.

### RESPONSE Keys

Performs operations on measurement traces after data is measured - not including Data Analysis operations.

Hard Key	Invokes these Softkeys - Click to learn more
Meas	Measurement selections <ul style="list-style-type: none"><li>S-Parameters</li><li>Balanced Parameters</li><li>Measurement Class</li></ul> More Meas <ul style="list-style-type: none"><li>Receivers</li></ul>
Format	Format
Scale	Scale Electrical Delay Phase Offset More <ul style="list-style-type: none"><li>Velocity Factor</li><li>Media -Waveguide/coax</li><li>Waveguide cutoff freq</li></ul>
Display	Display settings Arrangements (Overlay...)

	<p>Windows (Managing)</p> <p>Measurement Setups</p> <p>Display Items</p> <ul style="list-style-type: none"><li>• Title</li><li>• Trace Status</li><li>• Freq Stimulus</li><li>• Marker Readout</li><li>• Toolbars</li><li>• Tables</li><li>• Status Bar</li><li>• Hide Sofkeys</li><li>• Minimize App</li></ul>
<b>Avg</b>	<p>Averaging</p> <p>Smoothing</p> <p>IF Bandwidth</p>
<b>Cal</b>	<p>Start Cal</p> <ul style="list-style-type: none"><li>• Cal Wizard</li><li>• Preferences</li><li>• Global Delta Match</li></ul> <p>Correction</p> <p>Power Cal</p> <ul style="list-style-type: none"><li>• Source Cal</li><li>• Receiver Cal</li></ul> <p>Manage Cals</p> <ul style="list-style-type: none"><li>• Cal Set</li><li>• Cal Type</li><li>• Cal Set Viewer</li></ul>

	Properties (must have a Cal ON) no idea what this is Port Ext Toolbar Interpolation Fixtures <ul style="list-style-type: none"> <li>• ON  Off</li> <li>• Port matching</li> <li>• lots more</li> </ul> Manage Cal Kit Manage ECal System Z0 Velocity Factor
--	---

## MARKER/ANALYSIS Keys

Control all aspects of Data Analysis including Markers and Math functions..

Hard Key	Invokes these Softkeys - Click to learn more
<b>Marker</b>	Markers Properties <ul style="list-style-type: none"> <li>• Delta Markers</li> <li>• Discrete</li> <li>• Type</li> <li>• Coupled</li> </ul> Functions
<b>Search</b>	Marker Search
<b>Memory</b>	Data/ Memory Math 8510 Mode
<b>Analysis</b>	Limit Lines <ul style="list-style-type: none"> <li>• Limit Test</li> <li>• Global Pass/Fail</li> </ul> Trace Statistics

	Gating
	Transform
	<ul style="list-style-type: none"> <li>• Windowing</li> <li>• Coupling</li> <li>• Distance Marker</li> </ul>
	Equation Editor

### STIMULUS Keys

Controls settings that determine **what** data (stimulus range), and **how** data (sweep type and triggering), is measured.

Hard Key	Invokes these Softkeys - Click to learn more
<b>Freq</b>	Frequency Range Frequency Offset Mode
<b>Power</b>	RF Power level Power Slope Power and Attenuator settings
<b>Sweep</b>	Sweep Time Number of Points Sweep Type Sweep Setup Segment Table settings
<b>Trigger</b>	Trigger settings

### UTILITY Keys

Performs global PNA operations.

Hard Key	Invokes these Softkeys - Click to learn more
<b>Save</b>	File Save Save (State) As Save Data As Auto Save Manage Files Delete Files User Preset
<b>Print</b>	Print Print to file Page Setup
<b>Macro/Local</b>	Macro Setup Run Macros
<b>Recall</b>	File Recall
<b>System</b>	Security Configure <ul style="list-style-type: none"> <li>• SICL / GPIB</li> <li>• Control Panel (Windows)</li> <li>• System Z0</li> <li>• Power Meter Settings</li> <li>• Millimeter Module</li> </ul> Service Help <ul style="list-style-type: none"> <li>• Error Messages</li> <li>• About NA</li> </ul> User Key Touchscreen
<b>Preset</b>	Preset User Preset

## ENTRY Keys

Hard Key	Invokes these Softkeys
<b>OK</b>	Closes a dialog box and enters any values made in the dialog box.
<b>Cancel</b>	Closes a dialog box.
<b>Help</b>	Launches this Help file.
<b>Bk Sp</b>	Back Space. Backs up the cursor and deletes any previous selection.
<b>1 to 9</b>	Selects values for measurement settings, then press Enter or G/n - M/u - k/m to complete the selection.
<b>G/n</b> <b>M/u</b> <b>k/m</b>	Completes the value selection, assigning a unit of measurement. <ul style="list-style-type: none"><li>• G/n (Giga/Nano) E12 or E-12</li><li>• M/u (Mega/micro) E6 or E-6</li><li>• k/m (kilo/milli) E3 or E-3</li></ul>
<b>Enter</b> <b>Off</b>	Enters the values that you select for the measurement settings.
<b>Decimal point</b>	Enters a decimal point to designate fractions of a whole number.
<b>+/-</b>	Plus - Minus Toggles between a positive and negative value entry if it is the first key pressed in the entry.

## Knob

Rotate to increase or decrease the value of the active entry.

## Navigation Keys

These keys allow you to navigate through menus and dialog boxes and select choices from the active entry toolbar.

Hard Key	Invokes these Softkeys
<b>Left / Right</b>	Moves left and right through menus. Moves tab-left and tab-right within dialog boxes.
<b>Up / Down</b>	Moves up and down through menus. Behaves as follows in a dialog box: <ul style="list-style-type: none"> <li>• Modifies a numeric value</li> <li>• Moves through items in a drop-down list</li> <li>• Moves through options buttons in a group of option buttons</li> </ul>
<b>Click</b>	Makes a selection just like a mouse click.

Last Modified:

15-Apr-2008    Added ALL front panel jumper images

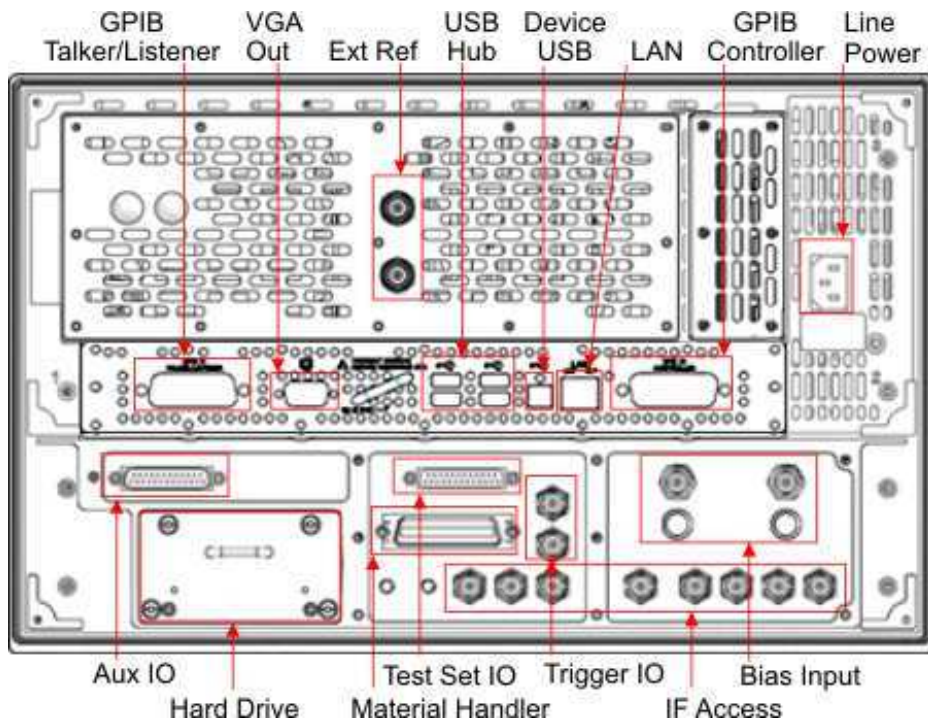
23-Aug-2007    Added front panel jumpers image



## Rear Panel Tour

This image includes ALL rear-panel features for all [PNA models](#), excluding the PNA-X. [See the PNA-X Rear Panel](#).  
Your PNA may not have this capability or look.

Click on a connector for detailed information.



### 10 MHz Reference IN/OUT

**10 MHz Reference Input** When a 10 MHz external reference signal is detected at this port, it will be used as the instrument frequency reference instead of the internal frequency reference.

**10 MHz Reference Output** This BNC(f) connector outputs a frequency reference signal for use by other test equipment.

[See PNA specifications](#)

**VGA Connector** [Learn more](#)

### USB Hub

This USB hub contains four Hi-Speed USB 2.0 ports to power your PNA peripherals.

Each USB 2.0 port is capable of providing .5 amps. If this limit is exceeded, all USB ports are disabled until a device is removed and power consumption falls below the limit.

**Note:** When first connected, Agilent ECal modules 8509x and N4431 draw significantly more current than other modules. [See PNA specifications](#)

**USB Device** [Learn more](#)

### **LAN Connector**

This 10/100BaseT Ethernet connection has a standard 8-pin configuration and auto selects between the two data rates.

### **Line Power**

[See PNA specifications](#)

### **GPIO Controller and Talker/Listener Ports**

The PNA-X can be a GPIO Controller and Talker/Listener. [Learn more.](#)

### **Test Set I/O**

[See Details](#)

### **Bias IN and Fuses**

Apply Bias to the PNA ports through these BNC connectors.

[See PNA specifications](#)

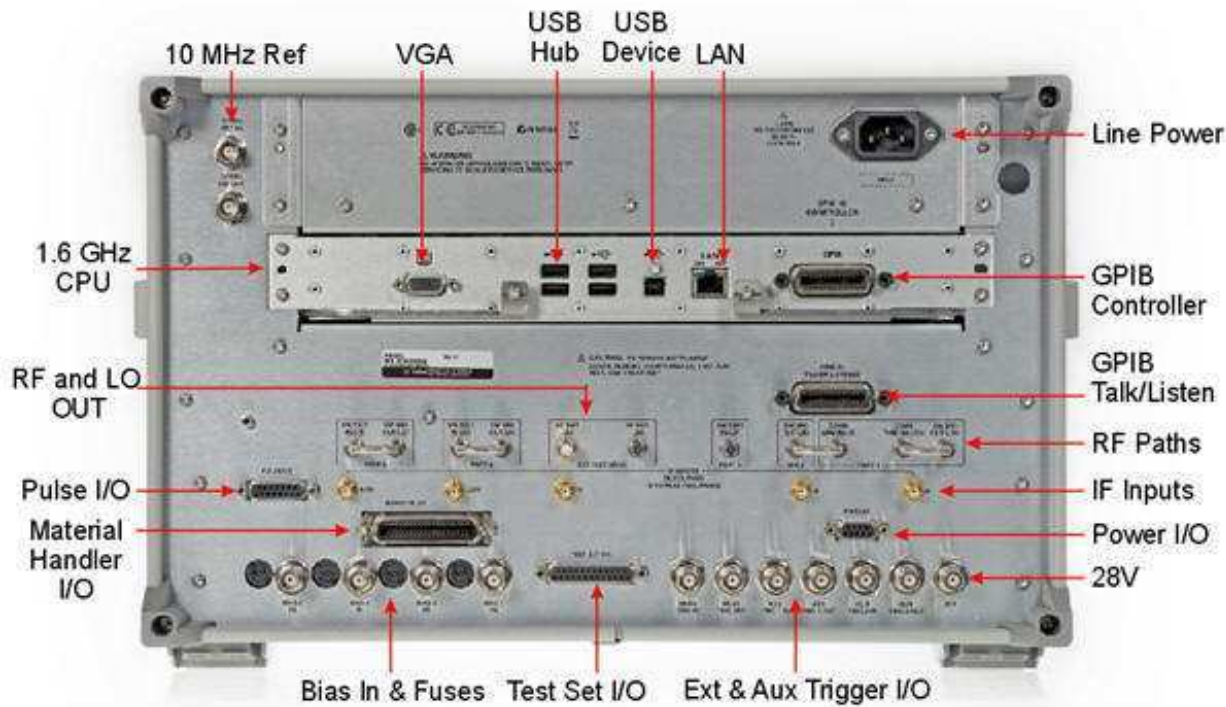
### **Material Handler I/O**

[See details.](#)

## PNA-X Rear Panel

---

Click image to learn more.



Beginning April 2009, PNA-X and N5264A models shipped with this newer CPU board. The major changes include:

- 2 GHz [CPU](#)
- **A** Removable Hard disk drive (Learn more in the [Service Guide](#))
- **B** Relocated [GPIB Controller port](#)
- **C** Relocated [VGA connector](#)

---

### 10 MHz Reference IN/OUT

**10 MHz Reference Input** When a 10 MHz external reference signal is detected at this port, it will be used as the instrument frequency reference instead of the internal frequency reference.

**10 MHz Reference Output** This BNC(f) connector outputs a frequency reference signal for use by other test

equipment.

[See PNA-X specifications](#)

## VGA Connector [Learn more](#)

## USB Hub

This USB hub contains four USB ports to power your PNA peripherals. There are also four USB ports on the [front panel](#).

**Limitation:** The total power consumption for all eight USB ports is limited to 4.0 amps. If this limit is exceeded, all USB ports are disabled until a device is removed and power consumption falls below the limit. When first connected, Agilent ECal modules 8509x and N4431 draw significantly more current than other modules. [See PNA-X specifications](#)

## USB Device [Learn more](#)

## LAN Connector

This 10/100BaseT Ethernet connection has a standard 8-pin configuration and auto selects between the two data rates.

## Line Power

[See PNA-X specifications](#)

## GPIO Controller and Talker/Listener Ports

The PNA-X can be a GPIO Controller and Talker/Listener. [Learn more.](#)

## RF Path Access

These connectors allow [RF Path Configuration](#).

Ports 3 and 4 are not available on 2-port models.

**Note:** These connectors are NOT available on the [N5264A](#)



## RF and LO OUT



**Caution:** LO OUT has more power than previous PNA models.

**Note:** The RF OUT connector is NOT available on the [N5264A](#)

[See PNA-X specifications](#)

## IF Path Inputs

Option 020 adds these connectors, which allow access to the PNA Receiver / IF paths.



These are labeled A, B, C/R1, D/R2, R.

- For 2-port PNA-X models, use A, B, R1, R2.
- For 4-port PNA-X models, use A, B, C, D, R.

[See IF Path Configuration settings and block diagram.](#)

## Power I/O

Has some of the [AUX I/O connector](#) functionality on the PNA-L and E836xB models.

[See Details](#)

## 28 V (BNC output)



Used to power a noise source for the [Noise Figure App.](#)

## External and AUX Trigger I/O



**MEAS TRIG IN** - When enabled, PNA is triggered by signals on this connector. [Learn more.](#)

**MEAS TRIG RDY** When enabled, PNA outputs a 'READY' signal on this connector to other devices. [Learn more.](#)

**AUX TRIG 1&2 IN** When enabled, PNA accepts signals on these connectors which indicates that the external devices is ready to be triggered. [Learn more.](#)

**AUX TRIG 1&2 OUT** When enabled, PNA outputs signals on these connectors either before or after a measurement. [Learn more.](#)

## Test Set I/O

[See Details](#)

## Bias IN and Fuses



Apply Bias to the PNA ports through these BNC connectors.

The PNA-X will meet all of its RF specifications with bias up to 200 ma. As the DC bias is increased, corrected source match and directivity will degrade at low RF frequencies. Above 100 MHz, with up to 1 A bias, there is very little degradation in corrected source match and directivity.

[See PNA-X specifications](#)

## Material Handler I/O

[See details.](#)

## Pulse I/O

[See Details](#)

**1.6 GHz CPU** See [CPU Speed / Performance](#)

---

Last modified:

9-Apr-2009	Added new CPU board
14-May-2008	Added RF Path images
4-Sep-2007	Added 28V image
June 6, 2007	Added RF and IF connector images
January 11, 2007	MX New topic

## Powering the PNA ON and OFF

---

The following is described in this topic:

- [How to...](#)
- [Hibernate](#)
- [ON](#)
- [Shutdown](#)
- [Turn OFF Autostart](#)

### Notes

During boot up of Windows or of the Network Analyzer application program, do **NOT** press keys on the front panel, rotate the RPG knob, or connect a USB device. Doing so MAY lead to a front panel lockup state.

If the PNA front-panel keypad or USB ports are not responding, SHUTDOWN or RESTART the PNA; do NOT Hibernate. This causes the PNA drivers to awaken from hibernation in the same corrupt state.

### How to Log off, Shut down, Restart, or Hibernate the PNA.

1. BRIEFLY press the front-panel PNA power button.
2. In the **What do you want the computer to do?** list, choose an action:
  - Log off (closes programs)
  - [Shut down](#)
  - Restart (shutdown and start)
  - [Hibernate](#) (NOT available on N5244A, N5245A, and N5264A models)
3. Press **OK** to perform the action

**Note:** ONLY if the PNA is locked and you cannot operate the mouse or keypad - Press and hold the power button for at least four seconds. **This practice should be avoided!** Repeated shutdowns in this manner WILL damage the hard drive. [Learn more about damaging the PNA hard drive.](#)

### Hibernate Mode

Hibernate is NOT available on N5244A, N5245A, and N5264A models.



- In hibernate mode the current [instrument state](#) is automatically saved to the hard disk before the PNA is powered OFF.
- When the PNA is powered ON, this instrument state is loaded, thus saving time over a full system boot-up.
- A password is usually NOT required to resume PNA operation after Hibernate mode.
- The hibernation state is the normal OFF state. A small amount of standby power is supplied to the PNA when it is in the hibernation mode. This standby power only supplies the power switch circuits and the 10 MHz reference oscillator; no other CPU-related circuits are powered during hibernation. To guarantee that your measurements meet the PNA specified performance, allow the PNA to **warm-up for 90 minutes** after the power button light has changed from yellow back to green.

## ON Mode

- To turn ON the PNA press the power button.
- The power indicator will change to green when power is ON.

## Turn OFF PNA Autostart

The PNA application (835x.exe) always starts automatically when power is turned ON. To cause the PNA to NOT Autostart, do the following:

1. Minimize the PNA application.
2. From Windows Explorer, navigate to and double-click the following file: C:/Program Files/Agilent/Network Analyzer/Service/Toggle\_PNA\_Autostart.

The script toggles the PNA Autostart mode ON and OFF.

## Shutdown Mode

- In shut down mode the current instrument state is NOT automatically saved before the PNA is powered OFF.
- When the PNA is again powered ON, a full system boot-up is performed and the PNA powers-up in the [preset settings](#).
- A password may be required to resume PNA operation after being in Shutdown mode. [Learn more.](#)
- To guarantee that your measurements meet the PNA specified performance, allow the PNA to **warm-up for 90 minutes** after the power indicator has turned green.
- The power indicator will change to yellow when power is OFF.

**Note:** If the PNA is locked and you cannot operate the mouse or keypad, shut down the PNA by pressing and holding the power button for at least four seconds.

**This practice should be avoided!** Repeated shutdowns in this manner WILL damage the hard drive. [Learn more](#)



## Unplugging the PNA

- Remove the power cord from the PNA ONLY when the power indicator is yellow, in either Hibernate or Shutdown mode. If the power cord is removed while the power indicator is green (PNA ON), damage to the hard drive is possible.
- The indicator will remain yellow for several seconds after the power cord has been removed.
- When plugged back in and the power button is pressed to ON, the PNA starts in the mode it was in when the power cord was unplugged, either Hibernate or Shutdown.

---

Last Modified:

3-Jun-2009     Updated for N5244A and N5245A

## Traces, Channels, and Windows on the PNA

---

It is critical to understand the meaning of the following terms as they are used on the PNA.

- [Traces](#) - [Managing](#)
- [Channels](#) - [Managing](#)
- [Windows](#) - [Managing](#)

**Note:** You may experience a significant decrease in computer processing speed with combinations of the following: increased number of points, number of traces, and calibration error terms (full 2-port or 3-port). If this becomes a problem, you can increase the amount of RAM with PNA [Option 022](#). To monitor the amount of PNA memory usage, press **Ctrl Alt Delete**, select **Task Manager**, then click on the **Performance** tab.

### Other Quick Start topics

**Traces** are a series of measured [data points](#). There is no theoretical limit to the number of traces. However, the practical limit is the [maximum number of windows](#) \* the maximum number of traces per window (**24**).

In addition, one memory trace can be stored and displayed for every data trace. [Learn more about Math / Memory traces](#).

Trace settings affect the presentation and mathematical operations of the measured data.

The following are Trace settings:

- [Parameter](#)
- [Format and Scale](#)
- [Smoothing](#)
- [Correction ON / OFF](#)
- [Electrical Delay](#)
- [Phase Offset](#)
- [Trace Math](#)
- [Markers](#)
- [Time Domain](#) (Opt 010)

### Managing Traces

- [How to \*\*Select\*\* a trace](#)
- [How to \*\*Delete\*\* a trace](#)

- [How to \*\*Move\*\* a trace](#)
- [How to \*\*Maximize\*\* a trace](#)
- [How to \*\*Create\*\* a new trace](#)
- [How to \*\*Change\*\* the trace parameter.](#)
- [How to display a custom \*\*trace title\*\*.](#)
- [How to display a \*\*wide\*\* active trace](#)

## How to Select a Trace

A trace must be selected (active) before its trace settings can be changed.

[How to know which trace is Active?](#)

### Using front-panel

#### HARDKEY [softkey] buttons

1. For Traces 1-4, press the corresponding Hard Key
2. For other trace numbers, press **TRACES**
3. then **[Select Traces]**
4. Select a trace number in the [Entry toolbar](#).

### PNA Menu using a mouse

1. Click the [Trace Status](#) label or trace.

◀ **Programming Commands** ▶

## How to Delete a Trace

### Using front-panel

#### HARDKEY [softkey] buttons

1. For Traces 1-4, press the corresponding Hard Key
2. For other trace numbers, press **TRACES**
3. then **[Select Traces]**
4. Select a trace number in the [Entry toolbar](#).

### PNA Menu using a mouse

1. Right-click the [Trace Status](#) label, then click Delete.

◀ **Programming Commands** ▶

## How to Move a trace to a different window

### Using front-panel HARDKEY [softkey] buttons

1. Select the trace to move.
2. Press **TRACES**
3. then **[Move Trace]**
4. Select a window number in the following dialog.

### PNA Menu using a mouse

1. Right-click the [Trace Status](#) label, then click **Move Trace**.

PNA-X ONLY



This dialog is launched by clicking **Trace/Chan**, then **Delete Trace**

The Select Trace dialog is launched by clicking **Trace/Chan**, then **Select Trace**

### Select, Delete, Move Traces dialog box help

Both the **Select Trace** and **Delete Trace** dialogs work the same.

Select a trace, then click **OK**.

Only ONE trace can be Selected or Deleted.

#### Note:

To EASILY select a trace, click the [Trace Status](#) label.

To EASILY delete a trace, right-click the Trace Status label, then click **Delete**.

### Trace Max

Makes the active trace the ONLY trace on the display. All other traces are hidden.

### How to do Trace Max

- Select **Trace**, then **Trace Max**.

- With Trace Max ON, select a different trace from the **Traces** softkeys to make that trace visible.
- To make all traces visible again select **Trace Max OFF**

### Trace Title

A Trace Title overwrites the Measurement Parameter in the [Trace Status](#) area, the [Status Bar](#), and [hardcopy prints](#).

- This title has priority over [Equation Editor](#) titles.
- The practical limit is about 70 characters if there is only one trace.
- Spaces are accepted but not displayed; use underscores.
- The title is annotated as follows:

**Tr 1 New Trace LogM 10.00dB/ 0.00dB**

### How to enter a Trace Title

1. Click the [Trace Status](#) label to select a trace.
2. Click **Trace/Chan**, then **Trace**, then **Trace Title**.



3. Click **Enable**, then type the trace title. Click **Keyboard** to type with a mouse.
4. To remove the trace title, clear the **Enable** checkbox, or delete the text from the dialog entry.

**Channels** contain traces. The PNA can have up to **200 independent channels** (32 before A.09.00)

Channel settings determine **how** the trace data is measured . All traces that are assigned to a channel share the same channel settings. A channel must be selected (**active**) to modify its settings. To select a channel, click the [Trace Status](#) button of a Trace in that channel. The following are channel settings:

- [Frequency range](#)
- [Power level](#)
- [Calibration](#)
- [IF Bandwidth](#)

- [Number of Points](#)
- [Sweep Settings](#)
- [Average](#)
- [Trigger](#) (some settings are global)

## Managing Channels

### How to Select a Channel

A channel must be selected (active) before its settings can be changed.

To make a channel active, [select a trace](#) in that channel.

### How to Turn ON or OFF a Channel

Click **Trace/Chan**, then **Channel**, then **Turn On / Off Channel**.



#### Turn ON | OFF Channel dialog box help

Both the Turn **ON** and Turn **OFF** dialogs work the same.

Select a channel, then click **OK**. Only ONE channel can be selected

When turning ON a channel, the new channel is always the Standard [Measurement Class](#) with an S11 trace.

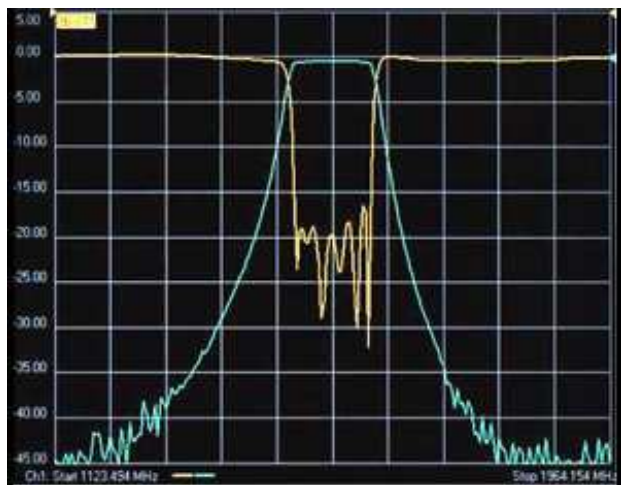
**Note:** To create more than one trace in a new channel, click Trace, then New Trace

**Windows** are used for viewing traces.

- The PNA can show an **UNLIMITED** number of windows on the screen (16 windows previous to PNA release 6.2) with the following limitations:
  - The COM property [MaximumNumberOfWindows](#) returns 1000 ('unlimited' is not a number).
  - The [SCPI status register](#) can only track the status of up to 576 traces.
- Each window can contain up to **24 traces** (8 traces previous to PNA release 8.2).

- Windows are completely independent of channels.
- Learn to [create and manage windows](#).

The following is a window containing two traces. Both traces use the same channel 1 settings as indicated by the annotation at the bottom of the window.



PNA-X shows the window number in the lower-left corner of the window. The following shows window 5.



## Managing Windows -

### How to make various window settings

New, Close, Tile, Cascade, Minimize, Maximize

#### Using front-panel HARDKEY [softkey] buttons

1. Press **RESPONSE**
2. then **[Display]**
3. then **[Windows]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **Windows**

◀ **Programming Commands** ▶



### Close Window dialog box help

Select a window, then click **OK**. The remaining windows are tiled.

Only ONE window can be selected.

Traces contained in a closed window are deleted.

**Note:** To EASILY close a window, click the X in the upper right corner of a window. The X is only visible when [Title Bars](#) are enabled. The remaining windows are NOT tiled.

See [Customize the PNA screen](#) to learn how to make other window settings

Last modified:

4-Sep-2008	Removed legacy content
1-May-2008	Increased max number of traces per window
15-Oct-2007	MX New settings
9/19/06	MQ Modified for unlimited number of windows



## Basic Measurement Sequence

---

The following process can be used to setup all PNA measurements:

### **Step 1. Set Up Measurements**

Reset the analyzer, create a measurement state, and adjust the display.

### **Step 2. Optimize Measurements**

Improve measurement accuracy and throughput using techniques and functions.

### **Step 3. Perform a Measurement Calibration**

Reduce the measurement errors by performing a calibration.

### **Step 4. Analyze Data**

Analyze the measurement results using markers, math operations, and limit tests.

### **Step 5. Print, Save or Recall Data**

Save or print the measurement data.

## Frequency Blanking

**Note:** To learn how to erase memory before moving your PNA out of a secure area, see <http://na.tm.agilent.com/pna/security.html>.

**Note:** An ECal Data Wipe Utility destroys all user data per US DoD 5220.22-M. Learn more at <http://na.tm.agilent.com/pna/apps/applications.htm>

For security reasons, you can prevent frequency information from appearing on the PNA screen and printouts.

### How to set Frequency Blanking

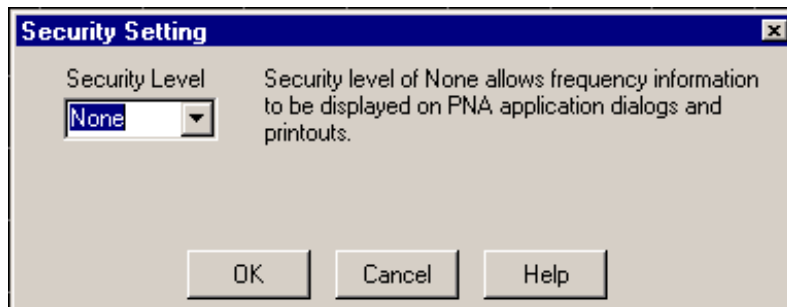
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Security]**

#### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Security**

◀ **Programming Commands** ▶



## Security Setting dialog box help

**None** All frequency information is displayed on the screen and printouts.

### Low security level

Frequency information is blanked from the following:

- Display annotation
- Calibration properties
- All tables
- All toolbars
- All printouts

To re-display frequency information, revisit this dialog box and select **None**.

### High security level

Low security level settings PLUS:

- [GPIB console](#) is inactive

To re-display frequency information, perform an [instrument preset](#), or recall an instrument state with security level of None.

### Extra security level

High security level settings PLUS:

- All ASCII [data saving](#) capability (.snp, .prn, .cti) is saved without frequency information. The X-axis information is replaced with data point numbers. Before A.08.50, saving these file types was NOT allowed.
- [Mixer setup files](#) (\*.mxr) can NOT be saved.

To re-display frequency information, perform an [instrument preset](#), or recall an instrument state with security level of None.

### For ALL security levels:

Frequency information is **NOT** blanked from the following:

- The [Frequency Converter Application](#) ([opt 083](#)) dialog box information or printouts.
- [Service Adjustment Programs](#)
- Your remote COM or SCPI programs.

Last Modified:

17-Feb-2009	Added Extra file saving with data point numbers
4-Sep-2008	Removed legacy content
13-Aug-2008	Added ECal note
29-Apr-2008	Fixed hardkey presses
17-Jul-2007	Added Extra setting

## Internal Second Source

---

The following PNA models include an internal second source.

Model	Total # of Ports
N5230A Opt 146	4
N5230A Opt 246	4
PNA-X Opt 224	<u>2</u>
PNA-X Opt 400	4

### How to use the second source

- Set frequency using the [Frequency Offset Opt 080 dialog](#).
- Set power using the [Advanced Power dialog](#).
- [Source power calibration](#) of the second source is performed as usual.
- Using FCA, [click the LO button](#) to set frequency and power.
- The [specifications](#) of the second source are the same as source 1.
- For PNA-X Opt 224 select **2-port Dual Source** in the [Path Configuration](#) dialog to provide Pulse Modulation on both port 1 and port 2.

### Benefits / Uses of the second source

- Up to five times faster than stepping an external source.
- Measure Mixers with internal swept or fixed LO.
- Measure TOI or Intermodulation distortion.

### Internal Second Source Restrictions

Source 1 and Source 2 are available at specific ports as follows:

#### N5230A (PNA-L) models:

- Source 1 power available at Port 1 OR Port 2; NOT at both ports simultaneously.
- Source 2 power available at Port 3 OR Port 4; NOT at both ports simultaneously.

#### PNA-X Opt 224 (PNA-X 2-port model):



- Source 1 power available at **Port 1** OR **Port 2**; NOT at both ports simultaneously.
- Source 2 (**SRC 2**) power available at **Out 1** AND **Out 2**; BOTH ports simultaneously.
- Select **2-port Dual Source** in the [Path Configuration](#) dialog to provide Pulse Modulation on both port 1 and port 2.

#### **PNA-X Opt 400, 419, 423 (PNA-X 4-port models):**

- Source 1 power available at Port 1 OR Port 2; NOT at both ports simultaneously.
- Source 2 power available at Port 3 AND Port 4; BOTH ports simultaneously.
- Opt 423 ONLY - Source 2 power is available at Port 4 AND either Port 3 OR Port 1 (through the combiner as "Port 1 Src2").

#### **Remotely Accessing the Internal Second Source**

See [Remotely Specifying a Source Port](#).

---

Last modified:

26-May-2009	Added dual source path
6-Apr-2009	Replace N5242A with PNA-X
13-May-2008	Added remote section
1123-Jul-2007	MX Added PNA-X models
10/02/06	MQQ New topic

## Networking and Connecting the PNA

---

### The PNA as a PC

- [PNA User Accounts and Passwords](#)
- [Drive Mapping](#)
- [Connecting the PNA to a PC](#)
- [Easy versus Secure Configuration](#)
- [Changing Network Client](#)
- [Using VNC to Control the PNA User Interface](#)

### GPIB / COM Programming

- [Configure for COM/DCOM Programming](#)
- [Configure for GPIB, SCPI, and SICL](#)

### Controlling External Devices

- [Configure an External Device](#)
- [E5091 TestSet Control](#)
- [External Testset Control](#)
- [Interface Control Feature](#)
- [TestSetIO Connector](#)
- [Handler IO Connector](#)
- [AuxIO Connector](#)

## PNA Preferences

PNA preferences are settings that survive a Preset or PNA Shutdown. PNA Preferences are listed on this page with links to locations that provide more information.

### How to set PNA Preferences

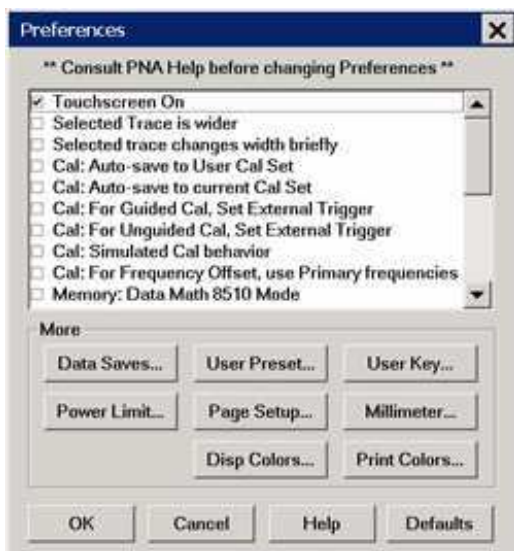
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Configure]**
3. then **[More]**
4. then **[Preferences]**

#### PNA Menu using a mouse

1. Click **Utility**
2. then **System**
3. then **Configuration**
4. then **Preferences**

### Programming Commands



### Preferences dialog box help

PNA Preferences survive a Preset and a system reboot.

A checked box makes the following statements true unless stated otherwise.

#### Touchscreen always ON

Active Trace is always wider.

Active Trace is briefly wider.



### Cal: Auto-save to User Cal Set

This setting only affects calibrations performed using SCPI or COM. Cals performed from the User Interface ALWAYS offer a choice to save to a named Cal Set.

With Rev 6.0 we implemented a change that defaults to saving completed calibrations to Cal Registers instead of User Cal Sets.

Check the box to revert to always automatically saving completed calibrations to an auto-named User Cal Set.

**Caution:** this can cause a lot of save User Cal Sets. [Learn more.](#)

The following message appears when both of the two Cal Set choices are selected:

**"Cal: Auto-save preferences conflict "**

**Auto-save to User Cal Set** (above)- or - **Auto-save to current Cal Set**.(below)

Uncheck one of these.

### Cal: Auto-save to current Cal Set

This setting only affects calibrations performed using SCPI or COM. Cals performed from the User Interface ALWAYS offer a choice to save to a named Cal Set.

Check the box to always automatically save a completed Cal to the Cal Set that is currently selected on the specified channel, which could be the channel Cal Register. If the channel does not yet have a selected Cal Set, the Cal will be saved to a new User Cal Set with an automatically-generated name.

### Cal: For Guided Cal, set external trigger.

With Rev 6.0 we implemented a change that allows the measurement of Cal Standards during a Guided Cal to be externally triggered when [trigger source](#) is set to External.

Check the box to revert to NOT allowing the measurement of Cal standard to be externally triggered. All Guided Cal Standard measurements are triggered internally regardless of the trigger source setting. [Learn more.](#)

### Cal: For Unguided Cal, set external trigger.

Same as above, except for Unguided Cals.

### Cal: Simulated Cal Behavior

For SCPI behavior only. [Learn more.](#)

### Cal: For Frequency Offset, use Primary Frequencies

Use when making mmWave measurements without a test set. [Learn more.](#)

### Memory: Data Math 8510 Mode

Check the box to simulate the Agilent 8510 data processing chain as it pertains to Trace Math and Memory. [Learn more.](#)

### Meas: Mathematical offset for receiver attenuation

By default, the PNA-X offsets the test port receivers by the amount of receiver attenuation. All other PNA models do **NOT** offset the receiver attenuation.

Check the box to offset the receiver attenuation.

Clear the box to NOT offset the receiver attenuation.

[Learn more.](#)

#### **Meas: Mathematical offset for source attenuation**

By default, the PNA offsets the power at the reference receiver by the amount of source attenuation.

Clear the box to NOT offset the source attenuation. [Learn more.](#)

#### **Meas: RF power On during frequency sweep retrace**

By default, the PNA leaves RF power ON during a retrace of single-band frequency or segment sweeps.

Clear the box to turn RF power OFF during a retrace of single-band frequency or segment sweeps. [Learn more.](#)

#### **Meas: Power Sweep Retrace**

By default, at the end of a power sweep, while waiting to trigger the next sweep, the PNA maintains source power at the start power level.

Clear the box to maintain source power at the STOP power level. [Learn more.](#)

#### **Meas: External Trigger OUT is Global**

By default, the PNA-X trigger output properties follow the main Trigger Scope setting. For example, when a channel is in point sweep mode, the trigger output will automatically be sent per point.

External triggering for all other PNA models defaults to Global. To send external trigger signals for each point, the Trig OUT Per Point setting must also be checked.

Check the box to set to Global.

Clear the box to set to Channel. [Learn more.](#)

#### **Meas: Port 1 Noise Tuner Switch state**

By default, the Noise Figure port 1 DPDT switch is set to EXTERNAL. This setting always provides incident power through the front panel loop.

Clear the box to set to Internal, which is one method to make accurate S-parameter measurements when an ECal module is connected as a Noise Tuner. [Learn more.](#)

The Noise Figure App will throw the switch as needed. However, External Test Sets (in Multiport mode) rely on this switch being set to External.

#### **Meas: Draw failed trace segments in red**

By default, failed data points (dots) are drawn in red.

Check the box to draw failed segments in red. [Learn more.](#)

#### **Report source unleveled events as errors**

For SCPI behavior only. [Learn more.](#)

#### **Ext Device: de-activate on PRESET and recall.**

By default, external devices are de-activated when the PNA is Preset or when a Instrument State is recalled.

Clear the box to have external devices remain active when the PNA is Preset or when a Instrument State is

recalled.

[Learn more about External Devices.](#)

### On PRESET set two-point group delay aperture

By default, group delay aperture is set to 11 points.

Check the box to have group delay aperture set to 2 points. [Learn more.](#)

### On Preset turn power ON

By default (checked), Instrument Preset always turns RF power ON.

Checkbox cleared - When the current power setting is OFF, leaves power OFF after Preset. When the current power setting is ON, turns power ON after Preset. [Learn more.](#)

The **More** buttons launch dialogs that contain predefined preferences:

#### Define Data Saves -

While not explicitly called Preferences, all of these settings survive a shutdown. [Learn more.](#)

#### User Preset

Specify the Instrument State file that the PNA will use when Preset. [Learn more.](#)

#### User Key

Sets softkey preferences. [Learn more.](#)

#### Page Setup

Standard printer settings (Paper, Orientation, and Size) do NOT survive a PNA shutdown.

All other settings DO survive a PNA shutdown. [Learn more.](#)

#### Millimeter settings

Sets MM Wave configurations. [Learn more.](#)

#### Power Limit

Sets Power Limits and Offsets. [Learn more.](#)

#### Disp Colors

Sets PNA display items to custom colors. [Learn more.](#)

#### Print Colors

Sets PNA print items to custom colors. [Learn more.](#)

Although they are called preferences, the following settings do NOT survive a PNA shutdown.

Calibration	UI Setting
Show or not, the first 'Method' Page of the Cal Wizard.	<a href="#">Cal Preferences</a>
Set and order default Cal Types	<a href="#">Cal Preferences</a>
Perform orientation of the ECal module during calibration?	<a href="#">ECal Wizard</a>
Specify ECal port mapping when orientation is OFF	<a href="#">ECal Wizard</a>

Last Modified:

- 24-Mar-2010    Removed Error messages button
- 31-Jul-2009    Added External Device, primary FOM, draw failed segments, Power Limit, disp colors, print colors (9.0)
- 6-Feb-2009    Added new Cal set selection
- 22-Apr-2008    Added UI
- 10-Apr-2008    Added RemoteCalStorage and replaced AuxTrigger properties
- 5-Feb-2007    MX New topic

## Using VNC to Control the PNA User Interface

---


VNC (Virtual Network Computing) allows you to control the User Interface of a PNA from any PC. The PNA display appears on the connected PC display. Mouse and keyboard control can occur from both the PNA and PC, although not simultaneously.

Both the PNA and PC must be connected to the internet. The responsiveness of the PNA while using VNC is dependent of the speed of your internet connection.

Every PNA is shipped with VNC installed. However, you must download and install the VNC software onto the PC. The following procedures can help you configure VNC to view and control the PNA application from your PC.

### On the PNA, run VNC Server

To do this:

1. Click View, then Minimize Application.
2. Click **Start**, then **Programs**, then **TightVNC**, then **Launch VNC Server**.
  - When the server is running, the  icon is visible in the lower right corner of the display.
  - The first time you run VNC Server, you may be required to set a password to control access from remote PCs.
  - To automatically start VNC when the PNA computer boots, drag a **Launch VNC Server** shortcut to your User "startup" folder. The following is the Administrator folder: C:/Documents and Settings/Administrator/Start Menu/Programs/Startup.

### On the PC, run VNC Viewer

To do this:

1. Download from <http://www.tightvnc.com/> and install TightVNC on the PC.
2. From the PC Desktop, click **Start**, then **Programs**, then **TightVNC**, then **TightVNC Viewer**
3. When prompted for the Hostname, type the [full computer name](#) or [IP address](#) of the PNA.
4. When prompted for the password, type the password you set when configuring VNC on the PNA.


---

Last Modified:

15-Apr-2008    Uptightened for DatedVNC

## LXI Compliance

Only PNA-X models that are shipped with PNA version A.08.20 and higher are LXI compliant.

A PNA is LXI compliant if the  logo appears on the PNA splash screen when the PNA application starts. Learn more about LXI at <http://www.lxistandard.org/>

## LAN Status

When a LAN connection is used with the PNA, the LAN Status dialog allows you to see the IP address and other LAN connection properties.

### How to view LAN Status

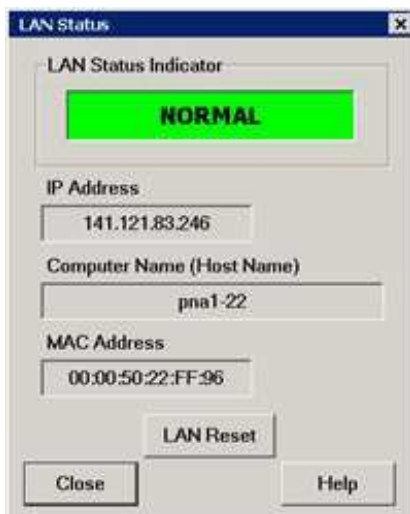
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Configure]**
3. then **[More]**
4. then **[LAN Status]**

#### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Configure**
4. then **More**
5. then **LAN Status**

**Programming Commands**



## LAN Status dialog box help

**Indicator** Shows the current status of the LAN connection.

**NORMAL** - Indicates that the PNA LAN is ready for communication.

**IDENTIFY** - Indicates that a remote computer has invoked an LXI identification operation on the PNA using the web-based interface or [LXIDeviceIDState](#) COM property.

**FAULT** - Indicates that the PNA LAN interface is not connected to the Internet.

**IP Address** Shows the current IP address of the PNA

**Computer Name** Shows the full computer name of the PNA. [Learn how to change this.](#) If you see the IP address listed here, that means there is no DNS server specified in the network setup.

**MAC Address** Shows the unique address of the PNA computer. Also known as HostID.

**LAN Reset** Provides a LAN Configuration Initialize (LCI) mechanism. Press to return the following settings to factory default conditions:

- **IP Address Configuration (DHCP):** Enabled
- **ICMP Ping Responder:** Enabled
- **Web Password for configuration:** Resets the password for the factory default account (Agilent) to 'agilent'. If the **pna-admin** account had been renamed by the PNA owner, this function creates a new Administrator account named **pna-admin** and sets its password to 'agilent'. Learn more about [User names and Passwords.](#)

## Web Server Software

If your PNA is LXI Class C compliant ([see above](#)), you can connect to the PNA using a web browser over an internet connection.

To do this, when the above dialog indicates a **NORMAL** condition:

1. From a web browser, type **http://<your\_PNA\_computer\_name>**. For example, to connect to the fictitious PNA in the dialog above, type: **http://pna1-22**
2. Type the log on User Name and Password
3. You will see the following welcome screen with connection links.

Welcome 9TUCLY4-HAN@na-admin | Software | Products | Agilent Site


# Agilent Technologies PNA Series Network Analyzer

Another web-enabled instrument from Agilent Technologies

Welcome to your **Web Enabled PNA**

Information about this Web-Enabled PNA:

Description:	PNA Series Network Analyzer
Hostname:	study4-pna
IP Address:	141.104.04.100
VISA TCP-Connect String:	tcpip0:study4-pna:hpib7,16:inst
LXI Version:	1.1
LXI Class:	C



Toggle LXI Identification

Advanced Information about this Web-Enabled PNA:

Model Number:	N5242A
Options:	400,004,010,080,082,083,086,006,423,551,000,021,022,025,014,061,502
Serial Number:	US46200061
Software Revision:	Z.06.10.18
Ethernet (MAC) Address:	00:00:50:29:B5:21
SCPI TCP- Socket Port:	5025
SCPI Telnet Port:	5024
SCPI Interface Name:	lan(141.104.04.100)hpib7,16
Auto-MDIX Capable:	No

© Agilent Technologies, Inc. 2007

Last Modified:

3-Jun-2010    MX New topic



## Using Help

---

Help Rev. 3-Jun-2010  
PNA Rev. A.09.22  
© Agilent Technologies, Inc. 2010

This topic discusses the following:

- [PNA Documentation](#)
- [Printing Help](#)
- [Copying Help to your PC](#)
- [Launching Help](#)
- [Navigating Help](#)
- [Help Languages](#)
- [Glossary](#)
- [Dialog Boxes](#)
- [Documentation Warranty](#)
- [Suggestions Please](#)

### See Also

[Help, About Network Analyzer](#)

### [Other Quick Start Topics](#)

## PNA Documentation

This Help file, which is embedded in the PNA, is the **Users Guide and Programming Manual for the PNA**. The help file is automatically updated on the PNA when firmware is updated. Only the PNA [Installation and Quick Start Guide](#) is shipped with new PNA instruments.

Hardcopy manuals are no longer available for purchase with the PNA.

All PNA documentation, including the **latest online Web Help version** of this Help file, and a printable .PDF version of the Help file, are available at <http://na.tm.agilent.com/pna/help/index.html>.

## Printing Help

Beginning with the PNA 5.2 release (March 2005), we once again offer a .pdf version of PNA Help. Download the .pdf file from <http://na.tm.agilent.com/pna/help/index.html>. You can still print individual PNA Help topics by clicking the Print icon at the top of the PNA Help window.

## Copying Help to your PC

With the Help system on your PC, you can read about the PNA while away from it. You can also Copy and Paste programming code from this Help system directly into your programming environment.

The Help file is located on your PNA hard-drive at **C:/ Winnt/ Help/ PNAHelp.chm**. If both the PNA and PC are connected to LAN, you can [map a drive](#) and copy the file directly.

The Help file can also be downloaded from <http://na.tm.agilent.com/pna/help/index.html>.

## Launching Help

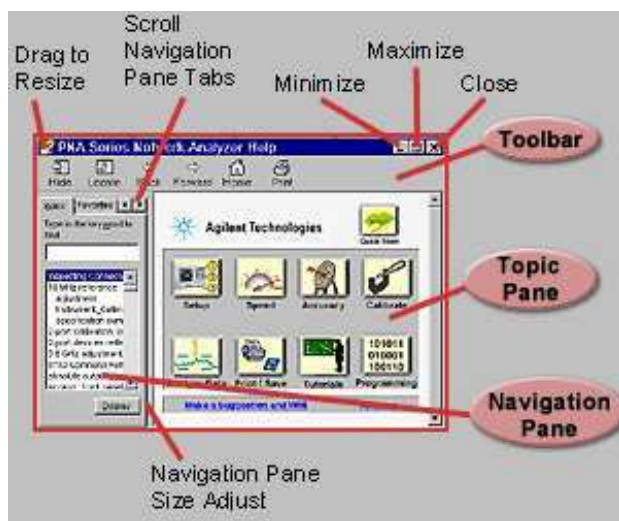
The Help system can be launched in three ways:

1. From the front panel Help button.
2. From the **Help** drop-down menu
3. From [Dialog Box](#) Help

## Navigating Help

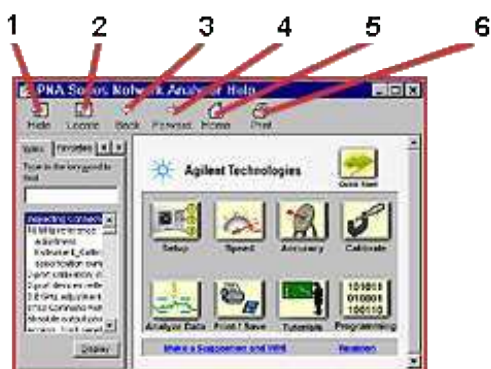
The Help Window contains 3 panes (regions):

1. [Toolbar](#) Pane
2. [Topic](#) Pane
3. [Navigation](#) Pane



### Toolbar Pane

The Toolbar is at the top of all Help windows. It allows you to resize the window, browse and print the selected topic.



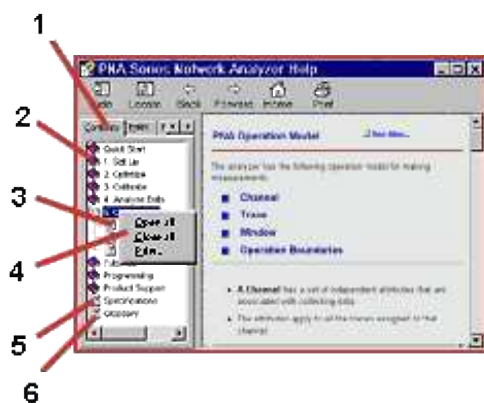
1. Hide or show the navigation pane
2. Locate the topic in the table of contents
3. Back to topic visited previously
4. Forward again if **Back** was clicked
5. Go to the Home page.
6. Print the topic pane.

## Navigation Pane

Click the following tabs in the Navigation Pane to access information in the Help system:

- [Table of Contents Tab](#)
- [Index Tab](#)
- [Search Tab](#)
- [Favorites Tab](#)

## (Table of )Contents Tab



1. Click tab to select Table of Contents.
2. Click a book to access related topics.
3. Click to display a topic.
4. Right click to access menu.
5. Click to display specifications
6. Click to display glossary

## Index Tab

The index tab allows you to type a keyword and go to only the most applicable topics.



1. Click tab to select index.
2. Type keyword to find topics of interest.
3. View suggested topics. (Double-click to display topic.)
4. Click to display topic.

## Search Tab

**TIP:** To Search any topic for a keyword, press **Ctrl** and **F**.

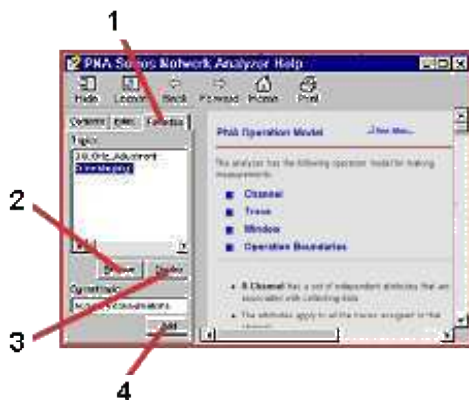
The following rules apply for using full-text search:

- Searches are not case-sensitive.
- You can search for any combination of letters (a-z) and numbers (0-9).
- Punctuation marks (period, colon, semicolon, comma, and hyphen) are ignored during a search.
- You can group the words of your search using double quotes or parentheses. Examples: "response calibration" or (response calibration). This requirement makes it impossible to search for quotation marks.
- Use Wildcard expressions:
  - To search for one undefined character use a question mark (?). For example, searching for **cal?** will find **calc** and **calf**.
  - To search for more than one undefined character use an asterisk (\*). Searching for **Cal\*** will find **calibration** and **calculate**.
- Use Boolean operators to define a relationship between two or more search words.

Search for	Example	Results will show topics containing:
Two words in the same topic	response AND calibration	Both the words "response" and "calibration".
Either of two words in a topic	response OR calibration	Either the word "response" or the word "calibration" or both.
The first word without the second word in a topic	response NOT calibration	The word "response" but not the word "calibration".
Both words in the same topic, close together.	response NEAR calibration	The word "response" within eight words of the word "calibration".

## Favorites Tab

The favorites tab allows you to store (bookmark) the topics you refer to most often so that they can be recalled easily.



1. Click tab to view stored topics in Favorites.
2. Remove selected topic.
3. Display selected topic.
4. Add (store) current topic.

## Topic Pane

The Topic pane allows you to view the contents of the selected topic.

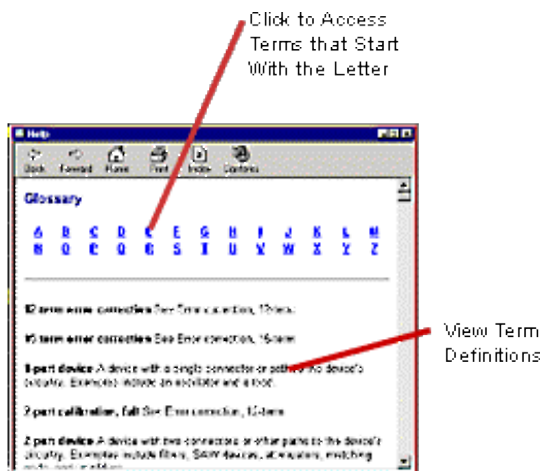


## Help Languages

Beginning with PNA Rev A.08.00, PNA Help is offered in English ONLY.

## Glossary

The [Glossary](#) holds definitions of words, in alphabetical order.



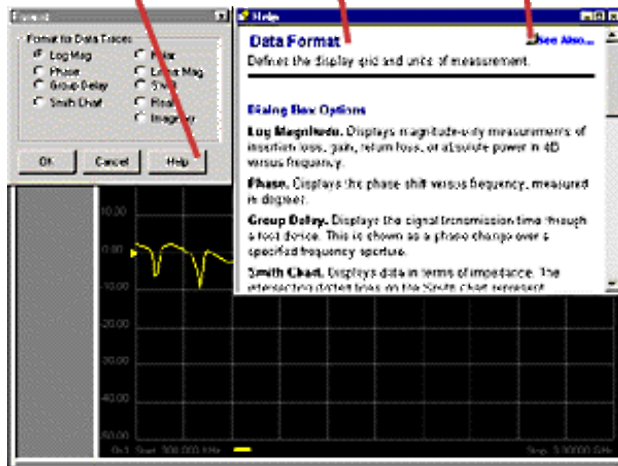
**Note:** Click on a word in **green** text throughout Help to see the glossary definition.

## Dialog Boxes

Click to Access Information on This Dialog Box

Help Topic That Appears

Click to See More Information



## Documentation Warranty

THE MATERIAL CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS," AND IS SUBJECT TO BEING CHANGED, WITHOUT NOTICE, IN FUTURE EDITIONS. FURTHER, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AGILENT DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED WITH REGARD TO THIS MANUAL AND ANY INFORMATION CONTAINED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

AGILENT SHALL NOT BE LIABLE FOR ERRORS OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, USE, OR PERFORMANCE OF THIS DOCUMENT OR ANY INFORMATION CONTAINED HEREIN. SHOULD AGILENT AND THE USER HAVE A SEPARATE WRITTEN AGREEMENT WITH WARRANTY TERMS COVERING THE MATERIAL IN THIS DOCUMENT THAT CONFLICT WITH THESE TERMS, THE WARRANTY TERMS IN THE SEPARATE AGREEMENT WILL CONTROL.

## Suggestions Please!

Please let us know about your experience using PNA Help. Send your comments to:

[pna\\_help@am.exch.agilent.com](mailto:pna_help@am.exch.agilent.com). Comment about any aspect of the help system. Here are a few areas that you might consider:

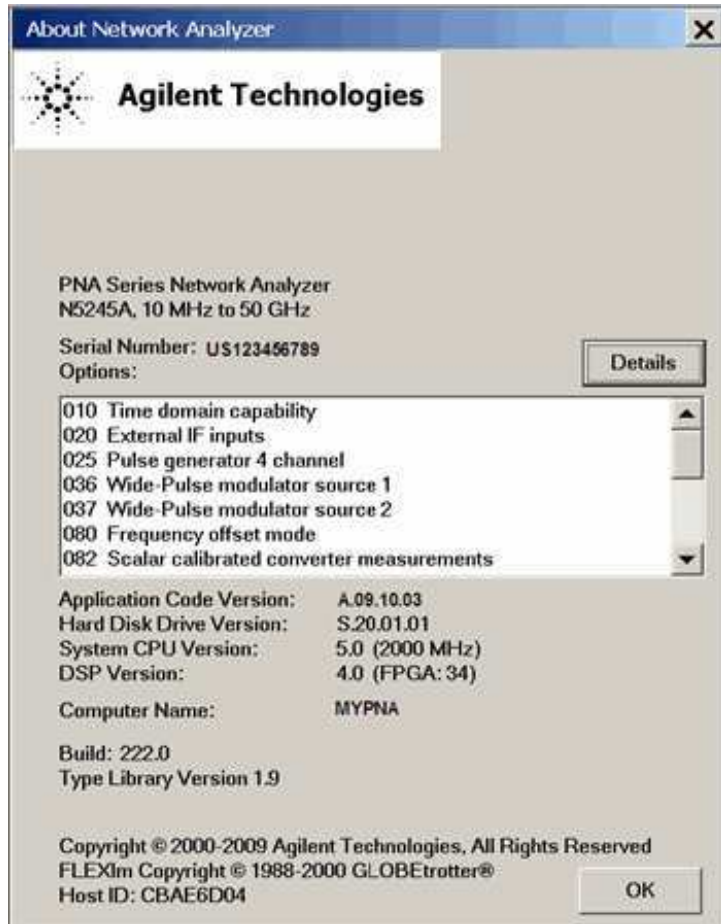
- Does anything appear to be broken?
- Did you find what you were looking for?
- Was the information you found helpful?
- Any suggestions as to how we can improve the help system?

Your comments go directly to the help system authors. For help with technical questions, please refer to [Technical Support](#).

## Help - About Network Analyzer

---

Click **Help**, then **About Network Analyzer** to learn the following about your PNA:



- Model number ([see list of PNA models](#))
- Frequency range
- Serial number
- Installed options ([Learn how to install software options](#))
- Application Code (firmware) Version
- Hard Disk Drive Version
- System CPU Version - [Learn more](#)
- DSP Version - Digital Signal Processor. For PNA-X, this number, along with the FPGA version, provides access to the following features:
  - 4.0 FPGA: 27 - Required for [FastCW triggering](#)



- 4.0 FPGA: 34 - Required for [configurable \(polarity and type\) for PulseSyncIn](#) external trigger signals.
- To access these features, [Contact Agilent](#) to upgrade the DSP.
- Computer Name - [Learn more](#). This is also reported on the [LAN Status / LXI Compliance dialog](#).

## Programming Commands

---

Last Modified:

30-Nov-2009    MX New topic

## Preset the PNA

---

When you Preset the PNA, it is set to known, or preset conditions. You can use the factory default preset conditions, or define your own User Preset conditions.

- [Preset \(Default\) Conditions](#)
- [User Preset Conditions](#)

[See other 'Setup Measurements' topics](#)

## Preset Default Conditions

### How to Preset the PNA

**Tip:** Press the **Preset** button to start the PNA application if it is not already running.

#### Using front-panel HARDKEY [softkey] buttons

1. Press **Preset**

#### PNA Menu using a mouse

1. Click **Utility**
2. then **Preset**

◀ **Programming Commands** ▶

Click to view the **factory preset conditions**.

- [Frequency Settings](#)
- [Power Settings](#)
- [Sweep Settings](#)
- [Segment Sweep Settings](#)
- [Trigger Settings](#)
- [Display Settings](#)
- [Response Settings](#)
- [Calibration Settings](#)
- [Marker Settings](#)
- [Limit Test Settings](#)

- [Time Domain Settings \(Option 010\)](#)
- [Global Display Settings](#)

## Frequency Settings:

Measurement Parameter	S11
Start Frequency	Minimum frequency of the PNA
Stop Frequency	Maximum frequency of the PNA
CW Frequency	1 GHz

See the [PNA configurations](#) for the minimum and maximum frequency of your PNA

## Power Settings:

Test Port	0 dBm for E8356/7/8A
Power	0 dBm for E8801/2/3A 0 dBm for N3381/2/3A  -5 dBm for N5230A - 20 GHz -10 dBm for N5230 - 40 GHz -15 dBm for N5230 - 50 GHz  -12 dBm for E8362/3/4 A or B, standard -15 dBm for E8361A -17 dBm for E8362/3/4 A or B with option UNL or 014

Power On - See also [Power ON and OFF during Save / Recall, User Preset, and Preset.](#)

Port Power Coupled	On
Auto Attenuation	On
Attenuator Value	0 dB
Power Slope	Off
Slope Value	0 dB/GHz

### Sweep Settings:

Type	Linear Frequency
Mode	Continuous
Generation	Analog
Auto Sweep Time	On
Number of Points	201

### Segment Sweep Settings:

Active Segments	1
Start Frequency	<a href="#">PNA start frequency</a>
Stop Frequency	1 MHz for E8356/7/8A 1 MHz for E8801/2/3A 1 MHz for N3381/2/3A 1 GHz for E836xA/B
Number of Points	21
Power	PNA preset <a href="#">test port power</a>
IF Bandwidth	50 KHz for N5230A 35 kHz for all other models
<a href="#">Reduce IF BW at Low Frequencies</a>	ON
<a href="#">Dwell Time</a>	0

### Trigger Settings

Source	Internal
Mode	Sweep

### Display Settings:

Format	Log Mag
--------	---------

These settings apply for formats when selected:

Format	Scale	Reference Position	Reference Value
Log Mag	10 dB/	5	0 dB
Phase	45 degrees/	5	0 degrees
Group Delay	10 nsec/	5	0 s
Linear Mag	100 munits/	0	0 units
SWR	1 unit/	0	1 unit
Real	2 units/	5	0 units
Imaginary	2 units/	5	0 units
Polar	1 unit/	n/a	1 unit
Smith Chart	1 unit/	n/a	1 unit

## Response Settings:

Channel Number	1
IF Bandwidth	50 KHz for N5230A 35 kHz for all other models
Averaging	Off
Averaging Factor	1
Smoothing	Off
Smoothing Factor	1% of span
Electrical Delay	0 s
Velocity Factor	1.0
Phase Offset	0 degrees
Math/Memory Trace View	Data

### Calibration Settings:

Correction State	Off
Interpolation State	On
Calibration Type	None
Cal Kit Number	Current Cal Kit Number
System Z0	50 ohms
Port Extensions State	Off
Port Ext. Values	0
Input A, B	
Port 1, 2	

### Marker Settings:

Initial Frequency	Current Center Frequency
Reference	None
Interpolation	On
Format	Trace Default
Type	Normal
Function	Max Value
Domain	Full Span
Table	Empty
Coupling	Always uncoupled

### Limit Test Settings:

Limit Testing	Off
Line Display	ON
Sound on Fail	Off

### Limit List Settings:

Type (OFF, MAX, MIN) OFF

Begin Stimulus 0

End Stimulus 0

Begin Response 0

End Response 0

### Time Domain Settings:

Transform State Off

Transform Mode Band Pass

Transform Start -10 ns

Transform Stop 10 ns

Window 6.0 (Kaiser-Bessel factor)

Gating State Off

Gating Start -10 ns

Gating Stop 10 ns

Gate Type Band Pass

Gate Shape Normal

### Global Display Settings:

Trace Status On

Frequency/ Stimulus Off

Marker Readout On (when a marker is activated)

Toolbars Shown Active Entry

Status Bar State ON

## User Preset Conditions

The analyzer can be **preset** to either **factory default** conditions or **User Preset** conditions.

## How to set User Preset

### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[User Preset]**

### PNA Menu using a mouse

1. Click **Utility**
2. then **User Preset**

## Programming Commands



### User Preset dialog box help

With a User Preset saved and enabled, when the PNA is Preset, the User Preset settings are recalled instead of the factory default settings. Calibration data is NOT recalled with a User Preset. [Learn more about instrument state settings.](#)

#### User Preset Enable

Check - The PNA is preset to **User Preset** conditions when the Preset button is pressed.

Clear - The PNA is preset to **Default** conditions when the Preset button is pressed.

**Save current state as User Preset** Click to store the current instrument state as the User Preset conditions. File is stored as C:/ Program Files/ Agilent/ Network Analyzer/ Documents/ UserPreset.sta.

**Load existing file as User Preset** Click to retrieve an instrument state to be used as the User Preset conditions.

Last modified:



9-Mar-2010	Added link to Power ON/OFF
3-Sep-2008	Removed legacy content
9/27/06	MX Added UI
9/12/06	Added link to programming commands

## Measurement Parameters

---

This topic contains the following information:

- [S-Parameters](#) (pre-selected ratios)
- [Ratioed](#) (choose your own ratio)
- [Unratioed Power](#) (absolute power)
- [How to Select a Measurement Parameter](#)

[Learn about Balanced Measurements](#)

[See other 'Setup Measurements' topics](#)

### S-Parameters

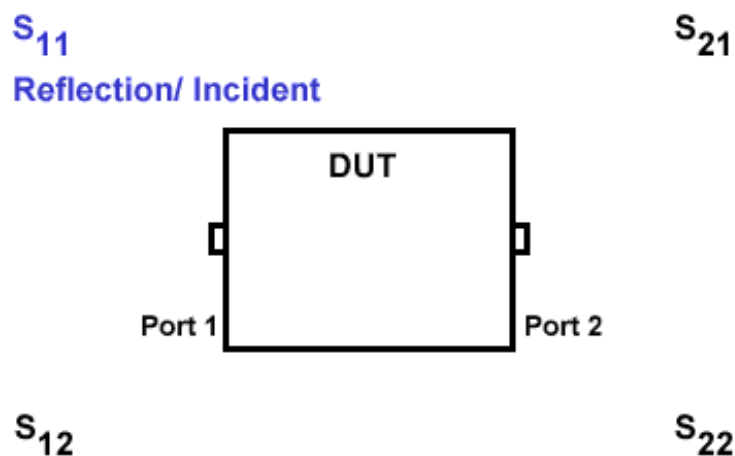
S-parameters (scattering parameters) are used to describe the way a device modifies a signal. For a 2-port device, there are **four S-Parameters**. The syntax for each parameter is described by the following:

**S out - in**

**out** = PNA port number where the device signal output is measured (receiver)

**in** = PNA port number where the signal is applied (incident) to the device (source)

Move the mouse over each S-parameter to see the signal flow:



For two-port devices:

- When the source goes into port 1, the measurement is said to be in the **forward** direction.

- When the source goes into port 2, the measurement is said to be in the **reverse** direction.

The analyzer automatically switches the source and receiver to make a forward or reverse measurement. Therefore, the analyzer can measure all four S-parameters for a two-port device with a single connection.

See the [block diagram](#) (including receivers) of your PNA.

## Common Measurements with S-Parameters

### Reflection Measurements (S11 and S22)

- Return loss
- Standing wave ratio (SWR)
- Reflection coefficient
- [Impedance](#)
- S<sub>11</sub>, S<sub>22</sub>

### Transmission Measurements (S21 and S12)

- [Insertion loss](#)
- Transmission coefficient
- Gain/Loss
- Group delay
- Deviation from linear [phase](#)
- [Electrical delay](#)
- S<sub>21</sub>, S<sub>12</sub>

## Receiver Measurements

A 2-port PNA typically has four receivers: A, B, R1, and R2.

Your PNA may not have 2 reference and 2 test port receivers. See the [block diagram](#) of your PNA.

- R1 and R2 are reference receivers. They measure the PNA source signal as it leaves the PNA and is incident on the DUT.
  - R1 measures the signal out of Port 1
  - R2 measures the signal out of Port 2.
- A and B are test port receivers. They measure the signal out (or reflecting off ) of the DUT.
  - A measures the signal into PNA Port 1
  - B measures the signal into PNA Port 2

You can specify measurements using one or two of the available receivers.

**Note:** Beginning with PNA Rev. 7.22, you can use the internal ADC (Analog-Digital Converters) as measurement receivers. [Learn more.](#)

## Ratioed Measurements

Ratioed measurements allow you to choose your own ratio of any two receivers that are available in your PNA. S-parameters are actually predefined ratio measurements. For example S<sub>11</sub> is A/R1.

The following are common uses of ratioed measurements:

- Comparing the phase between two paths of a device. An example could be something simple like a power splitter or more complicated like a dual-channel receiver.
- Measurements that require a higher dynamic range than the analyzer provides with S-parameters.

Your PNA **MAY** have front-panel jumper cables that go directly to measurement receivers. Learn about the [front-panel jumpers](#) on your PNA.

### Unratioed (Absolute Power) Measurements

The unratioed power parameter allows you to look at the absolute power going into any of the measurement receivers that are available on your PNA.

The reference receivers are internally configured to measure the source power for a specific PNA port. Performing an absolute power measurement of a reference receiver using a different source port will measure very little power unless the front panel jumpers are removed and signal is applied directly to the receiver. An example of this would be an R1 measurement using port 2 as the source.

- [Measuring phase](#) using a single receiver yields meaningless data. Phase measurements must be a comparison of two signals.
- [Averaging](#) for Unratioed parameters is computed differently from ratioed parameters.

### How to create a NEW trace

#### Using front-panel HARDKEY [softkey] buttons

#### PNA Menu using a mouse

#### For [PNA-X](#) and ['C' models](#)

The only measurements that can be created are those in the same measurement class as is currently assigned to the active channel. To create a measurement other than these, first assign the appropriate measurement class to a new or existing channel. [Learn how.](#)

After that is done...

1. Press **TRACE 1, 2, 3, OR 4**

1. Click **Trace/Chan**
2. then **New Trace**

### Programming Commands

### How to CHANGE the active trace

1. Press **MEAS**
2. then **select a new parameter**

1. Click **Response**
2. then **Measure**
3. then **select a new parameter**

## New / Change Measurement dialog box help

**Note:** The only measurements that are available are those in the [measurement class](#) currently assigned to the active channel. Other measurements are NOT compatible.

To create a measurement other than these, first assign the appropriate measurement class to a new or existing channel. [Learn how.](#)

Click a tab to create or change measurements.

- When creating NEW measurements, you can choose more than one.
- When changing an EXISTING measurement, you can choose ONLY one.

### Tabs

**S-Parameter** Select a predefined ratioed measurements. [Learn more about S-parameters.](#)

**Balanced** Select a balanced measurement type. (Multiport PNAs ONLY)

**Change** Click to invoke the [Balanced DUT Topology / Logical Port mappings](#) dialog box. [Learn more about Balanced Measurements.](#)

**Receivers** Select receivers to make Ratioed and Unratioed (absolute power) measurements. [Learn more about receiver measurements.](#)

### Logical Receiver Notation

With PNA [Rev 6.2](#), Ratioed and Unratioed measurements can also use **logical receiver notation** to refer to receivers. This notation makes it easy to refer to receivers with an [External Test Set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port.

- **aN** - Reference receiver for logical port N
- **bN** - Test port receiver for logical port N

For example:

- **Ratioed:** "b12/a1" refers to the logical test port 12 receiver / the logical port 1 reference receiver.
- **Unratioed:** "b10" refers to the logical test port 10 receiver.

The old style notation (A, B, R1 and so forth) can still be used to refer to the PNA **physical** receivers.

However, ratioed measurements MUST use the same notation to refer to both receivers; either the physical receiver notation (A, R1) or the logical receiver notation (aN, bN). For example, the following mixed notation is NOT allowed: A/b3 and a5/R2.

## Programming

When entering receiver letters using programming commands, neither logical or physical receiver notation are case sensitive.

**Ratioed** Check **Activate** to create or change a measurement. Select a receiver for the Numerator, select another receiver for the Denominator, then select a source port for the measurement.

The **Source port** is ALWAYS interpreted as a logical port number.

For convenience, the table is populated with common choices.

- [Learn about External Test Sets and Ratioed Measurements](#)
- [Learn more about Ratioed Measurements.](#)

**Unratioed** Same as Ratioed, but select **1** as the Denominator.

- [Learn More about Unratioed Measurements.](#)
- See the [block diagram](#) of receivers in YOUR PNA.
- The internal ADCs (Analog-Digital Converters) can be used as measurement receivers. [Learn more.](#)

## Channel / Window Selections

These selections are NOT AVAILABLE when changing an EXISTING measurement. [Learn how to change a measurement.](#)

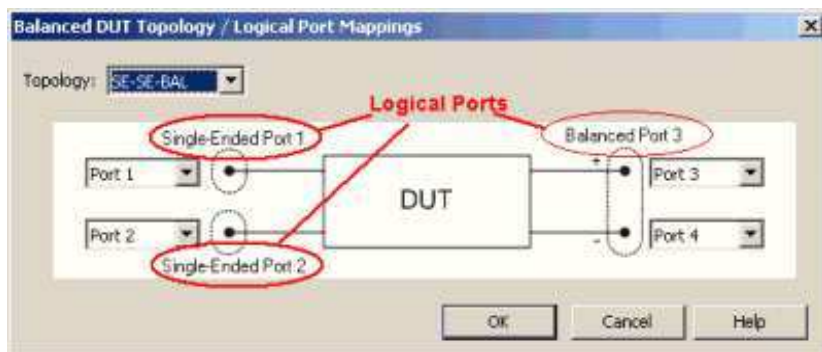
**Channel Number** Select the channel for the new traces.

### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the PNA [traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See PNA [number of windows limitation](#).

[About Measurement Parameters](#) (top of page)



### Balanced DUT Topology / Logical Port mappings dialog box help

See this dialog for [Integrated True Mode Stimulus Application](#) (iTMSA).

Create or edit DUT Topology and Logical Port Mapping.

A Logical Port is a term used to describe a physical PNA test port that has been remapped to a new port number.

- Any **Two** physical PNA ports are mapped to **One Balanced** Logical port
- Any **One** PNA physical port is mapped to **One Single-Ended** Logical port

**Note:** These selections apply to ALL measurements in the channel. If the device topology is changed, any existing measurements in the channel that are incompatible with the new topology will be automatically changed to one that is compatible.

**Topology:** Describes your DUT as you would like it tested. The following device topologies can be measured by a multiport PNA.

- **Balanced / Balanced**  
(2 logical ports - <4 actual ports>)
- **Single-ended / Balanced**  
(2 logical ports - <3 actual ports>)
- **Single-ended - Single-ended / Balanced**  
(3 logical ports - <4 actual ports>)

These topologies can be used in the reverse (<==) direction to measure:

- **Balanced / Single-ended** topology
- **Balanced / Single-ended - Single-ended** topology

For example, to measure a **Balanced / Single-ended** topology, measure the S12 (reverse direction) of a **Single-ended / Balanced** topology.

- Learn about [Logical Port mapping when using an External Test Set.](#)

- Learn more about [Balanced Measurements](#)
- Balanced parameters can be saved to SNP files. [Learn more.](#)

---

Last modified:

2-Mar-2010	Link to SNP files
10/11/06	Added new UI
9/19/06	MQ Added logical receiver notation and Multiport meas toolbar.
9/12/06	Added link to programming commands



# Measurement Classes

Measurement Classes are categories of measurements that can coexist on a channel.

- [What are Measurement Classes](#)
- [How to assign a Measurement Class to a Channel](#)
- [Measurement Class Dialog Box Help](#)

[See other 'Setup Measurements' topics](#)

## What are Measurement Classes

The following table shows the three Measurement Classes currently available for the PNA. Within each of these classes there are a number of measurements.

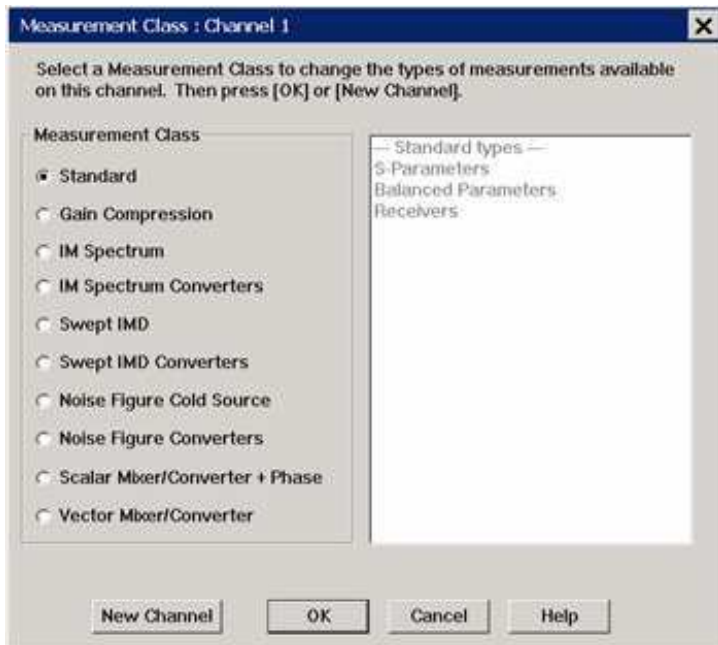
Measurement Classes are categories of measurements that can coexist on a channel. A measurement from one class can NOT reside in a channel with a measurement from another class. For example, a VC21 measurement can NOT reside in a channel that is currently hosting Scalar Mixer Measurements.

The Measurement Class dialog is accessed in the following ways:

### How to assign a Measurement Class to a Channel

Using front-panel HARDKEY [softkey] buttons	PNA Menu using a mouse
<ol style="list-style-type: none"><li>1. Press <b>MEAS</b></li><li>2. then <b>[Measurement Class]</b></li></ol>	<ol style="list-style-type: none"><li>1. Click <b>Trace/Chan</b></li><li>2. then <b>Measurement Class</b></li></ol>

[Programming Commands](#)



### Measurement Class dialog box help

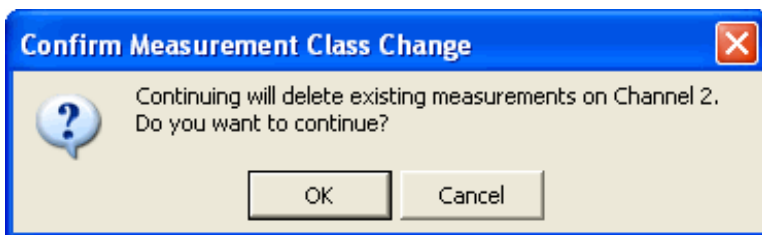
Measurements in a measurement class can NOT coexist in a channel with a measurement of a different measurement class. Select a measurement class for the active measurement channel.

**Title Bar** Indicates the active channel to which the measurement class will be assigned.

**Measurement Class** Choose the measurement class.

**Note:** The list of measurements is provided for display only. If you choose to create the measurement class in a new channel, a default measurement (usually S11) will be created. To change the measurement, click **Trace**, then select a new measurement.

**Next** Click to invoke the following dialog. NOT available when the selected measurement class is the same as the active channel.



Choose to do the following:

- **OK** - Delete the existing measurements in the active channel. Create the new measurement class, and default measurement, in that channel.
- **Cancel** - Do not create the new measurement class. Leave the old measurements (and class) in that channel and return to the Measurement Class dialog box.

Last Modified:

22-Mar-2010	Updated dialog image
3-Sep-2008	Removed legacy content
18-Jun-2007	MX New topic

## Frequency Range

---

Frequency range is the span of frequencies you specify for making a device measurement.

- [How to Set Frequency Range](#)
- [Zoom](#)
- [CW Frequencies](#)
- [Frequency Resolution](#)
- [Frequency Band Crossings](#)

[See other 'Setup Measurements' topics](#)

### How to set Frequency Range

There are two ways to set the frequency range:

- A. Specify the **Start** and **Stop** frequencies of the range.
- B. Specify the **Center** frequency and desired **Span** of the range.

See the [frequency ranges of all PNA models](#)

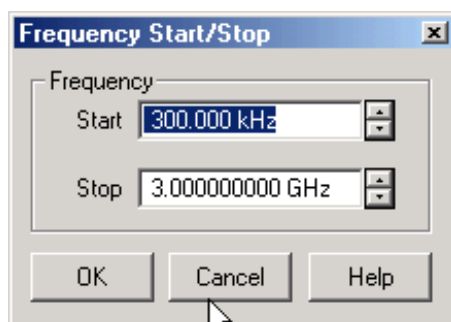
#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQUENCY**

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Frequency**

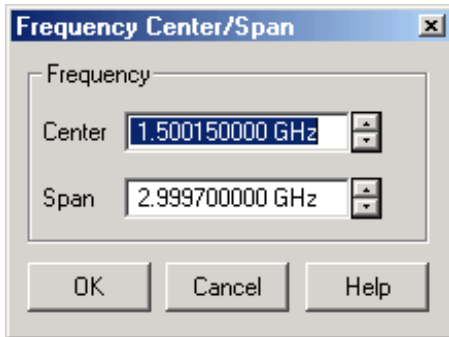
◀ Programming Commands ▶



### Frequency Start/Stop dialog box help

**Start** Specifies the beginning frequency of the swept measurement range.

**Stop** Specifies the end frequency of the swept measurement range.



### Frequency Center/Span dialog box help

**Center** Specifies the value at the center of the frequency sweep. This value can be anywhere in the analyzer range.

**Span** Specifies the span of frequency values measured to either side of the center frequency.

### Zoom - PNA-X ONLY

Zoom allows you to easily change the start and stop frequencies or start and stop power levels in a [power sweep](#).

Zoom operates on the [Active Trace](#) and all traces in the same channel as the active trace, regardless of the window in which they appear.

### How to Zoom in a measurement window

1. Left-click the mouse or use a finger, then drag across a portion of a trace.
2. Release the mouse or lift the finger and the following menu appears:
3. Select from the following:



**Zoom** - changes the channel stimulus settings to the left and right border values of the Zoom selection

**Zoom xy** - changes the channel stimulus settings as above. In addition, the Y-axis scale of the active trace changes to the approximate scale of the Zoom selection.

**Zoom Full Out** - changes the channel stimulus settings to the full span of the current calibration. If no calibration is ON, then the stimulus settings are changed to the full span of the PNA model.

## Notes

- The stimulus settings are changed for **ALL** traces in the active channel, regardless of the window in which they appear.
- If markers are in the selected area, they remain in place.
- If markers are in the unselected area, they are moved to the right or left edge of the new span. When Zoom Full Out is selected, the markers are moved back to their original location.

Zoom is NOT available for the following:

- Smith Chart or Polar [display formats](#)
- [CW Time](#) and [Segment sweep type](#)
- [Frequency Offset Measurements](#)
- [FCA Opt 083 Measurements](#)

## CW Frequencies

Measurements with a [CW Time sweep](#) or [Power sweep](#) are made at a single frequency rather than over a range of frequencies.

### How to set CW Frequency

1. Set [Sweep Type](#) to **CW Time** or **Power**.

You can also set CW frequency from within the Sweep Type dialog box.

**Using front-panel  
HARDKEY [softkey] buttons**

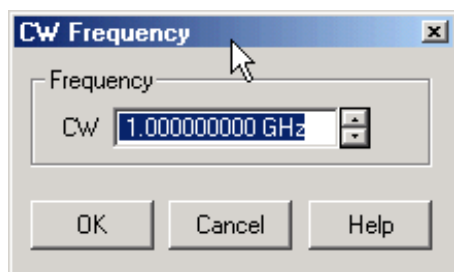
For [PNA-X](#) and ['C' models](#)

2. Press **FREQ**
3. then **[CW]**

**PNA Menu using a mouse**

◀ **Programming Commands** ▶

2. Click **Stimulus**
3. then **Frequency**
4. then **CW Frequency**



#### CW Frequency dialog box help

**CW** Type a value and the first letter of the suffix (k,m,or g) or use the up and down arrows to select any value within the range of the PNA.

## Frequency Resolution

The resolution for setting frequency is 1 Hz.

## Frequency Band Crossings

The frequency range of the PNA covers several internal frequency bands. The higher the frequency range of the PNA, the larger the number of bands. The source power to your DUT turns off as the stimulus frequency is swept through these band crossings. To learn more, see [Power ON and OFF during Sweep and Retrace](#).

The listed frequencies in the following tables are the stop frequency of the specified band, and the start frequency of the following band.

Frequency band crossings are different for the following models:

- [3 GHz, 6 GHz, and 9 GHz Models](#)
- [E8362A/B, E8363A/B, E8364A/B](#)
- [E8361A](#)
- [N5230A \(2-port models\)](#)
- [N5230A \(4-port models\)](#)
- [NPNA5242A](#)

**For 3 GHz, 6 GHz, and 9 GHz (discontinued) PNA models:**

Band	Frequency
1	10 MHz
2	748 MHz
3	1500 MHz
4	3000 MHz
5	4500 MHz
6	6500 MHz

**For E8362 / 63 / 64 A/B/C**  
(A models do not have band 0)

Band	Freq (GHz)	Band	Freq (GHz)	Band	Freq (GHz)
0	.045				
1	0.748	9	7.60	17	25.60
2	1.500	10	10.00	18	30.00
3	3.00	11	12.00	19	32.00
4	3.80	12	12.8	20	36.00
5	4.50	13	15.20	21	38.40
6	4.80	14	16.00	22	40.00
7	6.00	15	20.00	23	45.60
8	6.40	16	22.80	24	48.00

**For E8361A/C**



Band	Freq (GHz)	Band	Freq (GHz)	Band	Freq (GHz)
0	.045	9	10.00	18	32.00
1	0.748	10	12.00	19	36.00
2	1.500	11	12.80	20	40.00
3	3.00	12	15.40	21	44.70
4	3.80	13	16.00	22	46.20
5	4.00	14	20.00	23	51.20
6	4.50	15	24.00	24	60.00
7	6.00	16	25.60	25	64.00
8	7.70	17	30.00		

**For N5230A/C 2-port models**

Band	Freq (GHz)	Band	Freq (GHz)	Band	Freq (GHz)
1	.045	11	10.500	21	28.600
2	.748	12	12.500	22	31.250
3	1.5	13	15.750	23	31.500
4	3.125	14	16.667	24	33.333
5	4.167	15	18.750	25	37.000
6	5.250	16	21.000	26	40.500
7	6.250	17	22.500	27	41.667
8	7.875	18	25.000	28	42.000
9	8.333	19	26.250	29	46.800
10	9.375	20	26.500	30	

**For N5230A/C 4-port models**

Band	Freq (GHz)	Band	Freq (GHz)
0	.00112	9	8.333
1	0.010	10	9.800
2	0.040	11	10.500
3	0.748	12	12.500
4	1.500	13	15.000
5	3.125	14	15.750
6	4.166	15	16.666
7	5.250	16	18.750
8	6.250	17	20.100

**For PNA-X (Band Stop Frequencies)**

**N5241A and N5242A**

Band	Freq (GHz)	Band	Freq (GHz)	Band	Freq (GHz)
0	Reserved	12	.396	24	8.50
1	Reserved	13	.500	25	10.664
2	.014	14	.628	26	12.00
3	.019	15	1.00	27	12.80
4	.027	16	1.50	28	13.51
5	.038	17	2.00	29	15.40
6	.053	18	3.00	30	16.00
7	.075	19	3.20	31	18.00
8	.105	20	4.00	32	20.00
9	.146	21	5.332	33	21.328
10	.205	22	6.752	34	22.50
11	.250	23	8.00	35	24.00

**N5244A and N5245A**

<b>Band</b>	<b>Freq (GHz)</b>	<b>Band</b>	<b>Freq (GHz)</b>	<b>Band</b>	<b>Freq (GHz)</b>	<b>Band</b>	<b>Freq (GHz)</b>
<b>0</b>	Reserved	<b>12</b>	.396	<b>24</b>	8.50	<b>36</b>	27.008
<b>1</b>	Reserved	<b>13</b>	.500	<b>25</b>	10.664	<b>37</b>	32.00
<b>2</b>	.014	<b>14</b>	.628	<b>26</b>	12.00	<b>38</b>	36.50
<b>3</b>	.019	<b>15</b>	1.00	<b>27</b>	12.80	<b>39</b>	40.50
<b>4</b>	.027	<b>16</b>	1.50	<b>28</b>	13.51	<b>40</b>	42.656
<b>5</b>	.038	<b>17</b>	2.00	<b>29</b>	15.40	<b>41</b>	43.50
<b>6</b>	.053	<b>18</b>	3.00	<b>30</b>	16.00	<b>42</b>	46.20
<b>7</b>	.075	<b>19</b>	3.20	<b>31</b>	19.00	<b>43</b>	48.00
<b>8</b>	.105	<b>20</b>	4.00	<b>32</b>	20.00	<b>44</b>	50.20
<b>9</b>	.146	<b>21</b>	5.332	<b>33</b>	21.328		
<b>10</b>	.205	<b>22</b>	6.752	<b>34</b>	24.00		
<b>11</b>	.250	<b>23</b>	8.00	<b>35</b>	26.50		

## Last modified:

- 9-Mar-2010    Made minor corrections
- 6-Apr-2009    Modified models
- 3-Sep-2008    Removed legacy content
- 10/23/06    MX Added new band crossings
- 10/16/06    Moved phase lock lost indicator
- 9/12/06    Added link to programming commands
- 9/27/06    MX Added UI



## Power Level

---

Power level is the power of the PNA source at the test ports.

- [How to make Power Settings](#)
- [Power Dialog](#)
- [Power and Attenuator Dialog](#)
- [Source Unleveled](#)
- [Setting Independent Port Power](#)
- [Optimum Attenuation Value](#)
- [Receiver Attenuation](#)
- [Power ON and OFF during Save / Recall, User Preset, and Preset](#)
- [Power ON and OFF during Sweep and Retrace](#)

[See other 'Setup Measurements' topics](#)

## Power Settings

The test port output power is specified over frequency ( See the [Power Range](#) and [Frequency Range](#) specifications for your PNA)

### How to make Power settings

Use one of the following methods to set port power. Only the menu can be used to launch the [Power and Attenuators](#) dialog box.

#### Using front-panel HARDKEY [softkey] buttons

1. Press **POWER**
2. then **[Power]** or **[Power and Attenuators]**

#### Using a mouse with PNA Menus

1. Click **Stimulus**
2. then **Power** or [Power and Attenuators](#)

[Programming Commands](#)



### Power dialog box help

Defines and controls the PNA source power and attenuation.

**Power On (All Channels)** Check to enable source power for all channels. Only turns power ON if channel power setting is ON or Auto. [See Advanced Power.](#)

**Port 'n'** Active source port for which power is being set.

**Port Power** Sets the power level for the specified port.

- To accurately set the power level at any point after the test port, perform a [Source Power Calibration](#).
- See the [specified power range of your PNA model](#).

### Power Sweep

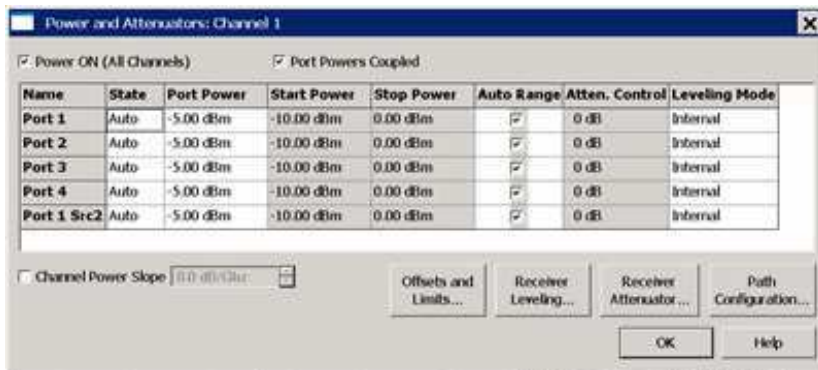
**Start / Stop Power** Set the start and stop power values of a power sweep. These settings are only available when [Sweep Type](#) is set to Power Sweep. Uncoupled power sweep power can be set from the [Advanced Power dialog box](#).

**Zoom** - allows you to easily change the start and stop power levels in a power sweep. [Learn how.](#)  
[Learn more about Power Sweep.](#)

### Power Slope

Helps compensate for cable and test fixture power losses at increased frequency.

**Slope** Select to set the power slope. Clear to set power slope OFF. [Learn more about power slope.](#)



## Power and Attenuators dialog box help

Defines and controls the PNA source power and attenuation for the active channel.

Beginning with PNA Rev. 7.21, external sources can be controlled from this dialog. [Learn more.](#)

**Power On (All Channels)** Check to enable source power for all channels. Only turns power ON if channel power setting is ON or Auto.

### Port Powers Coupled

- **Coupled** (checked) The power levels are the same at each test port. Set power at any test port and all test ports change to the same power level.
- **Uncoupled** (cleared) The power levels are set independently for each test port. Uncouple power, for example, if you want to measure the gain and reverse-isolation of a high-gain amplifier. The power required for the input port of the amplifier is much lower than the power required for the output port. A power sweep can also be performed with uncoupled power. Learn more about [Setting Independent Port Power](#)

**Name** Lists the PNA test ports.

**State** ON and OFF are valid ONLY on PNA models with an [internal second source](#).

- **Auto** Source power is turned ON at the specified test port when required by the measurement. This is the most common (default) setting. See also [Power ON and OFF during Save / Recall, User Preset, and Preset.](#)
- **ON** Source power is ALWAYS ON, regardless of measurements that are in process. Use this setting to supply source power to a DUT port that always requires power, such as an LO port. This could turn OFF power at another test port. [Learn about internal second source restrictions.](#)
- **OFF** Source power is never ON, regardless of the measurement requirements. Use this setting to prevent damage to a sensitive DUT test port.

**Port Power** Sets the power level at the output of the source.

- To accurately set the power level at any point after the test port, perform a [Source Power Calibration](#).
- See the [specified power range of your PNA model](#).
- See [ECal Module Compression Level](#)

**Start / Stop Power** Available ONLY when sweep type is set to Power Sweep. Set the start and stop power values of a power sweep. [Learn more about Power Sweep.](#)

- In PNA release 6.04 you can specify whether to maintain source power at either the start power or stop power level at the end of a power sweep. [Learn more.](#)
- In PNA release 8.20, a power sweep can be performed with [uncoupled power](#). Different power ranges can be swept in the forward and reverse directions.

**Auto Range** Check to allow the PNA to select the [optimum attenuation value](#) to achieve the specified test port power.

Clear to manually set the attenuation for each port. Type or select the attenuation value in the adjacent Attenuator Control box.

**Attenuator Control** When Port Powers are Uncoupled, manual attenuator control allows you to set a wide range of power levels by setting the attenuation. [See Setting Independent Port Power.](#) Also use manual attenuation control when a measurement requires a very good impedance match with the source, such as with oscillators or conditionally unstable amplifiers. Choose an attenuation level of 10 dB or more to ensure the best source match.

- The PNA does not allow attenuators or other mechanical switches to switch continuously. [Learn more.](#)
- When Port Powers are Coupled, changing one port Attenuation Control value changes all port values.
- Attenuators are located between the source and the test port. Power to the reference receiver is not attenuated and is therefore higher than at the test port by the amount of attenuation. This will make uncalibrated measurements that use a reference receiver appear as though there is added attenuation at the test device. [See the PNA Block diagram.](#)

**Note:** With PNA release 7.2, a preference can be set to mathematically offset (or NOT) the power at the reference receiver by the amount of source attenuation. Because the reference receiver is not in the attenuation path, there is more power at the reference receiver than at the test port by the amount of source attenuation.

By default, ALL PNA models currently offset the reference receivers. [See Block diagram.](#)

Learn how to [set the preference](#).

### Leveling Mode

- **Internal** Standard ALC leveling. Power level within an attenuator setting is limited to the ALC Range. [See Source Unleveled.](#)
- **Receiver Rx** Select a Reference receiver to use for leveling the source. [Learn more.](#)
- **Open Loop** (Used during pulse conditions with the internal source modulators). No leveling is used in setting the source power. The lowest settable power, without attenuation, is limited to -30dBm. The source power level accuracy is very compromised. Use a source power calibration to make the source power somewhat more accurate.



## Channel Power Slope

Helps compensate for cable and test fixture power losses at increased frequency. With power slope enabled, the port output power increases (enter positive value) or decreases (enter negative value) as the sweep frequency increases.

**Slope** Select to set the power slope. Clear to set power slope OFF.

Power slope is computed and applied from 0 GHz – not from the measurement start frequency. For example, with the following measurement settings:

- Start / Stop Freq: 10 GHz to 20 GHz
- Power level: 0 dBm
- Slope: 1 dB/GHz

The power into the DUT from 10 GHz to 20 GHz is 10 dBm sloping to 20 dBm

**Offset and Limits** Launches the [Offset and Limits](#) dialog.

**Receiver Leveling** Launches the [Receiver Leveling](#) dialog.

**Receiver Attenuator** Launches the [Receiver Attenuator](#) dialog.

**Path Configurator** Launches the [Path Configurator](#) dialog.

## Source Unleveled

When the power level that is required at a test port is **higher** than the PNA can supply, a Source Unleveled [error message](#) appears on the screen and the letters LVL appear on the [status bar](#).

To perform a power sweep, the range of power is usually limited to the range of the Automatic Leveling Control (ALC) loop. (The PNA-X allows a very wide power range using **Open Loop**).

PNA specifications guarantee the ALC power range over which the PNA can supply power without an unleveled indication. However, the actual achievable power range on your PNA is probably greater than the specified range.

### How to calculate the specified achievable power range

From the specifications for the N5230A Opt 245 for the frequency span from 15 GHz to 20 GHz:

- Max Leveled Power = **-8 dBm**
- Power Sweep Range (ALC) = **-17 dB**

For this frequency range the specified power range is calculated as:

- Max = **-8 dBm**
- Min =  $(-8) - (17) = \mathbf{-25\ dBm}$

When using Source Attenuators:

- with 10dB of attenuation, this becomes **-18 dBm to -35 dBm**

- with 20dB of attenuation, this becomes **-28 dBm** to **-45 dBm**, and so forth.

[See the output power specs for your PNA.](#)

To resolve an unlevelled condition, change either the Test Port Power or Attenuator setting. If an Unlevelled condition exists within the specified power range, [contact Technical Support](#).

## Setting Independent Port Power

The PNA allows you to [uncouple port power](#) and specify different power levels at each test port. There are a few things to consider when setting independent port powers.

- Does your required high and low power levels fall within the specified Min and Max power range of the PNA? [See Unlevelled Indicator](#). If they do not, you may need to use the internal Source Attenuators.
- Does the PNA have source attenuators? If so, how many source attenuators? Some PNA models have one attenuator for each port. In most multiport PNA systems, the attenuators are shared by at least two test ports. See [PNA Options](#) to see the availability and range of source attenuation on your PNA.

**Note:** To prevent premature wear, the PNA does not allow attenuators or other mechanical switches to switch continuously.

These mechanical devices are set for the entire channel. When more than one channel is used, and a mechanical device setting is NOT the same for all channels, only the ACTIVE channel is allowed to sweep. All other channels are NOT allowed to sweep (Blocked).

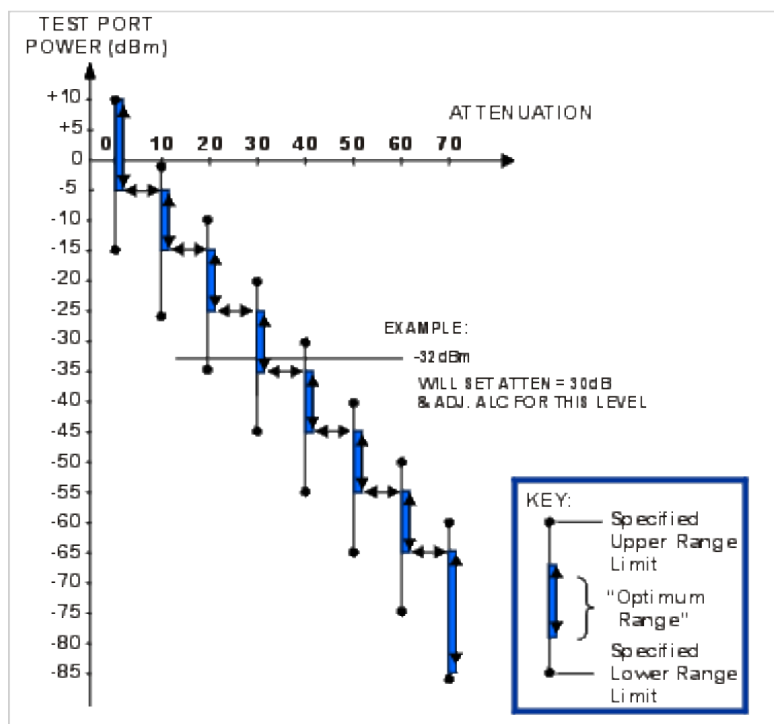
Learn how to [view the settings of all mechanical devices](#) in the PNA.

## Optimum Attenuation Value

The attenuator has different positions, allowing a wide range of power levels. The number of power ranges available is determined by the source attenuation installed in your PNA. See [PNA Options](#) to see the availability and range of source attenuation on your PNA.

- Each range has a total specified span ( 25 dB in the following Attenuation Values graphic ).
- The optimum setting is the middle of the range. This range provides the best accuracy and performance of the source leveling system. The optimum ranges are the blue regions in the following graphic.
- An attenuator setting can be selected manually or automatically. If automatic is selected, the blue optimum ranges (shown in the following graphic) are used.

(Attenuator ranges vary, this particular range is 70 dB)



**Note:** Error correction is fully accurate only for the power level at which a measurement calibration was performed. However, when changing power within the same attenuator range at which the measurement calibration was performed, ratioed measurements can be made with nearly full accuracy (non-ratioed measurements with less accuracy).

**Receiver Attenuators**

Receiver A: 0 dB

Receiver B: 0 dB

Receiver C: 0 dB

Receiver D: 0 dB

OK Help

## Receiver Attenuators dialog box help

Type or select independent attenuation values for each receiver.

Receiver Attenuation, available as [option 016](#) on some PNA models, is used to attenuate the output signal from the device under test to avoid damaging the PNA receivers. The PNA receivers typically start to compress at around +10 dBm. This causes the power level at the receiver to be less than the power at the test port by the specified amount of attenuation.

**Note:** With PNA release 7.2, a preference can be set to mathematically offset (or NOT) the displayed trace by the amount of receiver attenuation. This causes the display to show the power at the test port.

By default:

- PNA-L and E836xB do **NOT** offset the display.
- The PNA-X **DOES** offset the display.

Learn how to [set the preference](#).

When an [external test set](#) is connected, Receiver Attenuation control is only available for the physical receivers in the PNA. Switching receiver attenuation using [logical receiver notation](#) is NOT allowed.

**CAUTION!** You can damage the analyzer receivers if the power levels exceed the maximum values. See your analyzer's [Technical Specifications](#) for the maximum input power to a receiver.

The receiver attenuator characteristics are:

- Range:
  - 0 to 50 dB (E8361A only)
  - 0 to 35 dB (all other applicable PNA models)
- Resolution:
  - 10 dB (E8361A only)
  - 5 dB (all other applicable PNA models)

## Power ON and OFF during Save / Recall, User Preset, and Preset

To protect your DUT from being inadvertently powered ON, the following RF Power ON/OFF settings occur:

### [Instrument State Save/Recall](#)

If power is OFF when an instrument state is saved, then power will always be OFF after the instrument state is recalled.

If power is ON when an instrument state is saved, and the current power setting is OFF, then power will be OFF after the instrument state is recalled.

### [User Preset](#)

If power is OFF when a User Preset is saved, then power will always be OFF after a User Preset.

If power is ON when a User Preset is saved, and the current power setting is OFF, then power will be OFF after a User Preset.

## **Preset**

Instrument Preset sets power ON by default.

This can be changed with a [Preference setting](#) so that, if the current power setting is OFF, then power will be OFF after Preset.

## **Power ON and OFF during Sweep and Retrace**

The frequency range of the PNA covers several internal frequency bands. The higher the frequency range of the PNA, the larger the number of bands. For example, a 9 GHz PNA has 6 frequency bands, a 50 GHz PNA has 25 frequency bands. See the [frequency band crossings](#).

Power to the DUT is turned OFF during band changes to avoid causing power spikes to the DUT.

Retrace occurs when the source gets to the end of your selected frequency span and moves back to the start frequency. Power to the DUT is again turned OFF when **retracing** across frequency bands.

Therefore, the following occurs for various stimulus settings:

1. **Single band sweep** - The power to the DUT is always ON, even during retrace.  
In PNA release 6.04, [a preference setting](#) can turn power OFF during a retrace. Only available in single band frequency and segment sweeps.
2. **Multi-band sweep** - The power to the DUT is turned OFF while sweeping across a band crossing. It is turned OFF again during retrace.
3. **Power sweep** - Because power sweep is always done at a single frequency, the frequency is always within a single band and the source power is always ON. At the end of a power sweep, power is immediately set to the start power.  
In PNA release 6.04, this behavior can be changed with [a preference setting](#).
4. **Single sweep:**
  - Manual trigger mode - At the end of a multiband sweep, power is turned OFF during retrace, and then power is turned back ON before arming for the next trigger.
  - Hold mode - Power can be ON or OFF depending on when and how Hold mode is entered. However, power can be immediately turned OFF manually or remotely.

**Caution:** Avoid expensive repairs to your PNA. Read [Electrostatic Discharge Protection](#).

---

Last modified:

23-Feb-2010	Replaced Hold with Blocked channels
4-Aug-2009	Added Offset and Limits button
9-Feb-2009	Added Receiver Leveling
3-Sep-2008	Removed legacy content
27-Jun-2008	Added link to ECal compression
10-Jun-2008	Clarified power slope
28-May-2008	Updated for uncoupled power sweep
22-Apr-2008	Link to UI preferences
26-Mar-2007	Clarified retrace power OFF
11/16/06	Added new retrace features
10/23/06	Modified for new power diag
10/17/06	Clarified leveling
9/12/06	Added link to programming commands

## Receiver Leveling

Receiver Leveling uses PNA receiver measurements to adjust the source power level across a frequency or power sweep. Before each measurement sweep, a variable number of background sweeps are performed to repeatedly measure power at the receiver for each stimulus point. Those power measurements are then used to adjust the source power level and achieve greater source power level accuracy.

This is similar to a [Source Power Calibration](#), which uses a power meter or PNA receiver to accurately measure source power only during the calibration. The source power correction values are applied for all subsequent measurement sweeps. Receiver leveling provides more accurate source power levels, but also takes longer to perform each measurement sweep.

Receiver Leveling can be used with all [sweep types](#), including Segment sweep and Power sweep.

This feature is available for standard S-parameter measurements.

**Note:** Beginning with PNA Rev. A.09.00, Receiver Leveling can also be used with [FCA](#), [GCA](#), and [IMD](#) applications.

Applications for this feature include:

- Correcting for short term drift when using an external component, such as a booster amplifier. The booster amplifier must be connected to the front-panel jumpers, in front of the reference receiver. See the Block diagram for your PNA, located at the end of every [Specifications document](#).
- Extending the accuracy of power leveling at very low powers where the internal detector may be too noisy.
- Providing controlled power during [Pulsed measurements](#) in an open loop mode.
- Controlling the power at the outputs of [MM-Wave heads](#).

### How to make Receiver Leveling settings

Start the [Power and Attenuators](#) dialog box as follows:

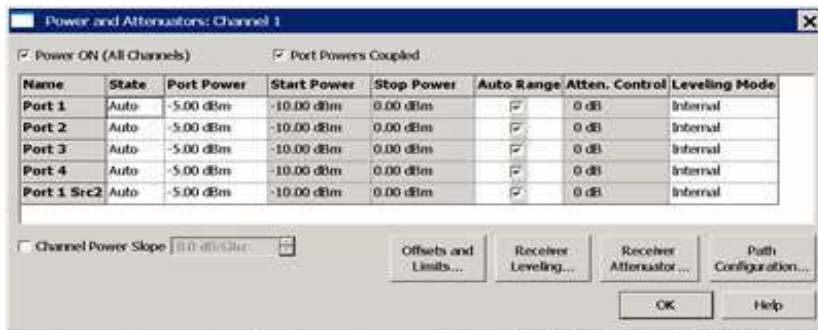
#### Using front-panel HARDKEY [softkey] buttons

1. Press **POWER**
2. then **[Power and Attenuators]**

#### Using a mouse with PNA Menus

1. Click **Stimulus**
2. then **Power**
3. then [Power and Attenuators](#)

◀ **Programming Commands** ▶



Click **Leveling Mode**, then select a receiver to use for leveling. More choices are available on the Receiver Leveling dialog.

Click the **Receiver Leveling** button to launch the Receiver Leveling dialog.



## Receiver Leveling dialog box help

### Overview

To understand Receiver Leveling, the term 'Target' power is defined here to mean the sum of the 'Port Power' as shown on the [Power and Attenuators dialog](#) plus the 'Power Offset' as shown on the Receiver Leveling dialog. When Power Sweep is selected, the 'Port Power' is the power level at each step (stimulus point). [Learn more about Power Offset](#).

$$\text{Target Power} = \text{Port Power} + \text{Power Offset}$$

The goal of Receiver Leveling is to adjust the source power until the measured receiver power is equal to the Target Power.

Leveling sweeps are performed in the background (not visible) before every measurement sweep to measure and apply source correction data.

### Receiver Leveling process:

- For each leveling sweep, source power is applied at each data point and measured by the specified receiver.
- The deviation is calculated between the measured power and the target power.



- The deviation is applied to the current source power, and the updated source power levels are applied on the following leveling sweep.
- This process continues until the receiver power at each data point has achieved the target power within the specified tolerance value, or until the specified number of leveling sweeps (iterations) has been reached. See [Leveling Properties](#) below.

### Select a (Source) Port

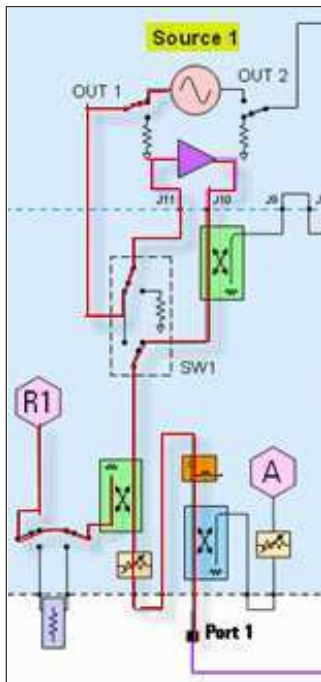
Each source port to be leveled is configured individually. In the box at the left of the dialog, select a source port to be configured for receiver leveling.

### Receiver Selection

**Receiver** Select a receiver to be used to level the specified source.

To level power at the source output or DUT input choose the reference receiver for the source port. For example, to level the source power at port 1, then choose "R1". To level power at the DUT output, choose the receiver that is used to measure the DUT output. If the DUT output is connected to port 2, then select the "B" receiver.

**Power Offset** The 'Port Power' in the Power and Attenuators dialog is normally used as the target leveling power. However, when there exists an additional attenuator or booster amplifier in the source path, an offset should be set to improve the leveling speed.



**Example:** On the PNA-X, a 10dB amplifier is connected in the source path (red line) to the rear-panel J10 and J11.

Port Power is set to 0 dBm (out of Source 1).

Power Offset value is set to +10 dB to compensate for the amplifier.

The target power becomes  $0 + 10 = 10$  dBm.

Source 1 is set to 0 dBm, but the R1 reference receiver expects to see +10 dBm and applies correction based on that value.

Without the Power Offset setting, it would take several leveling sweeps until power was set to the correct value.

**ALC On** The PNA leveling hardware can be in either:

- **ALC ON** - (checked) uses Internal leveling hardware (Recommended)
- **ALC OFF** - (cleared) uses Open Loop hardware.

**Use Last Result for Source Power Cal** - When checked, the latest correction data is copied to the Source Power Cal correction array. When Leveling Mode is switched back to Internal (on the Power and Attenuators dialog), [Source Power Cal](#) is automatically turned ON using this correction data. **Note:** there is NO programming command for this feature.

## Leveling Properties

The following settings specify the quality and speed of leveling that is to be achieved.

If the maximum port power out of the PNA is reached at any time during the leveling sweeps, the following message appears: **Power set to user power limit.**

**Apply Settings to All Ports** Check to apply the following three settings to all PNA ports.

**Tolerance** The source is considered leveled when each stimulus data point has achieved the target power level +/- (plus or minus) this tolerance value.

**Max Iterations** If every stimulus data point does NOT achieve the target leveling power after this number of leveling sweeps, the measurement sweep occurs using the correction values obtained from the last leveling sweep. The message: **Not settled, noisy trace** appears when the Max Iterations is reached. If you see this message, you can increase the Max Iterations, reduce the IFBW, or increase the Tolerance setting.

**Leveling IFBW** By default, the IFBW for the leveling sweeps is set to 100 kHz. [Learn more about IFBW.](#)

- Increase this value to make faster, but noisier leveling sweeps.
- Decrease this value to make slower, more repeatable leveling sweeps.
- Uncheck the box to use the same IFBW as the measurement sweeps.

## Safe Mode

To protect your DUT, these settings control the extent to which the source power will be changed to achieve the target power as measured at the reference receiver. These settings could be necessary when using external components with a large variation in frequency response (flatness).

**Max Power Step** Limits the change in source power at each data point from one sweep to the next. For example, assume Safe Mode is enabled, and Max Power Step is set to 1 dB. On the first leveling sweep, the first data point measures 3 dB lower than the target power, then source power for data point 1 will be increased by 1 dB for the next sweep, and likely for the following two sweeps.

**Max Port Power** Limits the maximum power out of the source. The message: **Power set to Max Power** appears when this limit is reached.

**Min Port Power** Limits the minimum power out of the source. The message: **Power set to Min Power** appears when this limit is reached.

Last Modified:

11-Mar-2010    Added Use Last checkbox (A.09.20)

10-Feb-2009    MX New topic (A.08.50)

## Sweep Settings

---

A sweep is a series of consecutive data point measurements taken over a specified sequence of stimulus values. You can make the following sweep settings:

- [Sweep Type](#)
  - [Linear / Log](#)
  - [Power Sweep](#)
  - [CW Time](#)
  - [Segment Sweep](#)
- [Sweep Time](#)
- [Sweep Setup](#)

See [Triggering](#) and other 'Setup Measurements' topics

### How to set Sweep Type

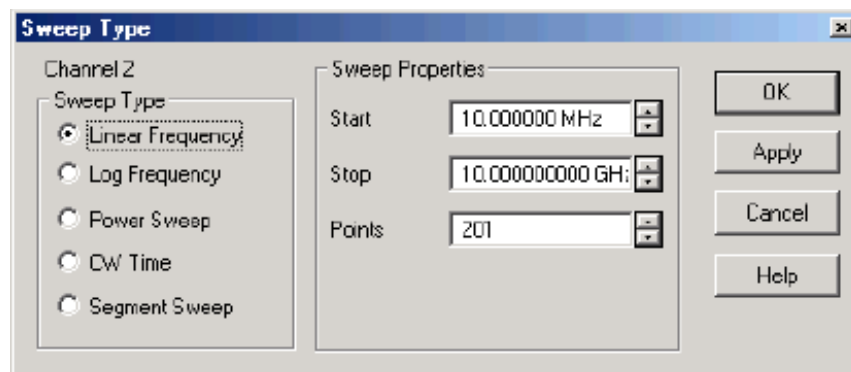
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SWEEP**
2. then [**Sweep Type**]

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Sweep**
3. then **Sweep Type**

### Programming Commands



## Sweep Type dialog box help

**Note:** Sweep Settings are not applied until either **OK** or **Apply** is pressed.

**Channel** The active channel when Sweep Type was selected. Sweep settings will be applied to this channel.

### Sweep Type

**Linear Frequency** Sets a linear frequency sweep that is displayed on a standard grid with ten equal horizontal divisions.

- **Start** Sets the beginning value of the frequency sweep.
- **Stop** Sets the end value of the frequency sweep.
- **Points** Sets the number of data points that the PNA measures during a sweep. Range: 2 to 20001. (Default is 201).

**Log Frequency** The source is stepped in logarithmic increments and the data is displayed on a logarithmic x-axis. This is usually slower than a continuous sweep with the same number of points.

- **Start** Sets the beginning value of the frequency sweep.
- **Stop** Sets the end value of the frequency sweep.
- **Points** Sets the number of data points that the PNA measures during a sweep. Range: 2 to 20001. (Default is 201).

**Power Sweep** Activates a power sweep at a single frequency that you specify. [Learn about power sweep](#)

- **Start** Sets the beginning value of the power sweep.
- **Stop** Sets the end value of the power sweep.
- **CW Frequency** Sets the single frequency where the PNA remains during the measurement sweep.

**CW Time** Sets the PNA to a single frequency, and the data is displayed versus time.

- **CW Frequency** Sets the frequency where the PNA remains during the measurement.
- **Sweep Time** Sets the duration of the measurement, which is displayed on the X-axis.
- **Points** Sets the number of data points that the PNA measures during a sweep. Range: 2 to 20001. (Default is 201).

**Segment Sweep** Sets the PNA to sweep through user-defined sweep segments. [Learn how to make these settings.](#)

- **Independent Power Levels** Check to set the source power level for each segment. [Test port uncoupling](#) is also allowed.

- **Independent IF Bandwidth** Check to set the IF bandwidth for each segment.
- **Independent Sweep Time** Check to set the duration of the measurement for each segment.
- **X-Axis Point Spacing** Check to scale the X-Axis to include only the segments. [Learn more.](#)
- **Allow Arbitrary Segments** Check to allow arbitrary frequencies (overlapped or reverse sweeps). [Learn more](#)
- **Show Table** Shows the table that allows you to create and edit segments.
- **Hide Table** Hides the segment table from the screen.

**OK** Applies setting changes and closes the dialog box.

**Apply** Applies setting changes and leaves the dialog box open to make more setting changes.

**Cancel** Closes the dialog. Setting changes that have been made since the last Apply button click are NOT applied.

## Power Sweep

A power sweep either increases or decreases source power in discrete steps. Power sweep is used to characterize power-sensitive circuits, with measurements such as gain compression.

In the Sweep Type dialog, specify Start power, Stop power, and CW Frequency. Power can be swept over any attainable range within the [PNA ALC range](#).

**Note:** If the PNA has source attenuators, and the attenuation must be changed in order to achieve the requested start and stop power, click **Stimulus**, then **Power** then [Power and Attenuators](#).

The PNA does NOT allow a single power sweep over a range that requires attenuator switching. However, two power sweeps can be performed in different channels. The attenuators will not be allowed to switch continuously, but triggering can be performed using single or group triggering. [Learn more.](#)

The remaining power settings apply in power sweep mode:

- Test Port Power setting is not available.
- Port Power can be coupled or uncoupled.
- Attenuator Control is always Manual.
- Power Slope (dB/GHz) is ignored (output frequency is CW).
- Click **Stimulus** then **Sweep**, then [Number of Points](#) to change the step size of the power sweep.

### Notes:

- Using a [preference setting](#), you can specify whether to maintain source power at either the start power or stop power level at the end of a power sweep.
- Beginning with PNA Rev. A.08.50, Power Sweep has been optimized for speed. For highest measurement

accuracy during a power sweep, it may be necessary to increase the [Dwell Time](#) to allow the source more time to settle.

## Segment Sweep

Segment Sweep activates a sweep which consists of frequency sub-sweeps, called segments. For each segment you can define independent power levels, IF bandwidth, and sweep time.

Once a measurement calibration is performed on the entire sweep or across all segments, you can make calibrated measurements for one or more segments.

In segment sweep type, the analyzer does the following:

- Sorts all the defined segments in order of increasing frequency
- Measures each point
- Displays a single trace that is a composite of all data taken

Restrictions for segment sweep:

- The frequency range of a segment is not allowed to overlap the frequency range of any other segment.
- The number of segments is limited only by the combined number of data points for all segments in a sweep.
- The combined number of data points for all segments in a sweep cannot exceed 20001.
- All segments are FORCED to have power levels within the same attenuator range to avoid premature wear of the mechanical step attenuator. See [Power Level](#).

### How to make segment sweep settings

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SWEEP**
2. then **[Sweep Type]**

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Sweep**
3. then **Sweep Type**

◀ **Programming Commands** ▶

**Insert Segment** - adds a sweep segment before the selected segment. You can also click the "down" arrow on your keyboard to quickly add many segments.

**Delete Segment** - removes the selected segment.

**Delete All Segments** - removes all segments.

**Note:** At least ONE segment must be ON or [Sweep Type](#) is automatically set to **Linear**.

### To Modify an Existing Segment

To make the following menu settings available, you must first show the segment table.

Click **View**, point to **Tables**, then click **Segment Table**.

	STATE	START	STOP	POI	IFBW	P1 PwR	P2 PwR	TIME
1	ON	20.000000 MHz	1.000000 GHz	21	10.0 kHz	17.00 dBm	0.00 dBm	2.474 msec
2	ON	1.000000 GHz	4.000000 GHz	21	35.0 kHz	17.00 dBm	0.00 dBm	630.000 µsec

The above graphic shows the Segment table with all independent settings selected, including source power uncoupled (two power settings).

**STATE** Click the box on the segment to be modified. Then use the up / down arrow to turn the segment ON or OFF.

**START** Sets start frequency for the segment. Click the box and type a value and the first letter of a suffix (KHz, Mhz, GHz). Or double-click the box to select a value.

**STOP** Sets stop frequency for the segment. Click the box and type a value and the first letter of a suffix (KHz, Mhz, GHz). Or double-click the box to select a value.

**Note:** The segment table truncates the frequency resolution. To verify the frequency resolution that you input, create a marker at the start or stop frequency settings.

**POINTS** Sets number of data points for this segment. Type a value or double-click the box to select a value.

### To set IFBW, Power, and Sweep Time independently for each segment:

1. On the **Sweep** menu, click **Sweep Type**, then **Segment Sweep**.
2. Check the appropriate **Sweep Properties** boxes
3. Then click the box and type a value or double-click the box and select a value.

**Note:** If the following are NOT set, the entire sweep uses the channel IFBW, Power, and Time settings.

**IFBW** Sets the [IF Bandwidth](#) for the segment.

**POWER** Sets the [Power level](#) for the segment. You can also UNCOUPLE the test port power. See [Power Coupling](#).

**TIME** Sets the [Sweep time](#) for the segment.

### X-Axis Point Spacing - Segment Sweep ONLY

This feature affects how a segment trace is drawn on the screen.

## How to select X-Axis Point Spacing

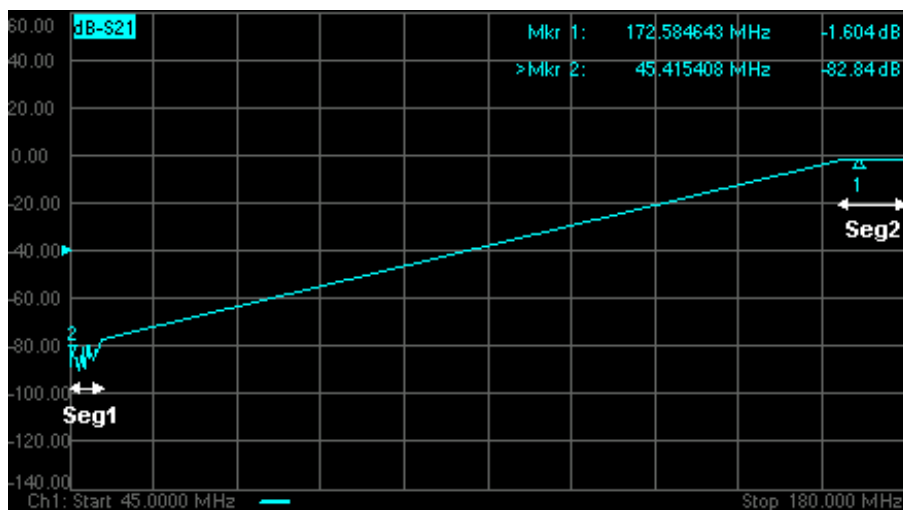
On the [Sweep Type](#) dialog box, click **Segment Sweep**

Then check **X-Axis Point Spacing**

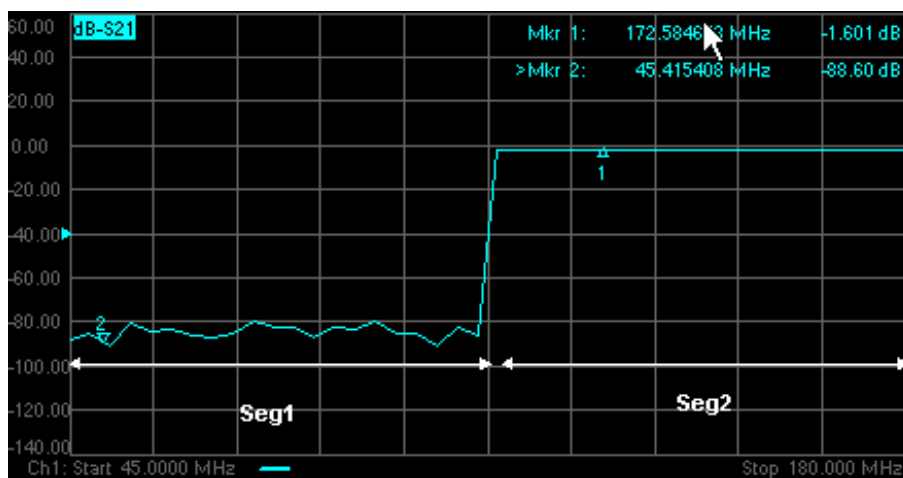
- **Without X-axis point spacing**, a multi-segment sweep trace can sometimes result in squeezing many measurement points into a narrow portion of the x-axis.
- **With X-axis point spacing**, the x-axis position of each point is chosen so that all measurement points are evenly spaced along the x-axis.

For example, given the following two segments:

	STATE	START	STOP	POINTS
1	ON	45.000000 MHz	50.000000 MHz	21
2	ON	170.000000 MHz	180.000000 MHz	21



Without X-Axis Point Spacing



With X-Axis Point Spacing



## Arbitrary Segment Sweep

This feature allows arbitrary frequencies to be entered into the segment sweep table. With this capability, segments can have:

- overlapping frequencies.
- the stop frequency less than the start frequency (reverse sweep).

### How to enable Arbitrary Segment Sweep

1. On the [Sweep Type](#) dialog box, click **Segment Sweep**
2. Check **Allow Arbitrary Segment Sweep**

#### Notes:

- Unusual results may occur when using arbitrary sweep segments with markers, display settings, limit lines, formatting, and some calibration features.
- When Allow Arbitrary Segment is checked, [X-axis point spacing](#) is automatically turned ON.

## Sweep Time

The PNA automatically maintains the fastest sweep time possible with the selected measurement settings. However, you can increase the sweep time to perform a slower sweep.

### How to set Sweep Time

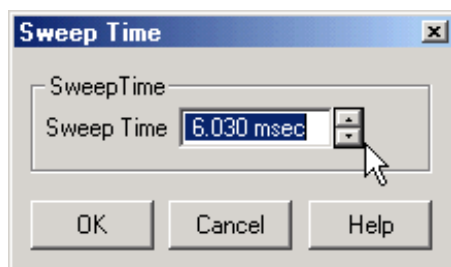
#### Using front-panel HARDKEY [softkey] buttons

1. Press **STIMULUS**
2. then **[Sweep]**
3. then **[Sweep Time]**

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Sweep**
3. then **Sweep Time**


### Programming Commands



## Sweep Time dialog box help

Specifies the time the PNA takes to acquire data for a sweep. The maximum sweep time of the PNA is 86400 seconds or 1 day. [Learn about other settings that affect sweep speed.](#)

**Note:** If sweep time accuracy is critical, use ONLY the up and down arrows next to the sweep time entry box to select a value that has been calculated by the PNA. Do NOT type a sweep time value as it will probably be rounded up to the closest calculated value. This rounded value will not be updated in the dialog box.

- The actual sweep time includes this acquisition time plus some "overhead" time.
- The PNA automatically maintains the fastest sweep time possible with the selected measurement settings. However, you can increase the sweep time using this setting.
- Enter **0** seconds to return the analyzer to the fastest possible sweep time.
- The Sweep Time setting is applied to the active channel.
- The sweep time is per sweep. A full 2-port cal requires two sweeps, both using the specified sweep time. [Learn more.](#)
- A **Sweep Indicator**  appears on the data trace when the Sweep Time is 0.3 seconds or greater, or if trigger is set to [Point](#). The indicator is located on the last data point that was measured by the receiver. If the indicator is stopped (point sweep mode) the source has already stepped to the next data point.

## Sweep Setup

### How to make Sweep Setup settings

#### Using front-panel HARDKEY [softkey] buttons

1. Press **STIMULUS**
2. then **[Sweep]**
3. then **[Sweep Setup]**

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Sweep**
3. then **Sweep Setup**

◀ **Programming Commands** ▶



### Sweep Setup dialog box help

**Channel** Specifies the channel that the settings apply to.

**Stepped Sweep** When checked (Stepped Sweep) the PNA source is tuned, then waits the specified Dwell time, then takes response data, then tunes the source to the next frequency point. This is slower than Analog Sweep, but is more accurate when testing electrically-long devices.

When cleared (Analog Sweep) the PNA takes response data AS the source is sweeping. The sweep time is faster than Stepped, but could cause measurement errors when testing electrically-long devices.

When the dialog checkbox is cleared, the PNA could be in either Analog or Step mode. The mode can change from sweep to sweep. There is **NO way** to determine whether the PNA is in Analog or Stepped Sweep. If you want to be sure what the current sweep mode is, then switch it to Stepped.

Stepped sweep is automatically selected for a number of reasons. Here are some of the reasons:

- [IF Bandwidth](#) is at, or below, 1 kHz.
- [Source Power Correction](#) is ON unless doing CW measurement.
- When more than one source is turned ON ([multisource PNA models](#)).
- When step mode is a faster way to take the data.
- For all [FOM](#) and [FCA](#) measurements.
- For all [ADC measurements](#).
- For all [MMwave measurements](#).

**Dwell Time** Specifies the time the source stays at each measurement point before the analyzer takes the data. Only applies to stepped sweep. The maximum dwell time is 100 seconds. See also [Electrically Long Devices](#).

**Sweep Delay** Specifies the time to wait just before acquisition begins for each sweep. This delay is in addition to Dwell Time (per point) and [External Trigger delay](#) if enabled.

**Fast Sweep** NOT available on [N5264A](#). When checked, in Analog Sweep mode the PNA source settling times are shortened in both frequency and power-control (ALC) circuits. In Stepped Sweep mode, the settling time at ALL data points are shortened. This nearly doubles the sweep speed at preset conditions, but at the expense of frequency accuracy and a few dB of amplitude variation. For ratioed measurements, such as S-Parameters, these errors substantially ratio out.

- By default, Fast Sweep is always OFF to provide maximum accuracy and stability.
- Fast Sweep is available on the [N5230A/C 40 & 50 GHz](#) models, and all [PNA-X models](#).
- Fast Sweep is NOT allowed with [Power Limit](#) enabled.
- **Note:** PNA performance specifications do NOT apply in Fast Sweep.

**Alternate Sweeps** This selection is longer available from the user interface. The inherent [crosstalk](#) on the PNA is sufficient without this feature. Remotely, this setting is still available and sweeps are performed alternately. However, receivers can NOT be turned off individually on the the PNA-L and PNA-X models.

### Sweep Sequence

**Standard Sweep** When checked, the PNA sweeps all data points for each source port in turn. For a 2-port PNA, this means that all data points are swept in the forward direction, then all data points are swept in the reverse direction. Even when NO reverse parameters are displayed (S22 or S12), reverse measurements are necessary when a full 2-port calibration is correcting the channel. This is the default behavior. [Learn more.](#)

**Point Sweep** Available ONLY on standard S-parameter channels.

When checked, the PNA measures all parameters at each frequency point before stepping to the next frequency. The display trace is updated as each data point is measured. Point sweep is the same as stepped sweep mode on the 8510 and 8530.

Last modified:

1-Mar-2010	Sweep Delay and Point sweep now on C models
28-Sep-2009	Added segment frequency resolution note.
6-Apr-2009	Replaced N5242A with PNA-X
5-Mar-2009	Added faster power sweep
17-Feb-2009	Added point sweep availability
26-Jan-2009	Removed 'band-crossings' at fast sweep.
10-Oct-2008	Added point sweep (8.35)
3-Sep-2008	Removed legacy content
19-Aug-2008	Added Fast Sweep
6-Jun-2008	Added uncoupled power sweep
15-Apr-2008	Updated again for Step mode
14-Nov-2007	Added Alternate Sweep note

21-Jun-2007	Increased max data points
3-May-2007	Updated Step mode conditions

## Trigger

---

A trigger is a signal that causes the PNA to make a measurement sweep. The PNA offers great flexibility in configuring the trigger function.

View the interactive [Trigger Model](#) animation to see how triggering works in the PNA.

- [How to Set Trigger](#)
- [Source](#)
- [Scope](#)
- [Channel Settings](#)
- [Restart](#)
- [External and Auxiliary Triggering](#) (separate topic)

[See other 'Setup Measurements' topics](#)

### How to set Triggering

#### Using front-panel HARDKEY [softkey] buttons

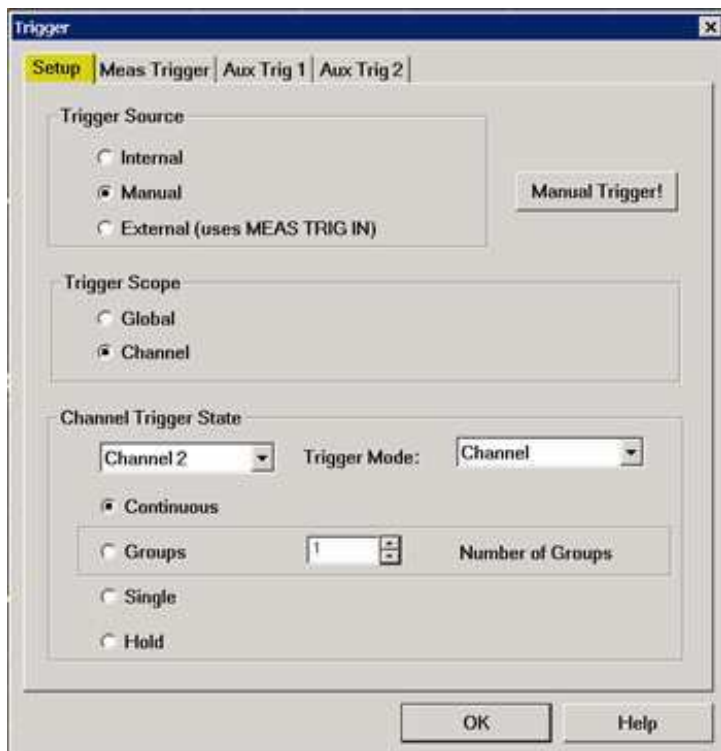
1. Press **TRIGGER**
2. then [**Trigger...**]

#### PNA Menu using a mouse

1. Click **Stimulus**
2. then **Trigger**
3. then **Trigger**

◀ [Programming Commands](#) ▶

**Note:** The **Continuous**, **Single**, and **Hold** settings apply ONLY to the active channel. These settings are available from the Trigger menu, Active Entry keys, and softkeys



### Trigger Setup dialog box help

View the interactive [Trigger Model](#) animation to see how triggering works in the PNA.

#### Trigger Source

These settings determine **where** the trigger signals originate for all existing channels. A valid trigger signal can be generated only when the PNA is not sweeping.

**Internal** Continuous trigger signals are sent by the PNA as soon as the previous measurement is complete.

**Manual** One trigger signal is sent when invoked by the Trigger button, the active toolbar, or a programming command.

**External** Trigger signals sent out or received from various connectors on the rear panel. [Learn more about External and AUX Triggering.](#)

**Manual Trigger!** - Manually sends one trigger signal to the PNA. Available ONLY when Manual trigger is selected.

#### Trigger Scope

These settings determine **what** is triggered.

**Global** All channels not in Hold receive the trigger signal [Default setting]

**Channel** Only the next channel that is not in Hold receives the trigger signal. This is not obvious or useful unless Trigger Source is set to Manual. This setting enables [Point Sweep](#) mode.

#### Channel Trigger State

These settings determine **how many** trigger signals the channel will accept.

**Continuous** The channel accepts an infinite number of trigger signals.

**Groups** The channel accepts only the number of trigger signals that is specified in the Number of Groups text box, then goes into Hold. Before selecting groups you must first increment the Number of Groups text box to greater than one.

**Number of Groups** Specify the number of triggers the channel accepts before going into Hold. If in Point Sweep, an entire sweep is considered one group.

First increment to desired number, then select 'Groups'.

**Single** The channel accepts ONE trigger signal, then goes into Hold.

Another way to trigger a single measurement is to set [Trigger Source](#) to Manual, then send a **Manual trigger**. However, ALL channels are single triggered.

**Hold** The channel accepts NO trigger signals.

## Trigger Mode

These settings determine what EACH signal will trigger.

**Sweep** and **Point** modes are available ONLY when both [Trigger Source](#) = MANUAL or EXTERNAL AND [Trigger Scope](#) = CHANNEL.

- **Channel** Each trigger signal causes **ALL traces** in that channel to be swept in the order specified below.
- **Point** Each Manual or External trigger signal causes one data point to be measured. Subsequent triggers go to the same trace until it is complete, then other traces in the same channel are swept in the order specified below. When in Groups or Single trigger, the count is decremented by one after ALL data points on ALL traces in the channel are measured. See Also, the (point) [Sweep Indicator](#) and [SCPI Triggering example](#) for use with External.
- **Trace** Available ONLY when [Point Sweep](#) is selected. Each trigger signal causes two identical measurements to be triggered separately - one trigger signal is required for each measurement. Other trigger mode settings cause two identical parameters to be measured simultaneously. Trace triggering is NOT permitted when a channel is using a 2 port (or more) S-Parameter calibration.
- **Sweep** Each Manual or External trigger signal causes **ALL traces that share a source port** to be swept in the order specified below. When in Groups or Single trigger, the count is decremented by one after ALL traces in ALL directions are swept.

When multiport correction is ON, which requires sweeps in more than one direction, traces on the screen will not update until all of the relevant directions have been swept. For example, with all four 2-port S-Parameters displayed:

- When Full 2-port correction is ON, trigger 1 causes NO traces to update; trigger 2 causes ALL S-Parameters to update. [Learn more about sweeps with correction ON.](#)
- When correction is OFF, trigger 1 causes S11 and S21 to update; trigger 2 causes S22 and S12 to update.



## Trace Sweep Order

For ALL Trigger Modes, trigger signals continue in the same channel until all traces in that channel are complete. Triggering then continues to the next channel that is not in HOLD.

Traces within each channel are always swept in the following order:

- Traces are swept sequentially in source-port order. For example, in a channel with all four 2-port S-parameters, first the source port 1 traces (S11 and S21) are swept simultaneously. Then the source port 2 traces (S22 and S12) are swept simultaneously.
- In addition, when [Alternate sweep](#) is selected, traces are swept sequentially in source-port / receiver-port order. In the above example, first the S11 trace is swept, then S21, then S12, then S22.

**Restart** (Available only from the Trigger menu) Channels in Hold are set to single trigger (the channel accepts a single trigger signal). All other settings are unaffected, including decrementing trigger Groups.

## See Also

- [External and AUX Triggering.](#)
- Interactive [Trigger Model](#) animation

Last modified:

10-Oct-2008	Added Trace trigger (8.35)
3-Sep-2008	Removed legacy content
26-Oct-2007	Added Trigger Mode
15-Dec-2006	Added MX capability
9/12/06	Added link to programming commands

## External and Auxiliary Triggering

---

External and Auxiliary triggering are both used to synchronize the triggering of the PNA with other equipment or events.

- [Overview](#)
- [Capability Summary for each PNA Model](#)
- Dialogs
  - [Auxiliary Triggering](#) (PNA-X only)
  - [External Trig \(IN\) Dialog](#) (All models)
  - [I/O2 Trig Out Dialog](#) (PNA-L and E836x)

### See Also

- [Controlling a Handler](#)
  - [Synchronizing an External Source](#)
  - [PNA Triggering](#)
  - [Pulse Triggering](#)
- 

## Overview

The manner in which External Triggering is performed has evolved throughout the PNA history, with each new model becoming more comprehensive and flexible. Unfortunately, our ability to update the older models is limited as a large part of external triggering is dependent on the PNA hardware. Where possible, we have updated the capability of the older models with software.

### Ready Signals versus Trigger Signals

A 'Ready for Trigger' signal is different from a Trigger signal. The ready signal indicates that the instrument sending the signal is ready for measurement. The instrument receiving the ready signal would then send a trigger signal, indicating that the measurement will be made, or has been made. Usually the slower instrument sends the trigger signal. The following two scenarios illustrate when the PNA is faster, and slower than the external device:

- A material handler is very mechanical and takes a relatively long time to load and discharge parts. Therefore, the PNA sends a 'Ready' signal when it is setup to measure, and the handler sends a trigger signal to the PNA when it is ready for a measurement. Additional signals are available on the PNA Handler I/O to indicate that the PNA sweep has ended, and that the handler can setup for the next measurement. [See a procedure.](#)
- Alternatively, an external source usually sets up for the next measurement much faster than the PNA. This is because the PNA acquires data, and moves both source and receivers for the next measurement. In this

case, the external source sends a 'Ready' signal. The PNA then begins the measurement and sends a trigger signal AFTER the measurement has been made. This indicates that the measurement is complete and that the source should setup for the next measurement. [See a procedure.](#) Beginning with A.07.22, the PNA can control an external source from within the firmware. [Learn more.](#)

## Capability Summary for each PNA Model

The following describes the capabilities and recommended method of triggering for each PNA model.

<b>PNA-X</b> The PNA-X has the most comprehensive, flexible, and easy to understand of all models in the PNA family. The following are two reciprocal pairs that can be used to accomplish efficient triggering.		
Signal Pair	Rear-Panel Connectors	Control Settings (click to learn more)
PNA Ready for Trigger (OUT)	<a href="#">MEAS TRIG RDY</a> and <a href="#">Handler I/O p21</a>	<a href="#">Meas Trigger TAB</a>
Trigger IN to PNA	<a href="#">MEAS TRIG IN</a> and <a href="#">Handler I/O p18</a>	<a href="#">Meas Trigger TAB</a>
Trigger OUT of PNA	<a href="#">AUX TRIG OUT (1&amp;2)</a>	<a href="#">AUX Trig TAB</a>
Ext Device Ready (IN to PNA)	<a href="#">AUX TRIG IN (1&amp;2)</a>	<a href="#">AUX Trig TAB</a>

<b>PNA-L models</b> The <b>I/O (TRIG IN)</b> and <b>I/O TRIG OUT</b> signal pair is the recommended signal pair to synchronize the PNA-L and external devices. Both signals result in triggering the other instrument; neither of these signals indicate a 'Ready' condition.		
Recommended Signal Pair	Rear-Panel Connectors	Control Settings
Trigger IN to PNA	<a href="#">BNC IN</a>	<a href="#">External TAB</a>
Trigger OUT of PNA	<a href="#">BNC OUT</a>	<a href="#">I/O Trig TAB</a>
Other Signals	Rear-Panel Connectors	Control Settings
PNA Ready for Trigger (OUT)	<a href="#">AUX I/O p18</a>	<a href="#">SCPI</a> and <a href="#">COM</a> Only
	<a href="#">Handler I/O p21</a> (some PNA-L models)	<a href="#">SCPI</a> and <a href="#">COM</a> Only

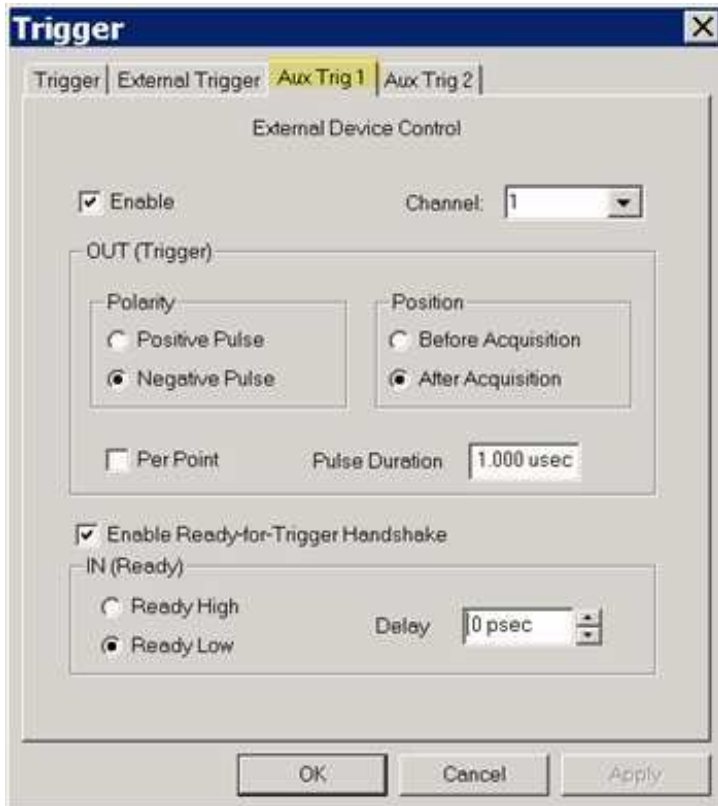
Trigger IN to PNA	<a href="#">AUX I/O p19</a>	<a href="#">SCPI</a> and <a href="#">COM</a> Only
	<a href="#">Handler I/O p18</a> (some PNA-L models)	<a href="#">SCPI</a> and <a href="#">COM</a> Only
Ext Device Ready (IN to PNA)	None	N/A

### E836xA/B/C

The **I/O (TRIG IN)** and **I/O TRIG OUT** signal pair is the recommended signal pair to synchronize the E836x and external devices. Both signals result in triggering the other instrument; neither of these signals indicate a 'Ready' condition.

Recommended Signal Pair	Rear-Panel Connectors	Control Settings
Trigger IN to PNA	<a href="#">BNC IN</a>	<a href="#">External TAB</a>
Trigger OUT of PNA	<a href="#">BNC OUT</a>	<a href="#">I/O Trig Out TAB</a>
Other Signals	Rear-Panel Connectors	Control Settings
PNA Ready for Trigger (OUT)	<a href="#">AUX I/O p18</a>	<a href="#">SCPI</a> and <a href="#">COM</a> Only
Trigger IN to PNA	<a href="#">AUX I/O p19</a>	<a href="#">SCPI</a> and <a href="#">COM</a> Only
Ext Device Ready (IN to PNA)	None	N/A

[See how to access the Trigger Dialog](#)



#### Aux Trig 1 - Aux Trig 2 dialog box help

This reciprocal pair of signals on **PNA-X models ONLY**, offers high flexibility, and robust synchronization with external devices.

- When enabled, the PNA-X waits indefinitely for a 'Ready IN' signal on the AUX IN connector from an external device.
- When received, the PNA is triggered from the usual trigger sources (Internal, External, or Manual).
- The trigger output signal on the AUX OUT connector can be sent BEFORE or AFTER data acquisition.
- Each channel can be configured differently.
- [Two pair of AUX TRIG connectors](#) allow two external devices to be controlled simultaneously.

#### See Also

- See how to use these connectors to [synchronize with External Sources](#).
- To use the opposite reciprocal pair, see [Meas Trig IN and Ready OUT](#) pair.
- [Pulse Triggering](#)

#### Dialog Settings

**Note:** The Aux Trig 1 and Aux Trig 2 tabs are identical.

**Enable** Check to use the Aux1 or Aux2 connectors to output signals to an external device.

**Channel** All settings on this dialog affect the specified channel ONLY.

### OUT (Trigger)

After receiving the Aux Trig IN 'Ready' signal, the trigger signal comes from any of the following [Trigger Sources](#):

- **Internal** - trigger occurs immediately.
- **Manual** - trigger occurs when the Trigger button is pressed.
- **External** - trigger occurs when [Meas Trig In](#) signal is received. This must be configured independently.

The following settings control the properties of the signals sent out the rear panel [AUX TRIG OUT \(1&2\) connectors](#):

#### Polarity

**Positive Pulse** Outgoing pulse is positive.

**Negative Pulse** Outgoing pulse is negative.

#### Position

**Before Acquisition** Pulse is sent immediately **before** data acquisition begins.

**After Acquisition** Pulse is sent immediately **after** data acquisition is complete.

**Per Point** Check to cause a trigger output to be sent for each data point. Clear to send a trigger output for each sweep. This setting controls the trigger **output** signal regardless of the channel [Point trigger](#) setting, which causes the PNA channel to trigger per point. For example, to trigger the PNA channel per point, and output a trigger signal per point, both this, and the channel setting must be checked ON.

**Pulse Duration** Specifies the duration of the positive or negative output trigger pulse.

### Ready for Trigger Handshake

When checked, the PNA waits indefinitely for the input line at the rear panel [AUX TRIG IN \(1&2\) connectors](#) to change to the specified level before acquiring data. This signal indicates that the external device is ready for PNA data acquisition. If the signal arrives before the PNA is ready to acquire data, it is latched (remembered).

When NOT checked, the PNA-X does not wait, but outputs trigger signals when the PNA-X is ready.

This signal does NOT trigger the PNA-X. The trigger signal is generated from any of the usual sources: Internal, Manual, or External.

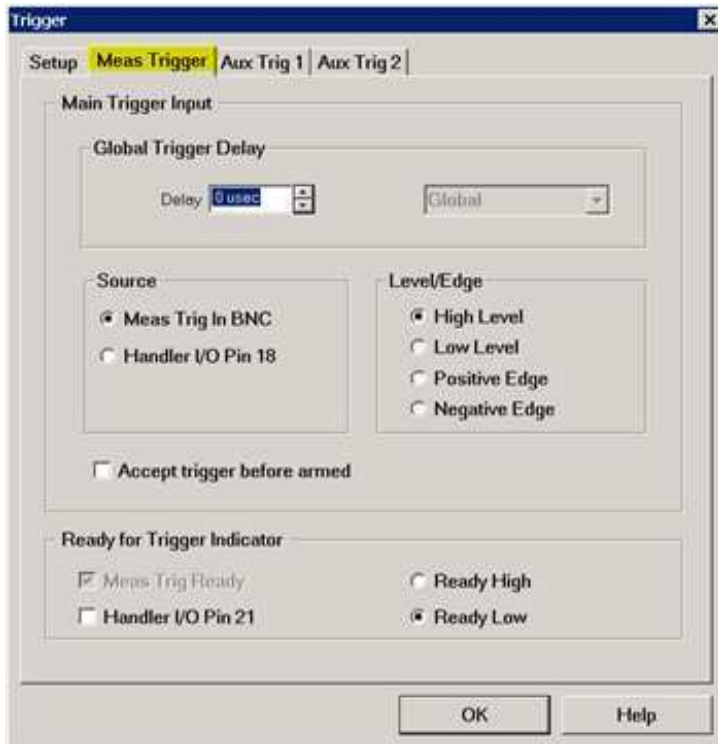
#### IN (READY)

**Ready High** PNA responds to the leading edge of a pulse on the Aux1 or Aux2 In connector.

**Ready Low** PNA responds to the trailing edge of a pulse on the Aux1 or Aux2 In connector.

**Delay** Time that the PNA waits after receiving the Handshake input before data acquisition.

[See how to access the Trigger Dialog](#)



Dialog box as it appears on a PNA-X. This tab is labeled **External** on PNA-L and E836x models

### Meas (External) Trigger dialog box help

#### Main Trigger Input

**Global / Channel Trigger Delay** After an external trigger is received, the start of the sweep is held off for this specified amount of time plus any inherent latency.

When [Trigger Scope](#) = Channel, the delay value is applied to the specified channel.

When Trigger Scope = Global, the same delay value is applied to ALL channels.

**Source** The PNA accepts Trigger IN signals through the following rear-panel connectors:

- [Meas Trig IN BNC](#) (PNA-X ONLY)
- [Handler I/O Pin 18](#) (PNA-L and PNA-X ONLY)
- [I/O 1 \(TRIG IN\) BNC](#) (PNA-L and E836x ONLY)
- [Aux I/O - pin 19](#) (PNA-L and E836x ONLY)

#### Level / Edge

**High Level** The PNA is triggered when it is armed (ready for trigger) and the TTL signal at the select input is HIGH.

**Low Level** The PNA is triggered when it is armed (ready for trigger) and the TTL signal at the select input is LOW.

**Positive Edge** After the PNA arms, it will trigger on the next positive edge. If [Accept Trigger Before Armed](#) is set, PNA will trigger as soon as it arms if a positive edge was received since the last data was taken.

**Negative Edge** After the PNA arms, it will trigger on the next negative edge. If [Accept Trigger Before Armed](#) is set, PNA will trigger as soon as it arms if a negative edge was received since the last data was taken.

**Note:** Edge triggering is NOT available on the following PNA models: E835xA, E880xA, N338xA, E8362A, E8363A, E8364A.

**Accept Trigger Before Armed** When checked, as the PNA becomes armed (ready to be triggered), the PNA will immediately trigger if any triggers were received since the last taking of data. The PNA remembers only one trigger signal. All others are ignored.

- When this checkbox is cleared, any trigger signal received before PNA is armed is ignored.
- This feature is only available when positive or negative EDGE triggering is selected.
- Configure this setting remotely using [CONTRol:SIGNal](#) (SCPI) or [ExternalTriggerConnectionBehavior](#) (COM).

### Ready for Trigger Indicator

On the PNA-X, when Meas Trig IN is enabled, then Meas Trig Ready (OUT) is also enabled.

Choose connector to send the PNA Ready OUT signal:

- [Meas Trig RDY](#)
- [Handler I/O p21](#)
- [AUX I/O p18](#)

Choose Polarity of the 'Ready OUT' signal.

This capability is not available on E836x models.

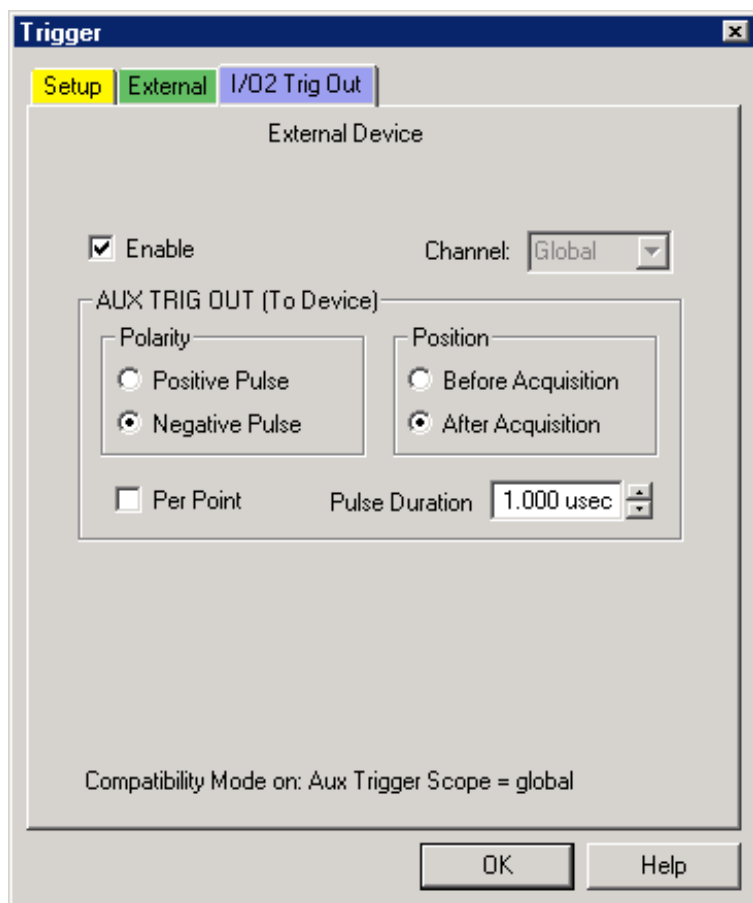
- **Ready High** - TTL High indicates the PNA is ready for trigger
- **Ready Low** - TTL Low indicates the PNA is ready for trigger (default setting).

### See Also

- [Pulse Trigger Tab](#)
- [Learn how to External Trigger during Calibration](#)

[See how to access the Trigger Dialog](#)





### I/O2 Trig Out dialog box help

**This TAB appears ONLY on E836X and PNA-L models with a Trig I/O rear-panel connector.**

**Enable** When checked, the PNA sends synchronized trigger signals out the rear-panel [I/O \(TRIG OUT\)](#) BNC connector.

#### Channel

**Global** - Trigger output properties apply for ALL channels. This is the default setting and is consistent with pre-7.2 release behavior. In this mode, the Per Point setting (below) is not allowed, but is coupled to the channel [Point trigger](#) property. In other words, when a channel is in point sweep mode, the trigger output will be sent per point.

**Channel** Trigger output properties are channel dependent. To output trigger signals for each point, check Per Point (see below).

**Note:** This Channel / Global setting can be changed ONLY by [setting a Preference](#).

The current setting is annotated at the bottom of the dialog as:

Compatibility Mode on: Aux Trigger Scope = global

#### AUX (I/O) TRIG OUT (To Device)

**Polarity** The trigger pulse output from the PNA is either in the Positive or Negative direction.

**Position** The trigger pulse output is sent either BEFORE or AFTER data is acquired.

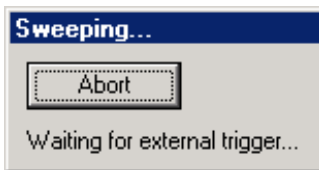
**Per Point** ([Channel mode](#) only) Check to cause a trigger output to be sent for each data point. Clear to send a trigger output for each sweep. This setting controls the trigger **output** signal regardless of the channel [Point trigger](#) setting, which causes the PNA channel to trigger per point. For example, to trigger the PNA channel per point, and output a trigger signal per point, both this, and the channel setting must be checked ON.

**Pulse Duration** Specifies the duration of the positive or negative output trigger pulse.

**Note:** Beginning with PNA Rev 6.0, Guided and Unguided Calibration CAN be performed in External Trigger mode. With this optional behavior, while Trigger Source is set to External, trigger signals must be sent for Calibration sweeps. This behavior does not apply to [FCA calibrations](#).

You can [set a Preference](#) to revert to pre-6.0 behavior - the PNA calibrates using Internal trigger signals while Trigger Source is set to External.

The following dialog box appears on the PNA screen while the PNA is waiting for an External trigger signal.

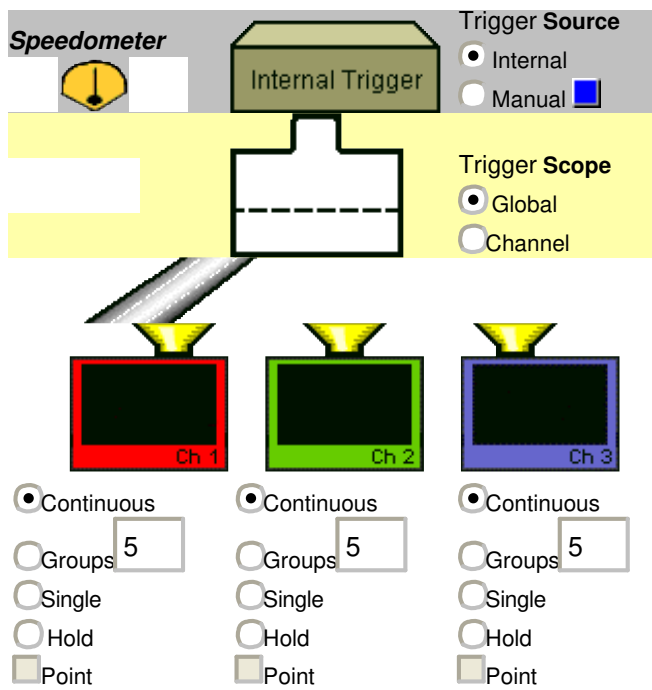


Click **Abort** to cancel the wait for a trigger signal.

---

Last Modified:

15-Mar-2010	Added Pulse Trig links
13-Aug-2009	Corrected Aux Trig IN link
22-Apr-2008	Added link to UI preference
14-Mar-2008	Added Ready Out polarity
3-Mar-2008	Many edits
25-Jan-2007	MX New topic



#### [About the trigger model](#)

Read [Text description](#) of triggering behaviors.

This model does not include the new [Sweep trigger mode](#).

## Data Format

---

A data format is the way the PNA presents measurement data graphically. Pick a data format appropriate to the information you want to learn about the test device.

- [How to set Format](#)
- [Rectangular \(Cartesian\) Display Formats](#)
- [Polar](#)
- [Smith Chart](#)

[See other 'Setup Measurements' topics](#)

### How to set the Display Format

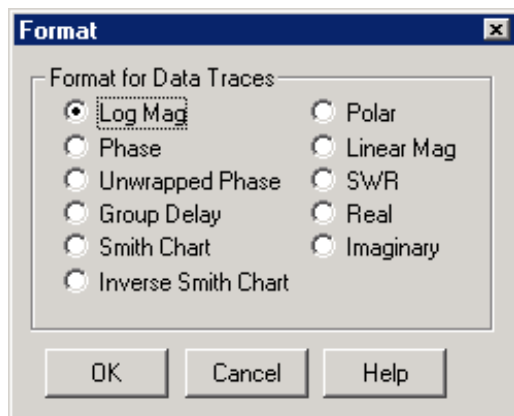
#### Using front-panel HARDKEY [softkey] buttons

1. Press **FORMAT**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Format**

◀ **Programming Commands** ▶



## Format dialog box help

- [Log Mag](#)
- [Phase / Unwrapped Phase](#)
- [Group Delay](#)
- [Smith / Inverse Smith Chart](#)
- [Polar](#)
- [Linear Mag](#)
- [SWR](#)
- [Real](#)
- [Imaginary](#)

## Rectangular Display Formats

Seven of the nine available data formats use a rectangular display to present measurement data. This display is also known as Cartesian, X/Y, or rectilinear. The rectangular display is especially useful for clearly displaying frequency response information of your test device.

- Stimulus data (frequency, power, or time) appears on the X-axis, scaled linearly
- Measured data appears on the Y-Axis.

## Log Mag (Logarithmic Magnitude) Format

- Displays Magnitude (no phase)
- Y-axis: dB
- Typical measurements:
  - Return Loss
  - Insertion Loss or Gain

## Phase Format

Measures the phase of a signal relative to the calibration reference plane with a range of +/- 180 degrees.

- Displays Phase (no magnitude)
- Y-axis: Phase (degrees)
- The trace 'wraps' every 180 degrees for easier scaling.
- Typical Measurements:

- [Deviation from Linear Phase](#)

## Unwrapped Phase

- Same as Phase, but without 180 degree wrapping.

**Note:** Phase is unwrapped by comparing the phase from one data point to the next. If the phase difference between two points is greater than 180 degrees, or if the phase of the first data point is greater than 180 degrees from DC, then the phase measurement is probably NOT accurate.

## Group Delay Format

- Displays signal transmission (propagation) time through a device
- Y-axis: Time (seconds)
- Typical Measurements:
  - Group Delay

### See Also:

[Group Delay](#) (Measurement)

[Comparing the PNA Delay Functions.](#)

[Phase Measurement Accuracy](#)

## Linear Magnitude Format

- Displays positive values only
- Y-axis: Unitless (**U**) for ratioed measurements  
Watts (**W**) for unratioed measurements.
- Typical Measurements:
  - reflection and transmission coefficients (magnitude)
  - time domain transfer

## SWR Format

- Displays reflection measurement data calculated from the formula  $(1 + \Gamma) / (1 - \Gamma)$  where  $\Gamma$  is reflection coefficient.
- Valid only for reflection measurements.
- Y axis: Unitless
- Typical Measurements:
  - SWR

## Real Format

- Displays only the real (resistive) portion of the measured complex data.
- Can show both positive and negative values.
- Y axis: Unitless
- Typical Measurements:
  - time domain
  - auxiliary input voltage signal for service purposes

### Imaginary Format

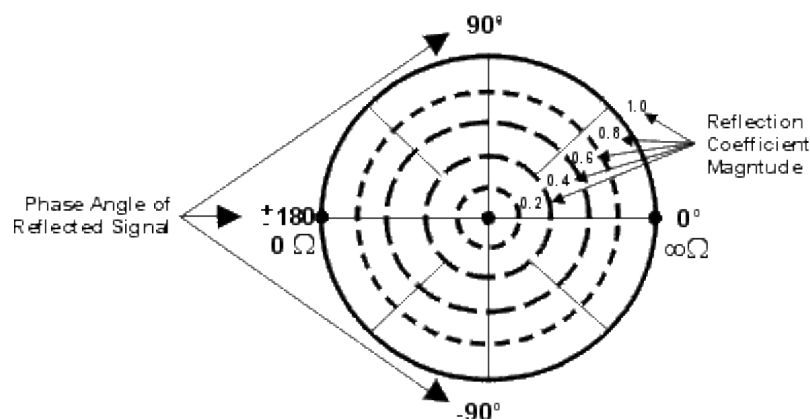
- Displays only the imaginary (reactive) portion of the measured data.
- Y - axis: Unitless
- Typical Measurements:
  - impedance for designing matching network

### Polar Format

Polar format is used to view the magnitude and **phase** of the reflection coefficient ( ) from your S<sub>11</sub> or S<sub>22</sub> measurement.

You can use Markers to display the following:

- Linear magnitude (in units) or log magnitude (in dB)
- Phase (in degrees)



- The dashed circles represent reflection coefficient. The outermost circle represents a reflection coefficient ( ) of 1, or total reflected signal. The center of the circle represents a reflection coefficient ( ) of 0, or no reflected signal.
- The radial lines show the phase angle of reflected signal. The right-most position corresponds to zero phase angle, (that is, the reflected signal is at the same phase as the incident signal). Phase differences of 90°,

$\pm 180^\circ$ , and  $-90^\circ$  correspond to the top, left-most, and bottom positions on the polar display, respectively.

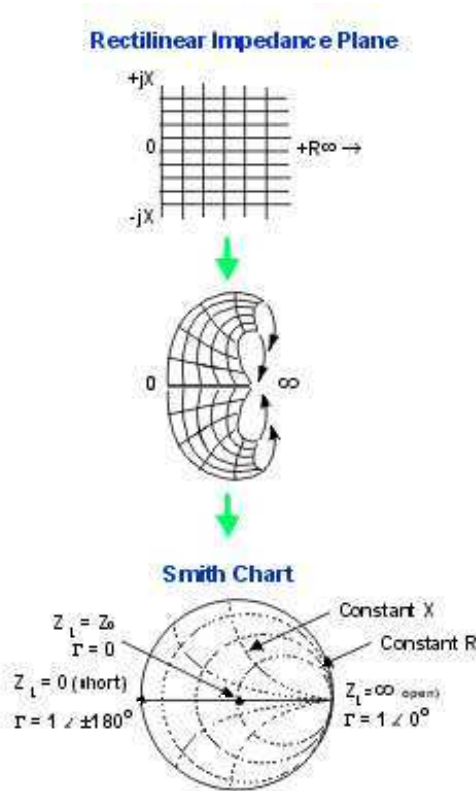
## Smith Chart Format

The Smith chart is a tool that maps the complex reflection coefficient (  $\Gamma$  ) to the test device's impedance.

In a Smith chart, the rectilinear impedance plane is reshaped to form a circular grid, from which the series resistance and reactance can be read ( $R + jX$ ).

You can use Markers to display the following:

- Resistance (in units of ohms)
- Reactance as an equivalent capacitance (in units of farads) or inductance (in units of henrys)



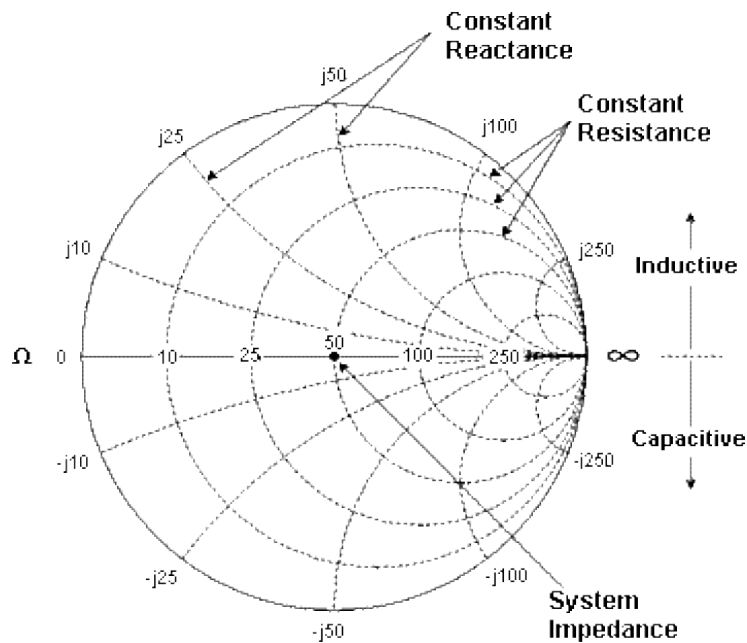
## Inverse Smith Chart (also known as Admittance)

Same as standard Smith Chart , except:

- The plot graticule is reversed right-to-left.
- Admittance (in units of siemens) instead of resistance.

## Interpreting the Smith Chart





- Every point on the Smith Chart represents a complex impedance made up of a real resistance (r) and an imaginary reactance ( $r+jX$ )
- The horizontal axis (the solid line) is the real portion of the impedance - the resistance. The center of the horizontal axis always represents the system impedance. To the far right, the value is infinite ohms (open). To the far left, the value is zero ohms (short)
- The dashed circles that intersect the horizontal axis represent constant resistance.
- The dashed arcs that are tangent to the horizontal axis represent constant reactance.
- The upper half of the Smith chart is the area where the reactive component is positive and therefore inductive.
- The lower half is the area where the reactive component is negative and therefore capacitive.

---

Last modified:

4-Jan-2010	Added link to group delay measurement
5-Aug-2009	Removed scale
6-Oct-2008	Added unwrapped phase note
3-Sep-2008	Removed legacy content
9/12/06	Added link to programming commands
9/27/06	MX Added UI



# Scale

The Scale, Reference Level and Reference Position settings (along with [Format](#)) determine how the data trace appears on the PNA screen.

- [Scale, Reference Level and Position](#)
- [Scale Coupling](#)
- [Magnitude Offset](#)

[See other 'Setup Measurements' topics](#)

## Scale, Reference Level and Position

The Scale, Reference Level and Reference Position settings (along with format) determine how the data trace appears on the PNA screen.

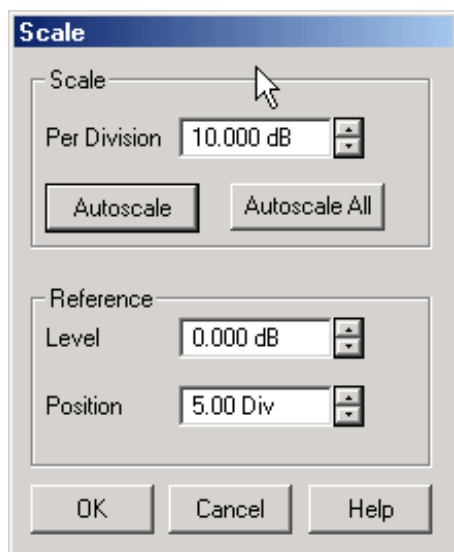
### How to set Scale, Reference Level, and Position

Using front-panel HARDKEY [softkey] buttons	PNA Menu using a mouse
<ol style="list-style-type: none"><li>1. Press <b>SCALE</b></li></ol>	<ol style="list-style-type: none"><li>1. Click <b>Response</b></li><li>2. then <b>Scale</b></li><li>3. then <b>Scale</b></li></ol>

◀

Programming Commands

▶



### Scale dialog box help

**Note:** Beginning with PNA Rev. 9.0, the scale settings are set to couple with other traces in each window. The following settings assume that Scale Coupling is set to OFF. [Learn more about Scale Coupling.](#)

#### Scale

**Per Division** Sets the value of the vertical divisions of a rectangular display format. In Polar and Smith Chart formats, scale sets the value of the outer circumference. Range: 0.001dB/div to 500 dB/div


**Autoscale** - Automatically sets value of the vertical divisions and reference value to fit the ACTIVE data trace within the grid area of the screen. The stimulus values and reference position are not affected.

The analyzer determines the smallest possible scale factor that will allow all the displayed data to fit onto 80 percent of the vertical grid.

The reference value is chosen to center the trace on the screen.

**Autoscale All** Automatically scales ALL data traces in the ACTIVE WINDOW to fit vertically within the grid area of the screen.

#### Reference

**Level** In rectangular formats, sets the value of the reference line, denoted by  on the PNA screen. Range: -500 dB to 500 dB.

In Polar and Smith chart formats, reference level is not applicable.

**Position** In rectangular formats, sets the position of the reference line. Zero is the bottom line of the screen and ten is the top line. Default position is five (middle).

In Polar and Smith chart formats, reference position is not applicable.

### Scale Coupling

With Scale Coupling enabled, traces that have the same format will have the same Scale, Reference Level, and Reference Position. You can choose to couple the scale of traces that are in the same window, couple the scale of all traces in all windows, or to have NO coupling.

## How to set Scale Coupling

1. Right-click on the Y-axis labels of a window
2. then select **Scale Coupling**

OR

### Using front-panel HARDKEY [softkey] buttons

1. Press **SCALE**
2. then **More**
3. then **Scale Coupling**

### PNA Menu using a mouse

1. Click **Response**
2. then **Scale**
3. then **Scale Coupling**

◀ Programming Commands ▶



## Scale Coupling dialog box help

Allows traces that share the same [format](#) to have the same [Scale](#), [Reference Level](#) and [Reference Position](#).

### Coupling Method

**Off** - No coupling. Traces are scaled individually. Default setting.

**Window** - All traces with the same format in each selected window share the same scale settings.

**All** - All traces in ALL selected windows with the same format share the same scale settings.

- When **Window** or **All** coupling is enabled, the scale settings for the active trace are assumed by other coupled traces with the same format.
- When there are traces with a different format present, all traces with that format assume the trace settings of the lowest-numbered trace of that format.
- Once enabled, scale settings for all coupled traces with the same format can be changed with any coupled trace being active.

### Selected Windows

Available when either the **Window** or **All** method is selected. Selected windows will participate in scale coupling. All windows are selected by default. Clear a checkbox to 'Opt-out' of scale coupling for that window.

### About Autoscale and Scale Coupling

**Autoscale** (not Autoscale All) affects the active trace in the active window. All traces that are coupled to this trace assume the new scale settings of the active trace. This could cause some traces to NOT show on the screen.

**Autoscale All** with Coupling Method...

- **Off** - All traces in the active window are autoscaled independently.
- **Window** - All traces in each selected window are autoscaled to fit within a common set of scaling factors.
- **All** - All traces in all selected windows are autoscaled to fit within a common set of scaling factors.

## Magnitude Offset

Magnitude Offset allows you to offset the magnitude (not phase) data by a fixed and / or sloped value in dB. If the display format is Linear Magnitude or Real (unitless), the conversion from dB is performed and the correct amount of offset is implemented.

## How to set Magnitude Offset

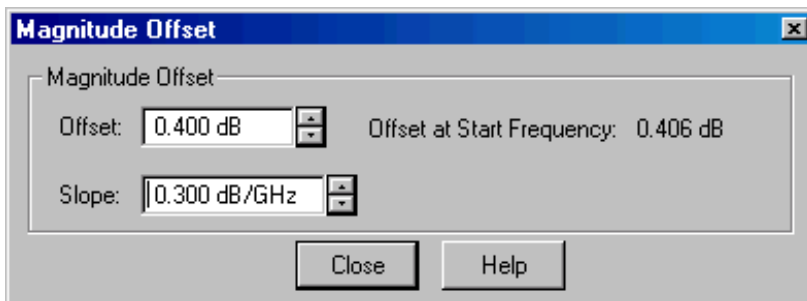
### Using front-panel HARDKEY [softkey] buttons

1. Press **SCALE**
2. then **[More]**
3. then **[Magnitude Offset]**

### PNA Menu using a mouse

1. Click **Response**
2. then **Scale**
3. then **Magnitude Offset**

## Programming Commands



### Magnitude Offset dialog box help

The Magnitude offset setting affects only the active trace.

**Offset** Offsets the entire data trace by the specified value.

**Slope** Offsets the data trace by a value that changes with frequency. The offset slope begins at 0 Hz.

For your convenience, the offset value at the start frequency is calculated and displayed.

See where this operation is performed in the [data processing chain](#).

Last Modified:

3-Jun-2010    MX New topic

## Pre-configured Measurement Setups

---

- [Pre-configured setups for NEW measurements](#)
- [Pre-configured arrangements for EXISTING measurements](#)

Before reading this topic, it is important to understand [Traces, Channels, and Windows](#) in the PNA.

[See other 'Setup Measurements' topics](#)

### Pre-configured Setups for NEW Measurements

Each of the following setups **creates new traces**. Existing traces and their settings will be lost, unless you first save them.

#### How to select a pre-configured measurement setup

##### Using front-panel HARDKEY [softkey] buttons

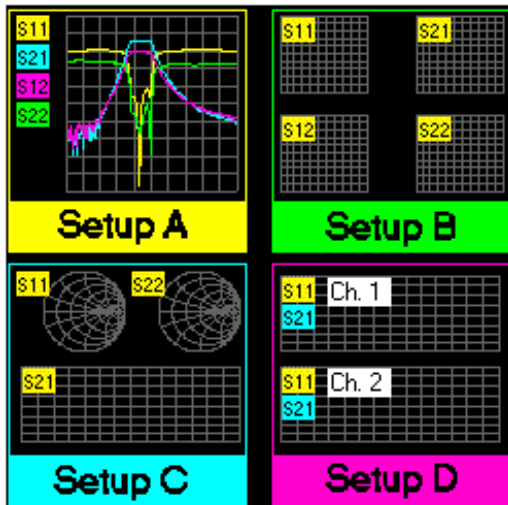
1. Press **RESPONSE**
2. then **[Display]**
3. then **[Meas Setups]**

##### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **Meas Setups**

No programming commands exist for this feature

The following are the four pre-configured measurement setups:





## Arranging Existing Measurements

The following arrangements place EXISTING measurements into pre-configured Window arrangements using a sort algorithm.

### How to select an Existing measurement arrangement

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**

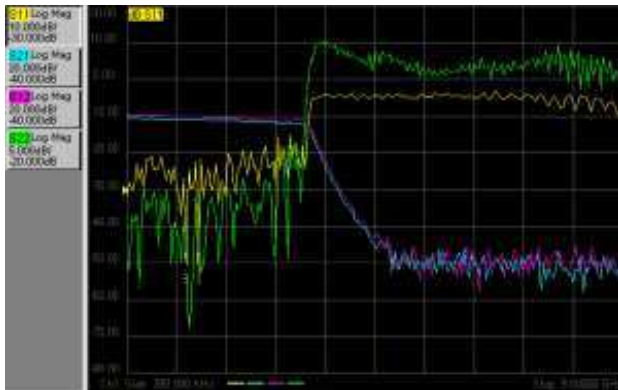
#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**

◀ Programming Commands ▶

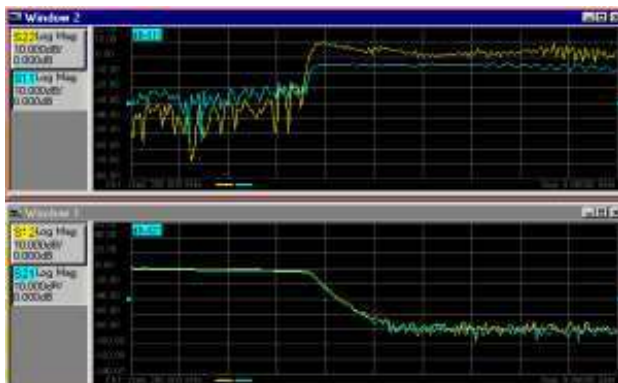
### Overlay Arrangement

This configuration places all existing traces in a single window, all overlaid on each other.



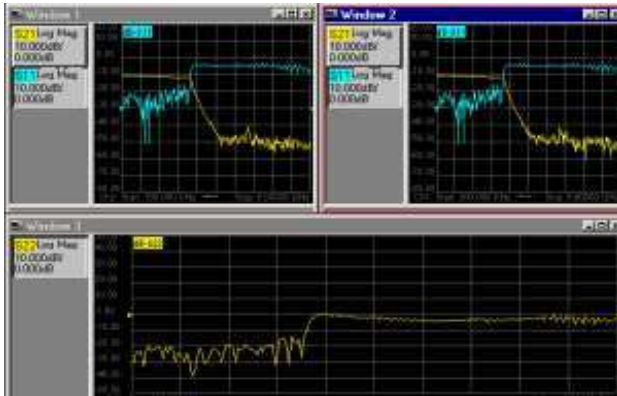
### Stack 2 Arrangement

This configuration places all existing traces in two "stacked" windows.



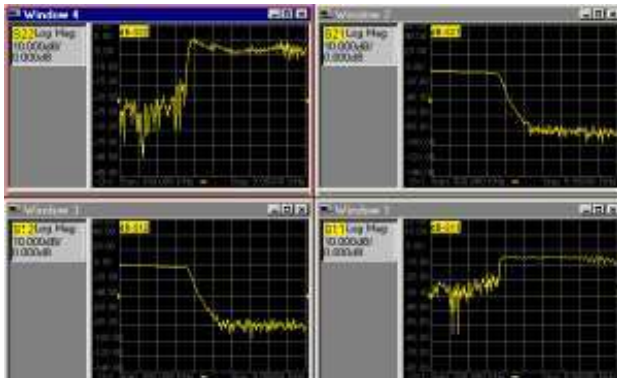
### Split 3 Arrangement

This configuration places all existing traces in three windows, two on top and one below.



## Quad 4 Arrangement

This configuration places all existing traces in four windows, one window in each screen quadrant.



## Sort Algorithm

The sort algorithm for the Arrange Windows feature is designed to:

- Divide traces among windows based on their properties
- Group traces with common properties

The algorithm sorting is based on the following trace properties, in order of priority:

1. Format: circular (polar or Smith) versus rectilinear (log mag, lin mag, group delay, etc.)
2. Channel number
3. Transmission versus reflection

**Note:** The PNA [traces per window limitation](#) overrides this algorithm. An error occurs if the arrange selection cannot be completed with the current number of traces on the screen.

Last modified:

3-Sep-2008	Removed legacy content
9/27/06	MX Added UI
9/12/06	Added link to programming commands

## Path Configurator

---

Allows you to configure hardware components that are available with selected [PNA-X options](#).

### How to access Path Configurator

#### Using HARDKEY [softkey] buttons:

1. Press **TRACE/CHAN**
2. then **[Channel]**
3. then **[More]**
4. then **[Hardware Setup]**
5. then **[Path Config...]**

#### PNA Menu using a mouse:

1. Click **Trace/Chan**
2. then **Channel**
3. then **Hardware Setup**
4. then **Path Config...**

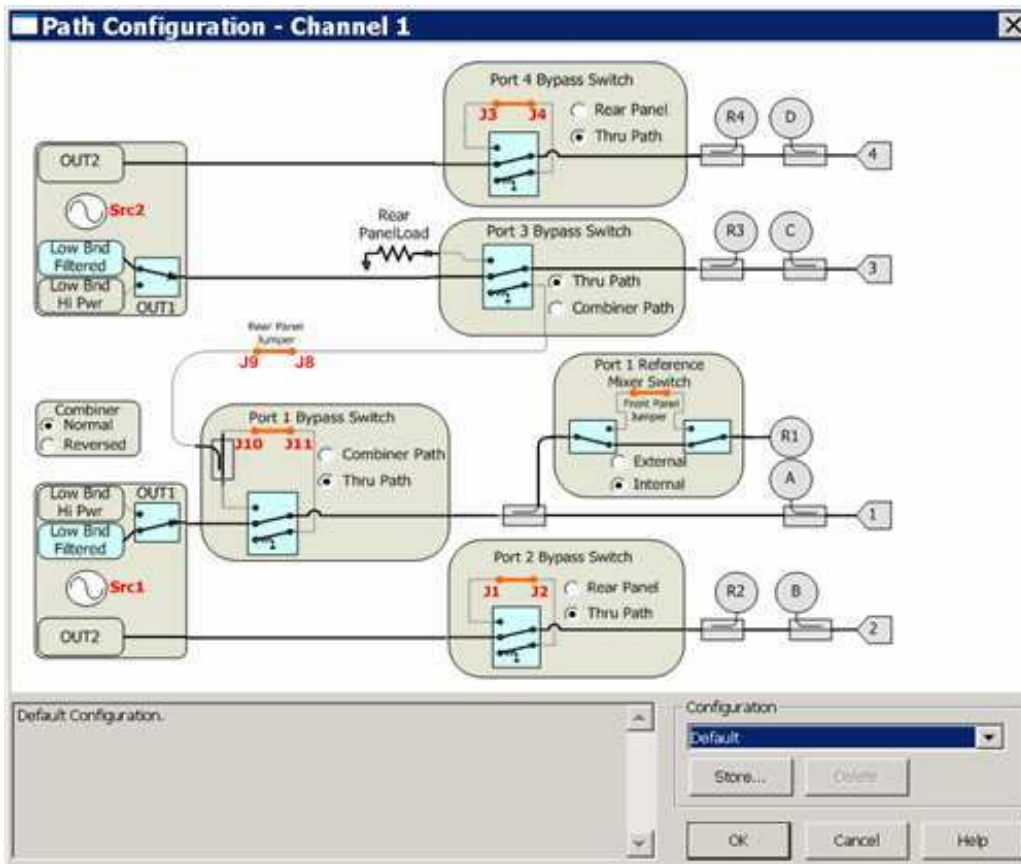
◀ **Programming Commands** ▶

The following image shows configuration with PNA-X Opt 423 (4-port, internal 2nd source, combiner, and mechanical switches). Your PNA-X may not include these options.

#### See Also

[IF Path configuration](#)

[PNA-X specs](#) for other RF path configuration diagrams.



### Path Configuration dialog box help

Different paths can be configured for each channel.

See [Noise Figure tab](#) help of the Path Configuration.

#### Configuration

Select, store, and delete factory configurations or user-defined configurations. Configurations are stored on the PNA hard drive.

Any configuration can be saved, and later recalled, from this dialog. Click **Store**, type a configuration name, then click **OK**.

**Text area** Displays text describing the physical connections required to complete the configuration. The text for factory configurations can NOT be edited. Text is saved as part of the configuration.

**Cancel** Closes the dialog and returns the configuration settings to the state they were in when the dialog was opened. Cancel does NOT undo **Store** and **Delete** actions that were performed while the dialog was open.

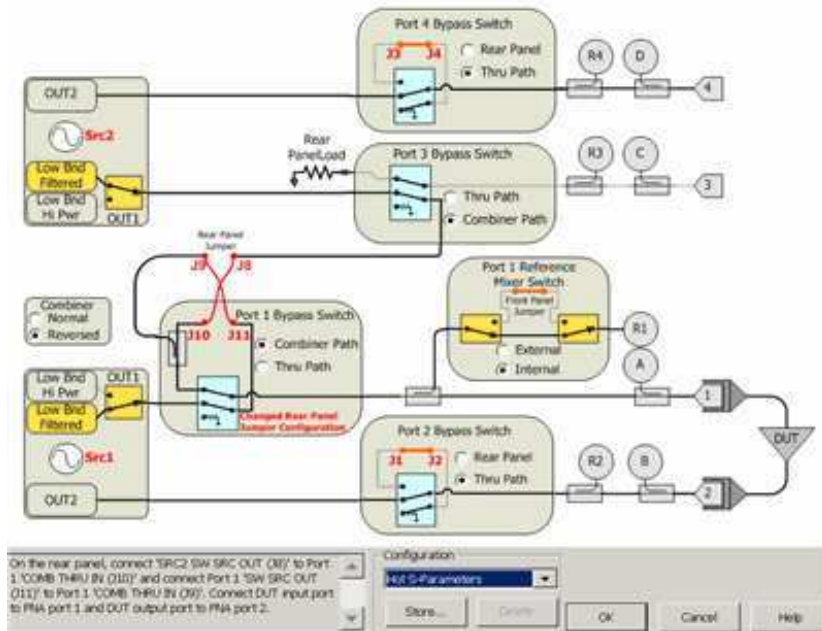
#### Factory Configurations

**Port 1 2-tone** Routes source 2 through the internal combiner to create a two-tone signal out port 1. The standard jumper configuration is used. This is the configuration that is used to make [IMD measurements](#).

**Hot S-parameters** The rear panel jumpers must be manually reconfigured to allow the source 2 signal to be routed through the thru path of the internal combiner. Source 2 is used to drive the AUT into compression, so the highest possible output power is required. Source 1, routed through the coupled path of the combiner, is used for S-parameter measurements, so a small signal is sufficient. These two signals are combined but the

frequencies are usually offset.

**2 port Dual Source** (Option 224 ONLY) Routes Source 2 out port 2 allowing pulsed measurements to be performed on both ports 1 and 2. This configuration requires an external cable to be attached.



## Hot S-Parameters configuration

### Notes

- Click or touch anywhere within a box on the screen to cycle through the available settings.
- Some switch settings alter graphics in areas other than where the switch is thrown.
- If you don't hear switches clicking, this could be why:
  - Electronic switches are **orange** on the path configuration dialog. These switches do not make noise when being thrown. Mechanical switches are **blue**.
  - The channel is not sweeping.

**Note:** To prevent premature wear, the PNA does not allow attenuators or other mechanical switches to switch continuously.

These mechanical devices are set for the entire channel. When more than one channel is used, and a mechanical device setting is NOT the same for all channels, only the ACTIVE channel is allowed to sweep. All other channels are NOT allowed to sweep (Blocked).

Learn how to [view the settings of all mechanical devices](#) in the PNA.

- Orange lines are jumpers on the front or rear panel.

- Notice on the block diagrams:
  - Extra filtering is available to optimize harmonics below 3.2 GHz on OUT1 of both sources. These filters are not used in the Hi Pwr setting.
  - Each source optionally has [pulse modulation](#) capability.
- [Copy channel](#) feature copies path configuration settings.
- Saved and recalled as part of an [instrument state](#).

#### See Also

- [Configuration for High-power measurements](#)

---

#### Last modified:

23-Feb-2010	Added Blocked channel note
26-May-2009	Added factory configs
3-Sep-2008	Removed legacy content
14-Aug-2008	Added link to high-power measurements
5-Jan-2007	MX- New topic

## Customize the PNA Screen

---

You can customize your PNA screen by showing or hiding the following display elements. All of these selections are made from the PNA **View** menu.

- [Status Bar](#)
- [Toolbars](#)
- [Tables](#)
- [Display Items](#)
- [Data and Memory Trace](#)
- [Title Bars](#)
- [Minimize Application](#)

See [expanded display capabilities of the PNA-X](#)

Learn about using [pre-configured measurements and windows arrangements](#)

Learn about [Traces, Channels, and Windows on the PNA](#)

[See other 'Setup Measurements' topics](#)

### Status Bar



When enabled, the status bar is displayed along the bottom of the PNA screen. The primary status bar shows the following:

- [Channel Trigger State](#) (Cont, Groups, Single, Hold)
- Active channel
- Measurement parameter for the active trace
- [Trace Math](#)
- [Error correction](#) for the active trace
- [Averaging Factor](#) for the active channel
- [Smoothing](#) Percentage
- [Transform](#) (On)
- [Gating](#) (On)
- IF Gating Enabled for [Pulsed App](#): (G)



- Manual IF Filtering for Pulsed App: (F)
- Delay if invoked using [Phase Offset](#), [Electrical Delay](#), or [Port Extensions](#).
- Loss if invoked using [Magnitude Offset](#) or [Port Extensions](#).
- GPIB status: Local (LCL), Remote Talker Listener (RMT), or System Controller (CTL).
- Error Status: (LVL, LCK, etc)

**Note:** A second level status bar appears when using [External Test Set Control](#) or [Interface control](#).

The status bar state (ON or OFF) will not change when the PNA is Preset.

### How to display the Status Bar

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[More]**
3. then **[Status Bar]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **Status Bar**

**Programming Commands**

## Toolbars

You can display up to five different toolbars to allow you to easily set up and modify measurements.

### How to display Toolbars

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[More]**
3. then **[Toolbars]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **More**
4. then **Toolbars**

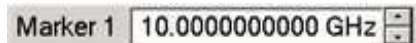
**Programming Commands**

**List of toolbars:**

- [Entry Toolbar](#)
- [Markers](#)
- [Measurement](#)
- [Sweep Control](#)
- [Stimulus](#)
- [Time Domain](#)
- [Port Extension](#)
- [All Off](#)

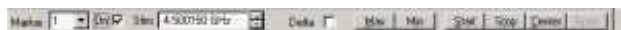
**Note:** There is also a Cal Set toolbar available for [Monitoring Error Terms](#).

### Entry Toolbar (For [PNA-X](#) and '[C](#)' models)



When used with softkeys, this area allows numeric values to be entered for PNA-X settings. From the keyboard, enter G for Giga, M for Mega or milli, K for kilo, and so forth.

### Markers Toolbar



The markers toolbar allows you to set up and modify markers. It shows:

- Marker number
- Stimulation value
- Marker functions:
  - Delta
  - Start/Stop
  - Center/Span

**Tip:** To use the Front Panel Knob to change marker position, first click the **Stimulus** field of the marker toolbar. Then turn the knob.

[Learn more about Markers](#)

### Measurement Toolbar



The measurement toolbar allows you to **create a new trace** for a desired S-parameter measurement in a current window or new window.

---

## Sweep Control Toolbar



In left to right order, the buttons on this toolbar set the active channel to:

- **Hold** mode
- **Single** sweep, then Hold mode
- **Continuous** sweep

Learn more about [Channel Trigger State](#).

---

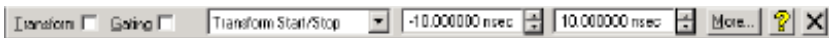
## Stimulus Toolbar



The stimulus toolbar allows you to view, set up, and modify the sweep stimulus. It shows the:

- **Start** value
  - **Stop** value
  - Number of **points**
- 

## Time Domain



The Time Domain toolbar allows you to do the following:

- Turn Transform and Gating ON / OFF
  - Change the Start / Stop times for both Transform and Gating
  - **More...** launches the [Time Domain Transform](#) dialog box
  - **X** Closes the toolbar
- 

The front panel **Tab** key steps through all of the settings on all of the toolbars on the display. If Tab does not work, press one of the Active Toolbar (color) keys.

---

## Port Extension



The Port Extension toolbar allows you to set Port Extensions while viewing the measurement trace. Learn more about [Port Extensions](#).

---

## All Off

This allows you to **hide all toolbars** with a single selection.

---

## Tables

Tables are displayed at the bottom of the selected window. Only one table may be displayed at a time for a window.

### How to display tables

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[More]**
3. then **[Tables]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **More**
4. then **Tables**

◀ **Programming Commands** ▶

#### List of tables:

- [Marker Table](#)
- [Limit Line Table](#)
- [Segment Table](#)

---

### Marker Table

You can display a table of marker settings. These settings include the:

- Marker number
- Marker reference (for delta measurements)
- Frequency
- Time and Distance (for Time Domain measurements)
- Response

Learn more about [Markers](#)

---

### Limit Line Table

You can display, set up, and modify a table of limit test settings. These include:

- Type (MIN, MAX, or OFF)
- Beginning and ending stimulus values
- Beginning and ending response values

Learn more about [Limit Lines](#)

---

## Segment Sweep Table

You can display, set up, and modify a table of segment sweep settings. These include:

- State (On/Off)
- Start and Stop frequencies
- Number of Points
- IF Bandwidth (if independent levels)
- Power Level (if independent levels)
- Sweep Time (if independent levels)

Learn more about [Segment sweep](#)

---

## Display Items

### How to show and hide Display items

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[Display Items]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **Display Items**


◀ **Programming Commands** ▶

### Display items:

- [Trace Status](#)
- [Frequency Stimulus](#)
- [Marker Readout](#)
- [Grid SOLID | dotted](#)

- [Limit Test Results](#)
- [Limit Lines](#)
- [Title](#)

## Trace Status

	<p><b>For <a href="#">PNA-X</a> and '<a href="#">C</a>' models</b></p> <p>Trace status is annotated at the top of each window.</p> <p>The highlighted trace number indicates the Active Trace.</p> <p>Click to select a trace.</p>
---	--

Trace Status shows the following:

- Trace number (Tr x). This is the trace number of the channel; NOT the window trace number which is used in many programming commands.
- Measurement parameter. This can be replaced with a custom [Trace Title](#).
- Format
- Scaling factor
- Reference level

## [How to show/hide Trace Status](#)

## Frequency/Stimulus

Ch1: Start 300.000 kHz ——— Stop 3.00000 GHz

Frequency/stimulus information is displayed at the bottom of each window on the screen. It shows:

- Channel number
- Start value
- Stop value

## [How to show/hide Frequency/Stimulus information](#)

## Marker / Bandwidth / Trace Statistics Readout

### [How to show/hide Readout settings](#)

The [Marker](#) / [Bandwidth](#) / [Trace Statistics](#) Readout area, in the upper-right corner of each window, can contain up to **20** total readout lines. However, all readout lines may not be visible depending on the window size and whether [Large Marker Readout](#) is enabled.

> 1:	13.255000 GHz	-15.928 dB
	Peak to Peak:	65.1137 dB
	Mean:	-2.07125 dB
	Std. Dev.:	8.33126 dB

The image shows 3 readout lines.

- Markers use one readout line per marker.
- Marker Bandwidth and Trace Statistics use three readout lines per trace.

### Marker Readout

- Checked - Shows readout information.
- Cleared - Shows **no** readout information.

### One Readout Per Trace

- Checked - Shows the readout of only the **active marker** for each trace.
- Cleared - Shows up to 20 total readouts lines.

### Large Marker Readout

This setting also controls Trace Statistics readout.

- Checked - Shows the marker readout in **large font size for easy reading**.
- Cleared - Shows the marker readout in **normal font size**.

Learn more about [Markers](#)

---

### Grid: SOLID | Dotted

Set whether to display ALL open window grid lines in solid or dotted lines. The selected setting is shown in CAPS. Once set, new windows are created using this setting. Grid lines return to SOLID when the PNA is Preset.

Set the color of the grid using [Display Colors](#).

[How to display grid settings](#)

---

### Limit Line Test Results

Limit line test results, **Pass** or **Fail**, are displayed on the right side of the designated window.

### Limit Lines

Limit lines are displayed for the active trace in the designated window. Their position depends on:

- Limit levels
- Format
- Scaling
- Reference level

Learn more about [Limit Lines](#)

[How to show/hide Limit Lines and Results](#)

---

## Title

You can create and display a title for each **window** using the keyboard. You can also use the following Title Entry dialog box.



The title is displayed in the upper-left corner of the selected window.

To clear a title, delete the title from the dialog box entry area and click OK.

See also [Trace Titles](#)

[How to show/hide a Title](#)

---

## Data Trace and Memory Trace

You can view or hide the active data or memory trace.

- Make a trace active by clicking the trace status button
- To view a memory trace you must first store a trace in memory. Click **Trace**, then **Math / Memory**, then **Data => Memory**.

Learn more about [Math operations](#)

---

## Title Bars



The Title bar shows the window number and Minimize / Maximize icons.

- Checked - Title bars for all PNA windows are shown.
- Cleared - Title bars for all PNA windows are hidden. This allows more room to display measurement results.



### How to show/hide the Title Bars

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[More]**
3. then **[Title Bars]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **Display**
3. then **More**
4. then **Title Bars**

◀ **Programming Commands** ▶

### Minimize Application

The Network Analyzer application can be minimized to show the desktop and Windows taskbar.

### How to minimize the Network Analyzer Application

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[Windows]**
3. then **[More]**
4. then **[Minimize]**

#### PNA Menu using a mouse

1. Click **File**
2. then **Minimize Application**

◀ **Programming Commands** ▶

To restore the PNA application, click the PNA application on the Windows taskbar.

Last modified:

16-Mar-2010	Added Grid lines (9.2)
3-Sep-2008	Removed legacy content
27-Aug-2007	Edited readout section
9/12/06	Added link to programming commands
9/27/06	MX Added UI

## Copy Channels

---

You can copy the channel settings from an existing channel to a new or another existing channel.

- [Why Copy Channels](#)
- [How to Copy Channels](#)
- [List of Channel Settings](#)

### Other Setup Measurements Topics

## Why Copy Channels

Copy channel settings if you need to create several channels that have slightly different settings.

For example, if you have an amplifier that you want to characterize over a frequency span with several different input power levels.

Follow these steps:

1. Create one measurement with your optimized channel settings.
2. Copy that channel to new channels.
3. Change the power level on the new channels.

The alternative to using Copy Channels is to create new default measurements on new channels. Then change every channel setting to your new requirement. This is very time consuming and thus shows the benefit of the Copy Channels feature.

**Note:** Copy Channels does NOT work with any of the PNA Applications, such as FCA, Gain Compression, or Noise Figure.

## How to Copy Channels

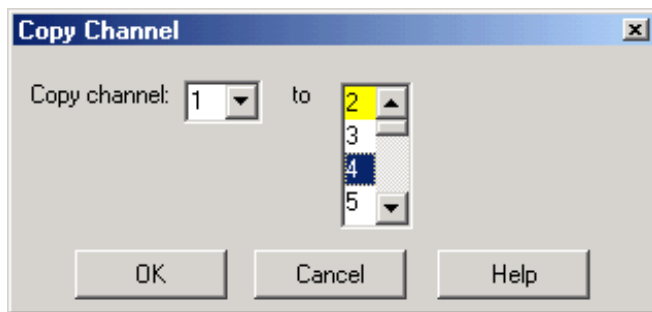
### Using front-panel HARDKEY [softkey] buttons

1. Press TRACE/CHAN
2. then [Channel]
3. then [More]
4. then [Copy Channel]

### PNA Menu using a mouse

1. Click **Trace/Chan**
2. then **Channel**
3. then **Copy Channel**

## Programming Commands



### Copy Channel dialog box help

Copies an existing channel's settings to another channel.

**Copy channel:** Select a channel to copy.

**to:** Scroll to select a channel to copy settings to. Channel numbers that are currently being used are highlighted. They can be selected and overwritten.

#### Notes:

- You can copy channel settings to **ONLY** one new or existing channel. Repeat this operation to copy to more than one channel.
- The new channel is **ALWAYS** copied to the Active window. If you want the new channel in its own window, first create a new measurement in a new window. Then make sure it is the Active window before you copy the channel into it.
- The measurement in the new channel becomes the active measurement.
- Only the channel settings are copied. The measurement trace is **NOT** copied to the new channel.
  - If measurements already exist on a channel being copied to, the measurements on that channel will not change, but they will assume the new channel settings.

- If a NEW channel is copied TO, an S11 measurement is created in order to view the channel settings.

For example:

1. **Existing** channel 1: S21 measurement
2. **Copy** channel 1 to NEW channel 2
3. **Result:** channel 2: S11 measurement with channel 1 copied settings. Both measurements are in the same window. The S11 measurement is the active measurement.

For more information see [Traces, Channels, and Windows on the PNA](#)

## List of Channel Settings

- [Frequency Span](#)
- [Power](#)
- [Cal Set usage](#)
- [Source Power Cal data](#)
- [IF Bandwidth](#)
- [Number of Points](#)
- [Sweep Settings](#)
- [Average](#)
- [Trigger \(some settings\)](#)

---

Last modified:

13-Feb-2008    Added note about Apps

9/12/06    Added link to programming commands

## ADC Measurements

The PNA is equipped with one or more ADC (Analog to Digital Converter) inputs. These ADC inputs can be used as measurement receivers and display measurements on the PNA screen.

- Analog Inputs can be used for measuring from -10V to +10V. These inputs can be considered auxiliary receivers and used in a similar way as S-Parameter receivers.
- Analog Output Sense inputs (AOS1 and AOS2) can be used to measure the corresponding DAC outputs.
- Analog Ground input (AIG) can be used to measure the instruments analog ground (PNA-X only).

### Supported Hardware

PNA-X: [Power I/O connector](#) (Pins 7 and 8)

Other models: [Aux I/O connector](#) (Pin 14)

#### How to create ADC receiver measurements

1. Press **TRACES**
2. then **[New Trace]**

1. Click **Trace/Chan**
2. then **New Trace**

◀ **Programming Commands** ▶



## New Trace (ADC) dialog box help

**Note:** Sweep speed slows dramatically when measuring more than two ADC receivers.

On the [New Trace dialog](#), click the **Receivers** tab.

**Activate** - check any empty line to create a trace.

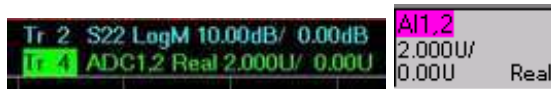
**Numerator** - select from the following:

- **AI1** - Input 1
- **AI2** - Input 2 (PNA-X only)
- **AOS1, AOS2** - Output sense 1 or 2
- **AIG** - Analog ground (PNA-X only)

**Denominator** - NOT available (ONLY unratioed measurements)

**Source Port** - The ADC receiver is measured when the specified source port is sweeping.

ADC receiver traces are labeled as shown in the following images:



- The ADC1 input is being measured, with 2 as the source port.
- The Y axis is U (unitless).
- The default trace [format](#) is Real (linear).

## ADC Traces and other useful PNA functions

Although most PNA functions work with ADC traces, the following may be especially useful.

- [Equation Editor](#) can be used with the trace data. Although the PNA-X ADC is measuring voltage (-10V to +10V range in 14 bits), by using a trace formula, this voltage can represent other types of measurement parameters (such as current, temperature, or a scaled voltage). [See PAE example.](#)
- [Trace averaging](#) and [Trace Smoothing](#) can be used to remove trace noise.
- [Dwell time](#) can be used to allow for settling.

## PNA Functions Not Supported

- Calibration for ADC receivers is NOT supported.
- Use with FCA is NOT supported.

- While the PNA is sweeping an ADC measurement, do NOT use the rear-panel [Analog I/O SCPI commands](#).

---

Last Modified:

31-Aug-2009	Fixed two typos
23-Mar-2009	Added pin numbers
19-Apr-2007	MX New topic

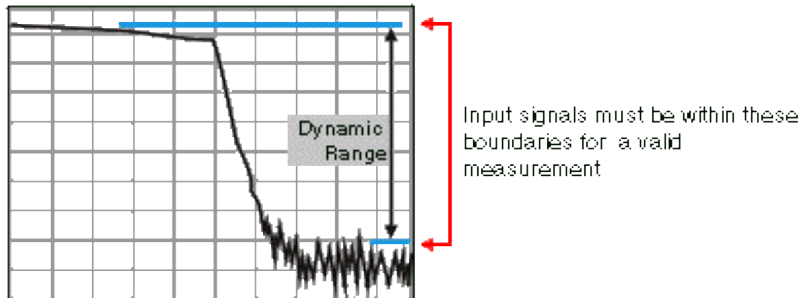


## Dynamic Range

---

Dynamic range is the difference between the analyzer receiver's maximum input power and the minimum measurable power (noise floor). For a measurement to be valid, input signals must be within these boundaries.

Increasing dynamic range is important if you need to measure very large variations in signal amplitude, such as filter bandpass and rejection. The dynamic range is shown below for an example measurement.



To help reduce measurement uncertainty, the analyzer dynamic range should be greater than the response that the DUT exhibits. For example, measurement accuracy is increased when the DUT response is at least 10 dB above the noise floor. The following methods can help you increase the dynamic range.

[Increase the Device Input Power](#)

[Reduce the Receiver Noise Floor](#)

[Improving Dynamic Range using the Front-Panel Jumpers](#)

### Other topics about Optimizing Measurements

## Increase Device Input Power

Increase the DUT input power so that the analyzer can more accurately detect and measure the DUT output power. However, use caution - too much power can damage the analyzer receiver or cause compression distortion.

**Caution! Receiver input damage level: +15 dBm.**

[See how to increase input power to the device](#)

**Tip:** You can further increase dynamic range by using an external booster amplifier to increase the input power to the DUT. See [High Power Amplifier Measurements](#).

## Reduce the Receiver Noise Floor

You can use the following techniques to lower the noise floor and increase the analyzer's dynamic range.

- Reduce crosstalk between the PNA receivers when measuring signals close to the noise floor. See [Receiver Crosstalk](#).)
- Use **Sweep Averaging** - learn more about [Sweep Average](#)

- Reduce the **IF Bandwidth** - learn more about [IF Bandwidth](#).
- In [Segment sweep](#) mode each segment can have its own IF bandwidth. For example, when measuring a filter:
  - In the passband, the IF bandwidth can be set wider for a fast sweep rate, as long as high-level trace noise is kept sufficiently small.
  - In the reject band, where noise floor contributes significantly to measurement error, the IF bandwidth can be set low enough to achieve the desired reduction in average noise level.

## Improving Dynamic Range using the Front-Panel Jumpers

**Note:** Bypassing a port's directional coupler increases the port mismatch by approximately 15 dB (the coupling factor of the directional coupler). [Learn more about using front-panel jumpers.](#)

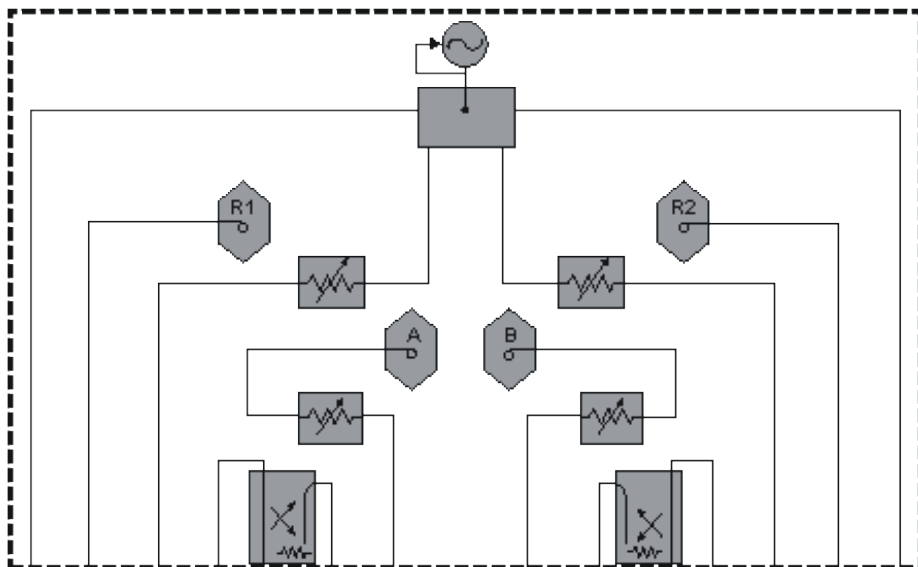
### For standard (non-PNA-X) models with Configurable Test Set Option

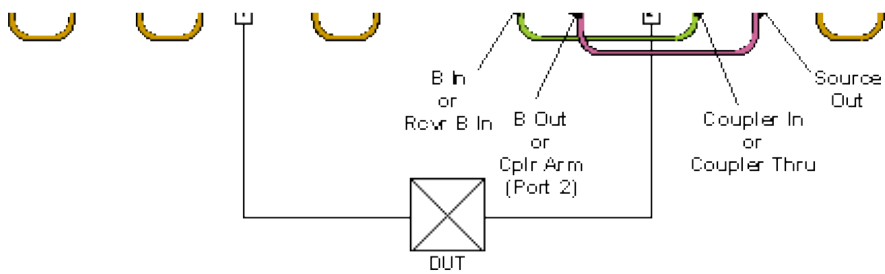
To improve dynamic range you can reverse the signal path in the test-port coupler and bypass the loss typically associated with the coupled arm. Using this configuration, you can increase the forward measurement dynamic range up to 143 dB with full 2-port calibration using segment sweep mode.

As shown in the following graphic, the signal is applied to Port 2. The signal bypasses the coupled arm via the jumper cable connected to the Coupler Thru (or Coupler In) and the Receiver B In (or B In) ports.

When making full 2-port error corrected measurements, the reverse measurement is degraded by 15 dB, with up to 113 dB of dynamic range available.

**Note:** Your analyzer's block diagram may contain different components than shown below. To see the block diagram for your PNA model, see the bottom of the [specs/data sheet](#).





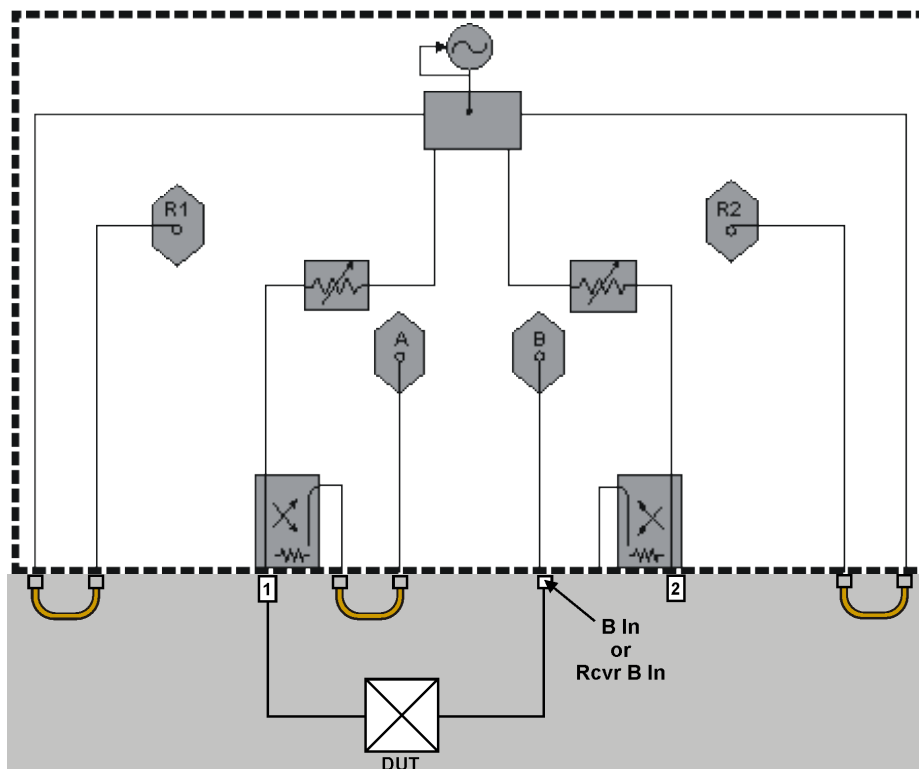
### For standard (non-PNA-X) models without Configurable Test Set Option

To improve dynamic range you can bypass the test-port coupler and apply the signal directly into the receiver. Using this configuration, you can achieve up to 143 dB dynamic range with **Response Calibration** using segment sweep mode.

As shown in the following graphic, the signal is applied to the front-panel connector for the B In or Rcvr B In front-panel jumper rather than Port 2.

Explore the graphic with your mouse.

**Note:** Your PNA may not be equipped with front-panel jumpers or all of the components shown in this block diagram.



Last Modified:

5-Aug-2008    Combined several topics

## Number of Points

---

A data point is a sample of data representing a measurement at a single stimulus value. You can specify the number of data points that the PNA measures across a sweep. (A "sweep" is a series of consecutive data point measurements, taken over a sequence of stimulus values.)

The PNA sweep time changes proportionally with the number of points. However, the overall measurement cycle time does not. See [Technical Specifications](#) for more information on how the number of points, and other settings, affect the sweep time.

**Note:** You may experience a significant decrease in computer processing speed with increased number of points, number of traces, and calibration error terms (full 2-port or 3-port). If this becomes a problem, you can increase the amount of RAM with PNA [Option 022](#).

### How to change the number of data points

Select a number or click Custom to invoke a [dialog box](#)

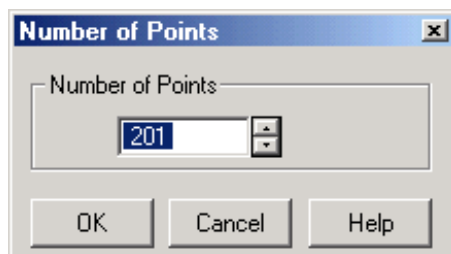
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SWEEP**
2. then **[Number of Points]**

#### Using a mouse with PNA Menus

1. Click **Stimulus**
2. then **Sweep**
3. then **Number of Points**

◀ **Programming Commands** ▶



## Number of Points dialog box help

Specifies the number of data points that the analyzer gathers during a measurement sweep. You can specify any number from **1** to **32,001**. The default value is 201.

**Note:** Some [measurement classes](#) may have different maximum points limitations.

Two data points are required for [Time Domain](#).

### Tips:

- To achieve the greatest trace resolution, use the maximum number of data points.
- For faster throughput use the smallest number of data points that will give you acceptable resolution.
- To find an optimized number of points, look for a value where there is not a significant difference in the measurement when you increase the number of points.
- To ensure an accurate measurement calibration, perform the calibration with the same number of points that will be used for the measurement.

Last modified:

2-Mar-2009	Increased max
3-Sep-2008	Removed legacy content
14-Dec-2007	Decreased min to 1
21-Jun-2007	MX Increased maximum
9/12/06	Added link to programming commands

## Phase Measurement Accuracy

---

You can increase the accuracy of phase measurements by using the following PNA features.

- [Electrical Delay](#)
- [Phase Offset](#)
- [Spacing Between Frequency Points \(Aliasing\)](#)

### See Also

[Port Extensions](#)

[Comparing the PNA Delay Functions.](#)

[Learn more about Phase measurements](#)

## Electrical Delay

Electrical delay is a mathematical function that simulates a variable length of lossless transmission line.

Use the electrical delay feature to compensate for the linear phase shift through a device. This feature allows you to look at only the [deviation from linear phase](#) of the device.

You can set the electrical delay independently for each measurement trace.

### How to set Electrical Delay

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SCALE**
2. then **[Electrical Delay]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Scale**
3. then **Electrical Delay**

[Programming Commands](#)

**Electrical Delay**

Electrical Delay

Time: 0 ps

Distance: 0 μm

Velocity Factor: 1.000000000

Softkey Display

☒ Time

☐ Distance

Distance Units

☒ Meter

☐ Feet

☐ Inches

Media

☒ Coax


☐ Waveguide

Cutoff Freq: 45000000 Hz

OK Cancel Help

### Electrical Delay dialog box help

**Electrical Delay** Specifies the value of delay added or removed, in Time or Distance. This compensates for the linear phase shift through a device. You can set the electrical delay independently for each measurement trace.

Click the Step  icon next to either Time or Distance to start the [Step Size dialog](#).

**Velocity Factor** Specifies the velocity factor that applies to the medium of the device that was inserted after the measurement calibration. The value for a polyethylene dielectric cable is 0.66 and 0.7 for Teflon dielectric. 1.0 corresponds to the speed of light in a vacuum.

Velocity factor can also be set from the [Port Extensions](#) dialog and [Time Domain Distance Marker Settings](#).

**Softkey Display** Allows you to enter delay in either Time or Distance using the softkeys and [Active Entry toolbar](#).

**Distance Units** Select from Meters, Inches, or Feet. The step size will not change automatically when this value is changed. [Learn more about Step Size](#).

#### Media

**Coax** Select if the added length is coax. Also specify the velocity factor of the coax.

**Waveguide** Select if the added length is waveguide. Also specify the low frequency cutoff of the waveguide.

**Cutoff Freq** Low frequency cutoff of the waveguide.

Learn about [Electrical Delay](#) (scroll up)

**Step Size**

☐ Auto

☒ User Defined: 5 μm

OK Cancel



### Step Size dialog box help

Changes the step size that occurs when the Time or Distance up/down arrows are pressed on the Electrical Delay dialog.

**Auto** Step Size is set to the default value.

**User Defined** Enter a step size value, then click OK.

This value remains the same when the units are changed. For example if a step size of 12 is entered on this dialog, then you change the units from Inches to Feet, the step size of 12 inches becomes 12 feet, not 1 foot. Therefore, change the units first, then set the step size.

## Phase Offset

Phase offset mathematically adjusts the phase measurement by a specified amount, up to 360°. Use this feature in the following ways:

- **Improve the display of a phase measurement.** This is similar to the way you would change the reference level in an amplitude measurement. Change the phase response to center or align the response on the screen.
- **Emulate a projected phase shift in your measurement.** For example, if you know that you need to add a cable and that the length of that cable will add a certain phase shift to your measurement, you can use phase offset to add that amount and simulate the complete device measurement.

### How to set Phase Offset

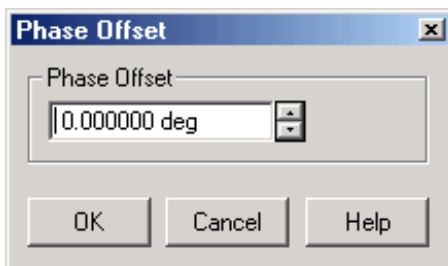
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SCALE**
2. then **[Phase Offset]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Scale**
3. then **Phase Offset**

◀ Programming Commands ▶



## Phase Offset dialog box help

**Phase Offset** Type a value or use the up and down arrows to select any value up to 360 degrees.

Learn about [Phase Offset](#) (scroll up)

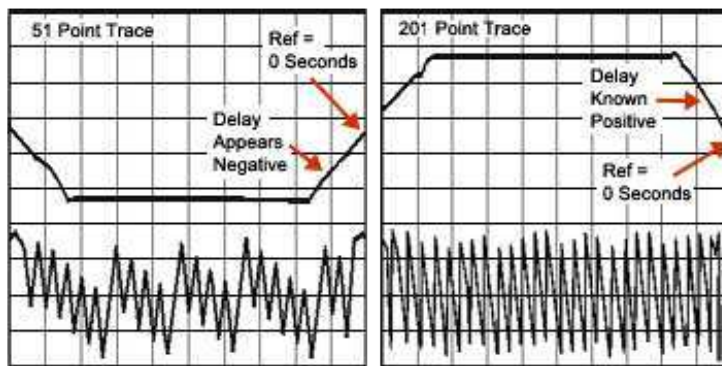
## Spacing Between Frequency Points (Aliasing)

The analyzer samples data at discrete frequency points, then connects the points, creating a trace on the screen.

If the phase shift through a device is  $>180^\circ$  between adjacent frequency points, the display can look like the phase slope is reversed. This is because the data is undersampled and aliasing is occurring.

If you are measuring group delay and the slope of the phase is reversed, then the group delay will change sign. For example, the following graphic shows a measurement of a SAW bandpass filter.

- The left measurement has 51 points and indicates the group delay is negative, which is a physical impossibility. That is, the response is below 0 seconds reference line.
- The right measurement shows an increase to 201 points which indicates the group delay is positive. That is, the response is above the 0 seconds reference line.



**Tip:** To check if aliasing might be occurring in a measurement, either [increase the number of points](#) or [reduce the frequency span](#).

Last modified:

12-Feb-2009 Added Step Size and Distance

3-Sep-2008 Removed legacy content

Sept.12, 2006 Added link to programming commands

## Electrically-Long Device Measurements

---

A signal coming out of a device under test may not be exactly the same frequency as the signal going in to a device at a given instant in time. This can sometimes lead to inaccurate measurement results. You can choose between two techniques to eliminate this situation and increase measurement accuracy.

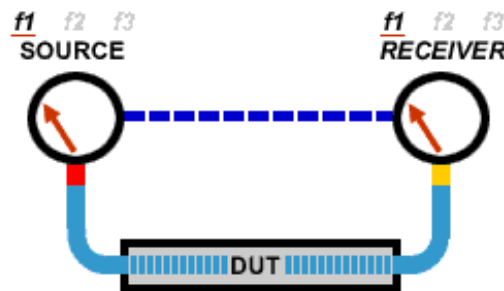
- [Why Device Delay May Create Inaccurate Results](#)
- [Solutions to Increase Measurement Accuracy](#)
  - [Slow the Sweep Speed](#)
  - [Add Electrical Length to the R Channel](#)

### Other topics about Optimizing Measurements

## Why Device Delay May Create Inaccurate Results

The following graphic shows an example of this situation:

- In the network analyzer, the source and receiver are **phase** locked together and sweep simultaneously through a span of frequencies.
- The signal flow through the Device Under Test (DUT) is shown as different colors for different frequencies.
- You can see as a stimulus frequency travels through the **DUT**, the analyzer tunes to a new frequency **just before** the signal arrives at the receiver. This causes inaccurate measurement results.



If the analyzer is measuring a long cable, the signal frequency at the end of the cable will lag behind the network analyzer source frequency. If the frequency shift is appreciable compared to the network analyzer's IF **detection** bandwidth (typically a few kHz), then the measured result will be in error by the rolloff of the IF filter.

**Note:** There is no fixed electrical length of a device where this becomes an issue. This is because there are many variables that lead to measurement speed. When high measurement accuracy is critical, lower the sweep speed until measurement results no longer change.

## Solutions to Increase Measurement Accuracy

Choose from the following methods to compensate for the time delay of an electrically long device.

### Slow the Sweep Speed

The following methods will slow the sweep speed.

- [Increase the Sweep Time](#)
- [Increase the Number of Points](#)
- [Use Stepped Sweep](#)
- [Set Dwell Time](#)

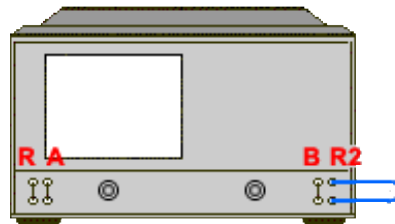
### Add Electrical Length to the R Channel

**Note:** This method applies to PNA models with front panel loops.

Instead of slowing the sweep, you can compensate for the electrical length of a cable or fixture.

- Remove the R-channel jumper on the front panel of the analyzer.
- Replace the jumper with a cable of about the same length as the device under test.
  - Add the cable on the R1 channel for  $S_{11}$  and  $S_{21}$  measurements.
  - Add the cable on the R2 channels for  $S_{22}$  and  $S_{12}$  measurements.
- Set the analyzer for a fast sweep.

### Configuration for $S_{22}$ and $S_{12}$ Measurements



This method balances the delays in the reference and test paths, so that the network analyzer's ratioed transmission measurement does not have a frequency-shift error.

**Note:** This method works well if the delay is in a cable or fixture. For devices with long delays, this method is only suitable for uncalibrated measurements.

## Reflection Accuracy on Low-Loss 2-Port Devices

---

To make accurate reflection measurements that have a 1-port calibration, you should terminate the unmeasured port.

- [Why Terminate the Unmeasured Port](#)
- [How to Terminate the Unmeasured Port](#)
- [Resulting Measurement Uncertainty](#)

### Other topics about Optimizing Measurements

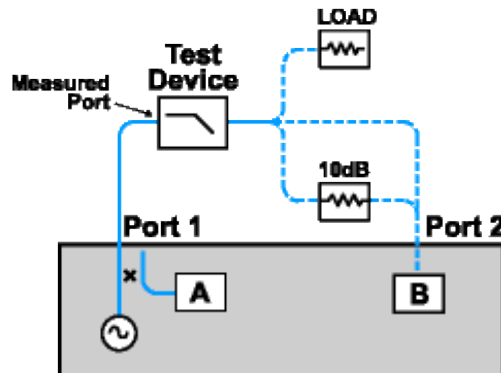
## Why Terminate the Unmeasured Port

A 2-port calibration corrects for all 12 twelve error terms. A 1-port calibration corrects for directivity, source match and frequency response, but not load match. Therefore, for highest accuracy, you must make the load match error as small as possible. This especially applies for low-loss, bi-directional devices such as filter passbands and cables. You do not need to be concerned with load match when you are measuring a device with high reverse isolation, such as an amplifier.

## How to Terminate the Unmeasured Port

Use one of the following methods:

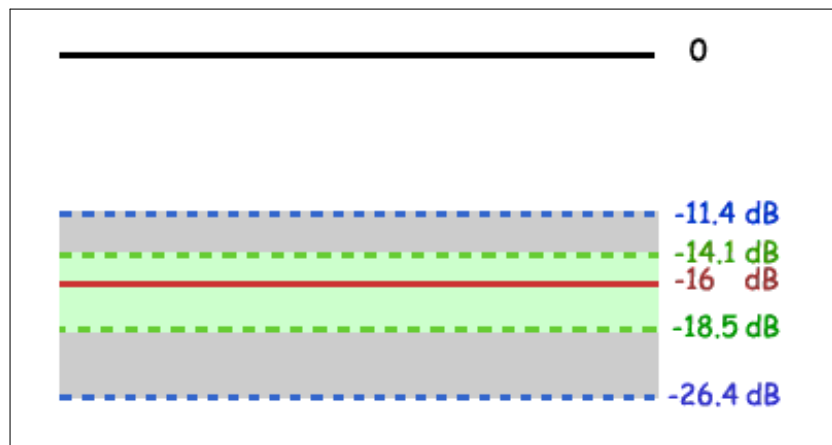
- Connect a high-quality termination load (from a calibration kit, for example) to the unmeasured port of your device. This technique yields measurement accuracy close to that of a Full SOLT 2-port calibration.
- Connect the unmeasured port of your device directly to the analyzer, inserting a 10 dB precision attenuator between the device output and the analyzer. This improves the effective load match of the analyzer by approximately twice the value of the attenuator, or 20 dB.



## Resulting Measurement Uncertainty

The following graph illustrates the measurement uncertainty that results from terminating **with** and **without** a

precision 10 dB attenuator on the output of the test device.



Legend

- Filter Reflection
- Uncertainty **with** attenuator
- ..... Uncertainty **without** attenuator

The calculations below show how adding a high-quality 10 dB attenuator improves the load match of the analyzer.

**Note:** The corresponding linear value is shown in parentheses.

**Network Analyzer:**

Load match (NALM) = 18 dB (.126)

Directivity (NAD) = 40 db (.010)

**Filter:**

Insertion loss (FIL) = 1dB (.891)

Return loss (FRL) = 16 dB (.158)

**Attenuator:**

Insertion loss (AIL) = 10 dB (.316)

SWR (ASWR) = 1.05 (.024)

32.26 dB Return Loss

---

**Calculations:**

**Without Attenuator**

$$\begin{aligned}\text{NA} &= (\text{FIL}) * (\text{NALM}) * (\text{FIL}) \\ &= (.891) * (.126) * (.891) \\ &= .100\end{aligned}$$

$$\text{Attenuator NA}$$

$$\begin{aligned}\text{Worst Case} &= \text{NA} \\ \text{Error (EWC)} &= .1\end{aligned}$$

**With Attenuator**

$$\begin{aligned}&= (\text{FIL}) * (\text{AIL}) * (\text{NALM}) * (\text{AIL}) * (\text{FIL}) \\ &= (.891) * (.316) * (.126) * (.316) * (.891) \\ &= .010\end{aligned}$$

$$\begin{aligned}&= (\text{FIL}) * (\text{ASWR}) * (\text{FIL}) \\ &= (.891) * (.024) * (.891) \\ &= .019\end{aligned}$$

$$\begin{aligned}&= \text{NA} + \text{Attn.} \\ &= .01 + .019 \\ &= .029\end{aligned}$$

---

$$\begin{aligned}\text{Uncertainty Adds} &= -20\log(\text{FRL}) + (\text{EWC}) + (\text{NAD}) \\ &= -20\log(.158) + (.100) + (.010) \\ &= \mathbf{11.4 \text{ dB}}\end{aligned}$$

$$\begin{aligned}&= -20\log(\text{FRL}) + (\text{EWC}) + (\text{NAD}) \\ &= -20\log(.158) + (.029) + (.010) \\ &= \mathbf{14.1 \text{ dB}}\end{aligned}$$

---

$$\begin{aligned}\text{Uncertainty Subtracts} &= -20\log(\text{FRL}) - (\text{EWC}) - (\text{NAD}) \\ &= -20\log(.158) - (.100) - (.010) \\ &= \mathbf{26.4 \text{ dB}}\end{aligned}$$

$$\begin{aligned}&= -20\log(\text{FRL}) - (\text{EWC}) - (\text{NAD}) \\ &= -20\log(.158) - (.029) - (.010) \\ &= \mathbf{18.5 \text{ dB}}\end{aligned}$$

---

Last Modified:

10-Mar-2010     Fixed Aswr- thanks Alex!

## Measurement Stability

---

There are several situations that can cause unstable measurements. To ensure that you are making repeatable measurements, you can use various methods to create a stable measurement environment.

- [Frequency Drift](#)
- [Temperature Drift](#)
- [Inaccurate Measurement Calibrations](#)
- [Device Connections](#)

### Other topics about Optimizing Measurements

## Frequency Drift

The analyzer frequency accuracy is based on an internal 10 MHz frequency oscillator. See [Technical Specifications](#) for stability and aging specifications.

If your measurement application requires better frequency accuracy and stability, you can override the internal frequency standard and provide your own high-stability external frequency source through the 10 MHz Reference [Input connector on the rear panel](#).

## Temperature Drift

Thermal expansion and contraction changes the electrical characteristics of the following components:

- Devices within the analyzer
- Calibration kit standards
- Test devices
- Cables
- Adapters

To reduce the effects of temperature drift on your measurements, do the following.

- Switch on the analyzer 1/2 hour before performing a measurement calibration or making a device measurement.
- One hour before you perform a measurement calibration, open the case of the calibration kit and take the standards out of the protective foam.
- Use a temperature-controlled environment. All specifications and characteristics apply over a 25 °C  $\pm$ 5 °C range (unless otherwise stated).
- Ensure the temperature stability of the calibration kit devices.



- Avoid handling the calibration kit devices unnecessarily during the calibration procedure.
- Ensure the ambient temperature is  $\pm 1^{\circ}\text{C}$  of the measurement calibration temperature.

## Inaccurate Measurement Calibrations

If a measurement calibration is inaccurate, you will not measure the true response of a device under test. To ensure that your calibration is accurate, you should consider the following practices:

- Perform a measurement calibration at the points where you connect the device under test, that is, the reference plane.
- If you insert any additional accessory (cable, adapter, attenuator) to the test setup after you have performed a measurement calibration, use the port extensions function to compensate for the added electrical length and delay.
- Use calibration standards that match the definitions used in the calibration process.
- Inspect, clean, and gage connectors. See [Connector Care](#).

See [Accurate Measurement Calibrations](#) for more detailed information.

## Device Connections

Good connections are necessary for repeatable measurements. To help make good connections, do the following:

- Inspect and clean the connectors for all of the components in the measurement setup.
- Use proper connection techniques.
- Avoid moving the cables during a measurement.

## Noise Reduction Techniques

---

Random electrical noise which shows up in the analyzer receiver chain can reduce measurement accuracy. The following PNA functions help reduce trace noise and the noise floor which can lead to better dynamic range and more accurate measurements.

**Note:** The trace noise in microwave PNAs becomes worse below 748 MHz and is especially obvious between 10 MHz and 45 MHz. See [Reduce IFBW](#).

- [Averaging](#)
- [IF Bandwidth](#)
- [Trace Smoothing](#)

### See Also

[Group Delay](#)

[Increase Dynamic Range](#)

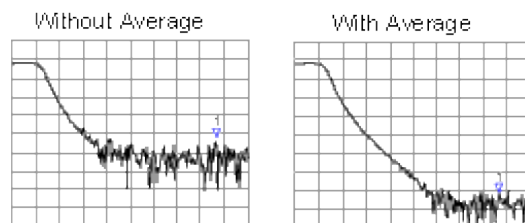
[PNA data processing map.](#)

### Other topics about Optimizing Measurements

## Averaging

Averaging is a feature that reduces the effects of random noise on a measurement. The PNA computes **each data point** based on the average of several measurements. You determine the number of measurements by setting the Average factor. The higher the average factor, the greater the amount of noise reduction.

### Effects of Sweep Average



- **Sweep averaging** - consecutive measurements occur over a number of sweeps.
- **Point averaging** - consecutive measurements occur immediately before moving to the next data point. Point averaging is usually much faster than sweep averaging. NOT available on ['C' models](#) and [Gain Compression](#) or [Noise Figure](#) Apps.
- An **Average Counter** appears on the screen when Sweep Averaging is selected, displaying the number of sweeps that has been averaged. The effect on the signal trace can be viewed as the Average Factor increases. This can assist in the selection of the optimum number of sweep averages.

- **Channel wide** - Averaging is applied to all measurements in a channel. The Average counter is displayed for each channel.
- **Unratioed** measurements - Although you can average unratioed (single receiver) measurements, you may get unexpected results:
  - Phase results may tend toward 0. This is because phase measurements are relative by nature. Measuring absolute phase with a single receiver appears random. Averaging random positive and negative numbers will tend toward 0.
  - The noise floor does not drop when averaging unratioed measurements as on ratioed measurements.
- **Average vs IF Bandwidth** - Both can be used for the same benefit of general noise reduction. For minimizing very low noise, using Average is more effective than reducing IF bandwidth. Generally, Averaging takes slightly longer than IF bandwidth reduction to lower noise, especially if many averages are required. Also, changing the IF bandwidth after calibration results in [uncertain accuracy](#).
- **Calibration** - Because averaging is a mathematical process that occurs after the raw measurement is made, averaging can be turned ON before, or after, calibration without invalidating the error correction terms. If averaging is ON before calibration, the measurement of calibration standards are averaged measurements. More time is needed to perform the calibration, but there will be less noise in the resulting error correction terms. Subsequent corrected measurements will also have less noise error. In addition, noise is further reduced by turning Averaging ON after calibration. [See the PNA data processing map](#).

## Averaging Formula

**NewAvg = (NewData/n) + [OldAvg\*(n-1/n)] 'where n = average factor**

From the formula, you can see that data from the first **n** sweeps continues to be included in the results of subsequent sweeps. Its effect is increasingly smaller but never diminishes to zero. For example, with  $n = 5$ , the average of the 5 sweeps is displayed. On the 6th sweep, you see 4/5 the average of the first 5 sweeps plus 1/5 the new sweep.

The effects of older data can be eliminated by restarting the average after **n** sweeps.

### How to Set Averaging

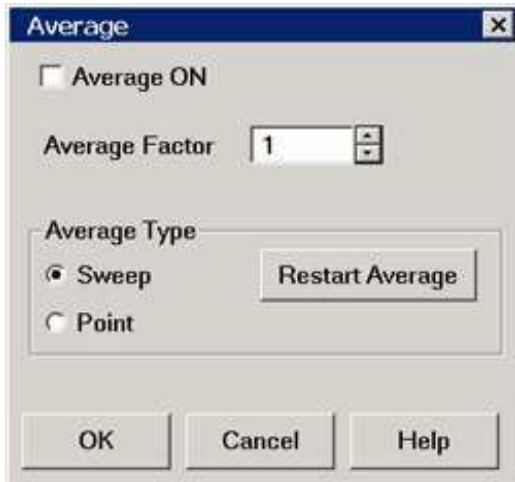
#### Using front-panel HARDKEY [softkey] buttons

1. Press **AVG**
2. then **[Averaging]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Avg**
3. then **Average**

**Programming Commands**



#### Average dialog box help

##### Average ON

Checked - Averaging is applied

Cleared - Averaging is NOT applied

**Average Factor** Specifies the number of measurements that are averaged. Range of 1 to 65536 ( $2^{16}$ ).

##### Average Type

**Sweep** Each data point is based on the average of the same data point measured over several consecutive sweeps.

**Point** Each data point is based on the average of the same data point measured repeatedly before going to the next data point. NOT available on ['C' models](#) and [Gain Compression](#) or [Noise Figure](#) Apps.

**Restart** Applies only to Sweep averaging - NOT Point. Begins a new set of measurements that are used for the average.

[Learn more about Averaging](#) (scroll up)

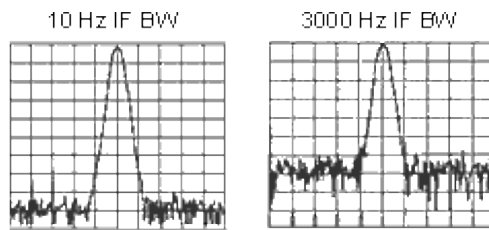
## IF Bandwidth

The PNA converts the received signal from its source to a lower intermediate frequency (IF). The bandwidth of the IF bandpass filter is adjustable from 40 kHz (for most PNA models) down to a minimum of 1 Hz.

Reducing the IF receiver bandwidth reduces the effect of random noise on a measurement. Each tenfold reduction in IF bandwidth lowers the noise floor by 10 dB. However, narrower IF bandwidths cause longer sweep times.

- **Channel wide** - IF bandwidth can be set independently for each channel
- **Segment sweep** - IF bandwidth can be set independently for each segment of segment sweep.
- **Calibration** - Changing the IF bandwidth after calibration will cause a ['C-delta' correction level](#), which means that calibration accuracy is uncertain.

### Effect of Reducing IF Bandwidth



## How to set IF Bandwidth

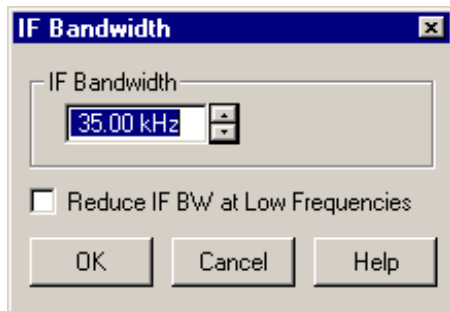
### Using front-panel HARDKEY [softkey] buttons

1. Press **AVG**
2. then **[IF Bandwidth]**

### Using a mouse with PNA Menus

1. Click **Response**
2. then **Avg**
3. then **IF Bandwidth**

## Programming Commands



## IF Bandwidth dialog box help

**IF Bandwidth** Specifies the IF (receiver) bandwidth. The value of IF bandwidth is selected by scrolling through the values available in the IF bandwidth text box. The IF BW is set independently for each channel.

The list of selectable IF Bandwidths is different depending on PNA model.

The following values are common to all models:

- 1 | 2 | 3 | 5 | 7 | 10 | 15 | 20 | 30 | 50 | 70 | 100 | 150 | 200 | 300 | 500 | 700 | 1k | 1.5k | 2k | 3k | 5k | 7k | 10k | 15k | 20k | 30k

In addition, the following values are PNA Model specific:

5230A Opts 020, 025, 120, 125, 140, 145, 146, 240, 245, 146

- 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k

N5230A Opts 220, 225, 420, 425, 520, 525:

- 50k | 70k | 100k | 150k | 200k | 250k

E836x models:

- 35k | 40k

PNA-X:

- 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k | 1M | 1.5M | 2M | 3M | 5M
- The following limitations apply for the highlighted IFBW settings (1 MHz and above).  
**Note:** These wider IFBW's are used to make [wideband pulsed](#) measurements. They do NOT provide faster sweep speeds for non-pulsed measurements.
  - [Dwell time](#) is not allowed.
  - [Sweep times](#) that are slower than the default value are not allowed.
  - [Step sweep](#) mode only - NOT available in Analog sweep.
  - [External Trigger Delay](#) is not allowed.
  - [Number of points](#) for CW sweep is limited to 1001.
  - A slight shift (.1dB) in Log Mag traces may be seen when switching in and out of these bandwidths.

### Reduce IF BW at Low Frequencies

On PNA models with a maximum frequency of 20 GHz and higher, the trace noise becomes worse below 748 MHz. This is especially obvious between 10 MHz and 45 MHz and also when Time Domain is ON. [See PNA models / maximum frequencies.](#)

When this box is checked, the PNA uses a smaller IF Bandwidth than the selected value at frequencies below 748 MHz.

This setting:

- can be made for each channel.
- is ON (checked) by default.
- also applies to [segment sweep](#).
- is NOT available on 4-port PNA-L (model N5230A Opt 240 and 245).

Use the following calculations to determine the actual IF Bandwidth value that is used below 748 MHz.

If the result is NOT a selectable IF BW value, the next higher selectable value is used.

	10 MHz to 44.999999 MHz	45 MHz to 748 MHz
<b>Less than 20 GHz models:</b>	Not applicable	Not applicable
<b>ALL 2-port 20 GHz PNA models :</b>	Actual IF BW = (selected IF BW) x (.05)	Actual IF BW = selected IF BW <b>(No reduction)</b>
<b>ALL 40 GHz and higher models:</b>	Actual IF BW = (selected IF BW) x (.025)	Actual IF BW = (selected IF BW) x (.5)

#### PNA-X Models

Start Freq	Stop Freq	Actual IF BW = (selected IF BW) x n
10MHz	19 MHz	n = .05
19MHz+	53 MHz	n = .1
53 MHz+	75 MHz	n = .5
75 MHz+	26.5 GHz	n = 1

+ indicates plus 1 Hz

#### Example:

On a 67 GHz PNA, the selected IF BW is 30 KHz.

With **Reduce IF BW at Low Frequencies** checked, the actual IF Bandwidths used are:

- From **10 MHz to 44.999999 MHz**:  $30,000\text{Hz} * .025 = 750\text{ Hz}$  (PNA uses next higher selectable value: **1000 Hz**.)
- From **45 MHz to 748 MHz**:  $30,000\text{Hz} * .5 = 15\text{ KHz}$
- From **748 MHz** to stop sweep: **30 KHz**

**OK** Selects the value of IF bandwidth shown in the text box.

[Learn about IF Bandwidth](#) (scroll up)

## Trace Smoothing

Trace smoothing averages a number of **adjacent** data points to smooth the displayed trace. The number of adjacent data points that get averaged together is also known as the smoothing aperture. You can specify aperture as either the number of data points or the percentage of the x-axis span.

Trace Smoothing reduces the peak-to-peak noise values on broadband measured data. It smooths trace noise and does not increase measurement time significantly.

Because Trace Smoothing follows Format in the PNA data processing map, the formatted data is smoothed.

Smoothing is automatically turned off if the format is Polar or Smith Chart.

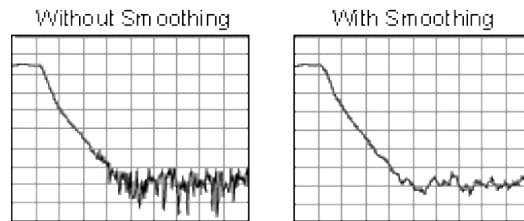
[Learn more about Data Format Types.](#)

[See the PNA data processing map.](#)

#### Tips:

- Start with a high number of display points and reduce until you are confident that the trace is not giving misleading results.
- Do not use smoothing for high-resonance devices, or devices with wide trace variations. It may introduce misleading information.
- Smoothing is set independently for each trace.

#### Effects of Smoothing on a Trace



#### How to set Trace Smoothing

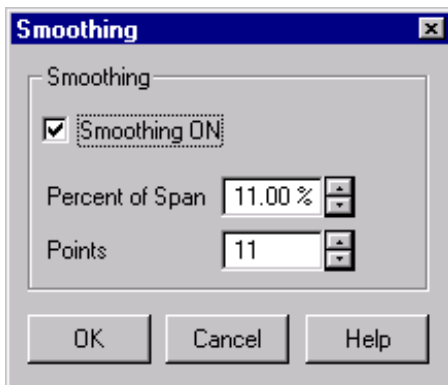
##### Using front-panel HARDKEY [softkey] buttons

1. Press **AVG**
2. then **[Smoothing]**

##### Using a mouse with PNA Menus

1. Click **Response**
2. then **Avg**
3. then **Smoothing**

#### Programming Commands





## Smoothing dialog box help

**Smoothing ON** When checked, applies smoothing to the displayed trace.

**Percent of Span** Specify percent of the swept stimulus span to smooth. For example, for a trace that contains 100 data points, and specify a percent of span = 11%, then the number of data points that are averaged is 11.

**Points** Specify the number of adjacent data points to average.

[Learn about Trace Smoothing](#) (scroll up)

Last modified:

22-Dec-2009	Added avg formula
10-Aug-2009	Point Avg Not available on GCA or NF Apps.
8-Jun-2009	Point averaging not available on C models.
6-Apr-2009	Replaced N5242A with PNA-X
12-Mar-2009	Edit IFBW for WB pulse
9-Oct-2008	Updated for Point Average (8.33)
3-Sep-2008	Removed legacy content

## Crosstalk

---

Crosstalk is energy leakage between analyzer signal paths. This can be a problem with high-loss transmission measurements. Although the [crosstalk specification](#) of the PNA is exceptional, you can reduce the effects of crosstalk by doing the following:

- [Set the Sweep to Alternate](#)
- [Perform an Isolation Calibration](#)

---

### [Other topics about Optimizing Measurements](#)

#### Set the Sweep to Alternate

This selection is longer available from the user interface. [Learn more.](#)

#### Perform an Isolation Calibration

For transmission measurements, a response and isolation measurement calibration helps reduce crosstalk because the analyzer measures and then subtracts the leakage signal during the measurement calibration. The calibration improves isolation so that it is limited only by the noise floor.

**Note:** Isolation is never performed on a Smart (Guided) Calibration. [Learn more.](#)

Generally, the isolation error falls below the noise floor. So when you are performing an isolation calibration you should use a noise reduction technique such as sweep averages or reducing the IF bandwidth.

---

Last Modified:

3-Mar-2010    Edited alternate sweep text

## Effects of Accessories

---

Accessories in a configuration may affect the results of a device measurement. You can choose between two analyzer features that reduce the effects of accessories.

- [Power Slope to Compensate for Cable Loss](#)
- [Gating to Selectively Remove Responses](#)

---

### [Other topics about Optimizing Measurements](#)

#### Power Slope to Compensate for Cable Loss

If you have a long cable or other accessory in a measurement configuration where a power loss occurs over frequency, apply the power slope function. This function increases the analyzer source power by a rate that you define (dB/GHz).

1. In the **Channel** menu, click **Power**.
2. If the slope function is not already switched on, click the **Slope** check box.
3. In the **dB/GHz** box, enter the rate that you want the source power to increase over the frequency sweep. Click **OK**.

#### Gating to Selectively Remove Responses

Gating is a feature in the time domain (option 010) that allows the analyzer to mathematically remove responses. You can set the gate for either a reflection or transmission response, but you will see different results.

- **Gating a reflection response** isolates a desired response (such as a filter's return loss), from unwanted responses (such as adapter reflections or connector mismatches).
- **Gating a transmission response** isolates a specific path in a multipath device that has long electrical lengths.

See [Time Domain Gating](#) for more information.

## Achieve Fastest Sweep

---

You can achieve the fastest measurement sweep by adjusting the following:

- [Sweep Settings](#)
- [Noise Reduction Settings](#)
- [Measurement Calibration Choice](#)
- [Unnecessary Functions](#)

### Other topics about Optimizing Measurements

## Sweep Settings

Consider changing each of the following settings as suggested.

- [Frequency Span](#) - Measure only the frequencies that are necessary for your device.
- [Segment Sweep](#) - Use segments to focus test data only where you need it.
- [Switch Off Stepped Sweep](#) - Use linear swept mode to minimize sweep time when possible.
- [Auto Sweep Time](#) - Use this default to sweep as quickly as possible for the current settings.
- [Number of Points](#) - Use the minimum number of points required for the measurement.

For more information on how number of points and other settings affect sweep cycle time, see [Technical Specifications](#).

## Noise Reduction Settings

Using a combination of these settings, you can decrease the sweep time while still achieving an acceptable measurement.

- [IF Bandwidth](#). Use the widest IF bandwidth that will produce acceptable trace noise and [dynamic range](#).
- [Average](#). Reduce the average factor, or switch Average off.

## Measurement Calibration Choice

Choose the appropriate type of calibration for the required level of accuracy.

When full 2-port error correction is applied, the PNA takes both forward and reverse sweeps to gather all 12 error correction terms. This occurs even with a single S11 measurement displayed. All displayed measurements are updated as the second sweep is performed. Both sweeps are performed using the specified sweep time.

When calibrating greater than 2 ports, the following formula is used to determine the number of sweeps required:

- $N * (N-1)$  where N = the number of ports.

When full 3-port calibration is applied, 6 sweeps are required; forward and reverse for each port pair. With full 4-port correction, 12 sweeps are required, and so forth.

To limit the measurement time, perform ONLY the level of calibration that your measurements require. For example, if making only an S11 measurement, perform a 1-port calibration on that port.

Sweep speed is about the same for uncorrected measurements and measurements done using a response calibration, or one-port calibration. For more information see [Select a Calibration](#).

## Unnecessary Functions

The analyzer must update information for all active functions. To achieve an additional increase in sweep speed, switch off all of the analyzer functions that are not necessary for your measurement application.

- [Delete Unwanted Traces](#)
- [Switch Off Unwanted Markers](#)
- [Switch Off Smoothing](#)
- [Switch Off Limit Testing](#)
- [Switch Off Math Functions](#)

Analyzer sweep speed is dependent on various measurement settings. Experiment with the settings to get the fastest sweep and the measurement results that you need.

## Switch Between Multiple Measurements

---

If you need to make multiple measurements to characterize a device, you can use various methods to increase throughput. Experiment with these methods to find what is best for your measurement application needs.

- [Set Up Measurements for Increased Throughput](#)
  - [Arrange Measurements in Sets](#)
  - [Use Segment Sweep](#)
  - [Trigger Measurements Selectively](#)
- [Automate Changes Between Measurements](#)
- [Recall Measurements Quickly](#)

### [Other topics about Optimizing Measurements](#)

## Set Up Measurements for Increased Throughput

To achieve optimum throughput of devices that require multiple measurements, it is helpful to know the operation of the PNA. This knowledge allows you to set up the measurement scenarios that are best for your applications.

[Learn more about Traces, Channels, and Windows on the PNA](#)

## Arrange Measurements in Sets

If you arrange measurements to keep the complete set of device measurements in one instrument state, you can save them so that you can later recall a number of measurements with one recall function.

See [Pre-configured Measurement Setups](#) for more information.

## Use Segment Sweep

Segment sweep is helpful if you need to change the following settings to characterize a device under test.

- Frequency Range
- Power Level
- IF Bandwidth
- Number of Points

The segment sweep allows you to define a set of frequency ranges that have independent attributes. This allows you to use one measurement sweep to measure a device that has varying characteristics.

See [Segment Sweep](#) for more information.

## Trigger Measurements Selectively

You can use the measurement trigger to make measurements as follows:

- Continuously update only the measurements that have rapidly changing data.
- Occasionally update measurements that have infrequently changing data.

For example, if you had four channels set up as follows:

- Two channels measuring the data that is used to tune a filter
- Two channels measuring the data for the out-of-band responses of the filter

You would want to constantly monitor only the measurement data that you use for tuning the filter. If you continuously update all of the channels, this could slow the response of the analyzer so that you would not be able to tune the filter as effectively.

**Note:** You must either trigger the infrequent measurement manually or with remote interface commands.

### To trigger measurements selectively:

This procedure shows you how to set up two different measurements with the following behavior:

- Channel 1 measurement will continuously update the data.
- Channel 2 measurement will occasionally update the data.

1. In the **Windows** menu, click **Meas Setups, Setup D**.

### Set Up a Measurement Trigger for Continuous Updates

2. In the **Sweep** menu, click **Trigger, Trigger....**
3. Under **Trigger Source**, click **Internal**.
4. Under **Channel Trigger State**, select **Channel 1**, and click **Continuous**.

### Set Up a Measurement Trigger for Occasional Updates

5. Under **Channel Trigger State**, select **Channel 2**, and click **Single, OK**.
  - If you want the analyzer to trigger more than a single sweep, click the **Enable Groups** check box and enter the number of sweeps.
6. In the **System** menu, click **Keys, Trigger**.

### Update the Measurement

7. Click on the lower window to make Channel 2 the active channel.

8. On the active entry toolbar, click the type of trigger you set up.
- Click **Single** if you set up the analyzer for a single sweep per trigger.
  - Click **Groups** if you set up the multiple sweeps per trigger.

**Note:** A trace must be active for you to initiate a trigger for that measurement.

## Automate Changes Between Measurements

If there are slight differences between the various measurements that you need to characterize a device, you may find that it is faster to change the measurement settings using programming.

## Recall Measurements Quickly

The most efficient way to recall measurements is to recall them as a set of measurements (instrument state).

- It only takes a short time longer to recall an instrument state that includes multiple measurements, than it does to recall an instrument state with only one measurement.
- Each recall function has time associated with it. You can eliminate that time by setting up the measurements as a set so you can recall them as a set.

See [Save and Recall Files](#) for more information.

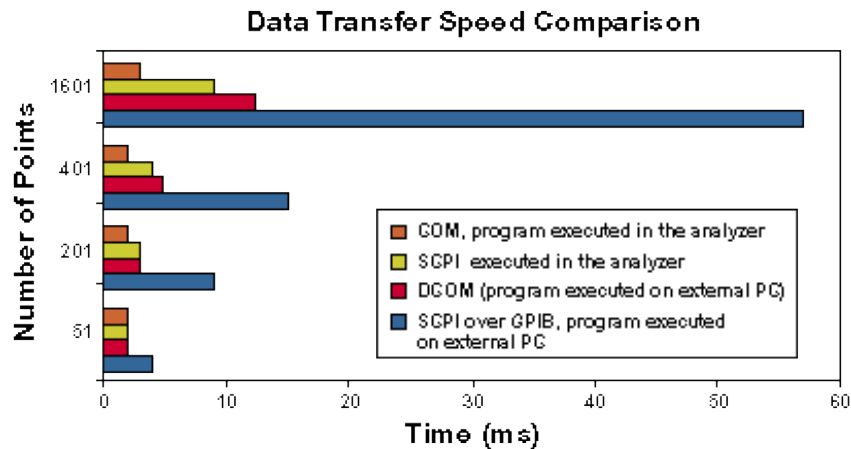


## Data Transfer Speed

When testing devices remotely using COM or SCPI, the following techniques can be used to transfer data quickly between the PNA and remote computer, helping you achieve the best measurement throughput.

- Use [single sweep \(trigger\) mode](#) to ensure that a measurement is complete before starting a data transfer.
- **Transfer the minimum amount of data** needed. For example, a trace with a few points, using segment sweep rather than a full trace with many linearly spaced points. Also, use markers instead of trace transfers.
- **Choose the REAL data format** to provide the fastest transfer speed when using SCPI programs for automated applications.
- **Use SCPI over LAN** for applications that are automated with SCPI programs.
- **Use COM programs** to provide the fastest transfer speed when using an automated application. See [Data Transfer Time](#) specifications.

**Note:** The following data is obsolete, but still serves to illustrate the relative speed between COM and SCPI.



[Other topics about Optimizing Measurements](#)

## Using Macros

---

Macros are executable programs that you write, load into the analyzer, and then run from the analyzer. You can have up to 25 macros set up to run on the analyzer.

[How to Setup Macros](#)

[How to Run Macros](#)

[Macro Example](#)

### How to Setup Macros

Using front-panel  
HARDKEY [softkey] buttons

Using a mouse with PNA Menus

For [PNA-X](#) and ['C' models](#)

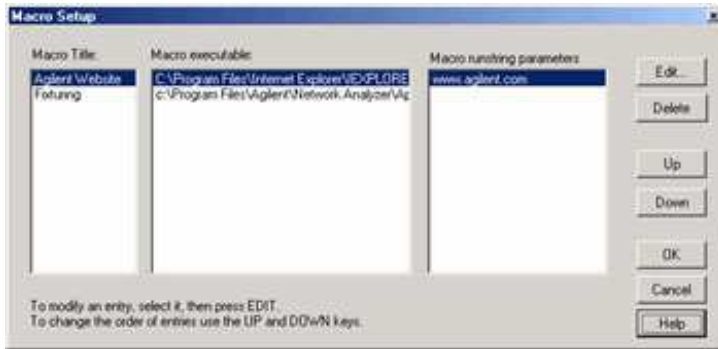
◀ Programming Commands ▶

1. Press **MACRO**
2. then **[Macro Setup]**

1. Click **Utility**
2. then **Macro**
3. then **Macro Setup**

#### In the Macro Setup dialog box:

1. Create an executable program and save it on the PNA hard drive. See [SCPI](#) or [COM](#) example programs in VBScript.
2. Use a mouse or the front-panel 'down-arrow' to select a blank line below the last entry. (There may be NO entry.)
3. Click **Edit** to start the [Edit Macro Setup](#) dialog.
4. In the **Macro Title** box, type a descriptive title for your macro.
5. Click **Browse**.
6. Change **Files of Type**.
7. Find and select your executable file. Change **Files of Type** if necessary.
8. Click **OK** on the Edit Macro Setup dialog.
9. Click **OK** on the Macro Setup dialog.
10. Press **MACRO** to run. It may be necessary to first Preset the PNA to see your macro in the menu.



### Macro Setup dialog box help

Allows you to create a set of up to 25 macros that can be launched from the PNA application.

An external keyboard is required to enter the Macro Title and the Run string parameters.

**To add a Macro**, use a mouse or the front-panel 'down arrow' (NOT the 'Down' key) to select a blank line. Then click **Edit**.

**Macro Title** Shows the titles that appear in the softkeys and menu when you press the Macro key. These titles are associated with the executable files and should be descriptive so you can easily identify them.

**Macro Executable** Lists the complete path to the executable file. To follow the example of launching the Agilent PNA Series Home Page, the path to the executable could be "C:/Program Files/Internet Explorer/iexplore.exe".

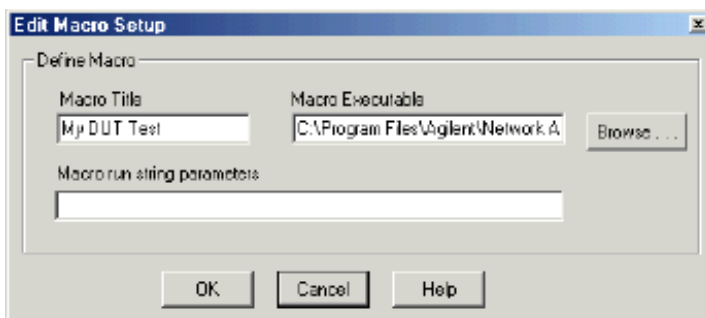
**Macro Runstring Parameters** Lists the parameters that get passed to the program that is referenced in the executable file. Again following the example of launching the PNA Series Home Page, you could assign the runstring parameters "http://www.agilent.com/find/pna".

**Edit** Invokes the [Macro Edit dialog box](#).

**Delete** Deletes the selected macro.

**Up** Allows you to reorder the macros, moving the selected macro up one line. This order determines how they appear in the PNA Menu and in the softkeys and when you press the Macro front-panel key.

**Down** Moves the selection down one line in the list of macros.



## Macro Edit dialog box help

**Macro Title** Allows you to modify the title that appears in the softkeys and menu.

**Macro Executable** Allows you to modify the complete path to the macro executable file.

**Browse** Allows you to look through drives and directories, to locate the macro executable file and establish the complete path to the file.

**Macro run string parameters** Allows you to modify the parameters that are passed to the program referenced in the executable file.

[See Macro Setup dialog box](#)

## How to Run Macros

Using front-panel  
HARDKEY [softkey] buttons

Using a mouse with PNA Menus

For [PNA-X](#) and ['C' models](#)

◀ Programming Commands ▶

1. Press **MACRO**
2. then select the macro to run

1. Click **Utility**
2. then **Macro**
3. then select the macro to run

## Macro Example

The following is an example Visual Basic Scripting (vbs) program that you can copy, install, and run on your PNA.

**Note:** Print these instructions if viewing in the analyzer. This topic will be covered by the Macro Setup dialog box.

1. Copy the following code into a Notepad file.
2. Save the file on the analyzer hard drive in the **C:/Documents** folder. Name the file **FilterTest.vbs**
3. Close Notepad
4. [Setup the macro in the PNA](#)
5. [Run the macro](#)

```
'Start copying here
'This program creates a S21 measurement, with Bandwidth
'markers for testing a 175MHz Bandpass filter
'It is written in VBscript using COM commands

Set PNA = CreateObject("AgilentPNA835x.Application")
PNA.Preset
```

```

Set chan=PNA.activechannel
Set meas=PNA.activemeasurement
Set limts = meas.LimitTest
Set trce = PNA.ActiveNAWindow.ActiveTrace

meas.ChangeParameter "S21",1
chan.StartFrequency = 45e6
chan.StopFrequency = 500e6
trce.ReferencePosition = 8
PNA.TriggerSignal = 3

'Do Test
for t=1 to 5
call measure
call compare
next
msgbox("Done Testing")

sub measure
msgbox("Connect Device " & t & " and press OK")
PNA.ManualTrigger True
meas.SearchFilterBandwidth
end sub

sub compare
BW = meas.FilterBW
if bw>6.5e7 then msgbox("Failed BW: " & BW)
Loss = meas.FilterLoss
if loss>5 then msgbox("Failed Loss: " & Loss)
end sub
'End copying here

```

---

Last Modified:

17-Feb-2009    Added 25 limit

4-Sep-2008    Removed legacy content

## Calibration Overview

---

The following is discussed in this topic:

- [What Is Measurement Calibration?](#)
- [Why Is Calibration Necessary?](#)
- [Conditions Where Calibration Is Suggested](#)
- [What Is ECal?](#)

[See other Calibration Topics](#)

### What Is Measurement Calibration?

Calibration removes one or more of the systematic errors using an equation called an error model. Measurement of high quality standards (for example, a short, open, load, and thru) allows the analyzer to solve for the error terms in the error model. See [Measurement Errors](#).

You can choose from different calibration types, depending on the measurement you are making and the level of accuracy you need for the measurement. See [Select a Calibration Type](#).

The accuracy of the calibrated measurements is dependent on the quality of the standards in the calibration kit and how accurately the standards are modeled (defined) in the calibration kit definition file. The calibration-kit definition file is stored in the analyzer. In order to make accurate measurements, the calibration-kit definition must match the actual calibration kit used. To learn more, see [Accurate Calibrations](#).

Calibration Wizard provides the different calibration methods used in the PNA. See [Calibration Wizard](#).

There are quick checks you can do to ensure your measurement calibration is accurate. To learn more see [Validity of a Measurement Calibration](#)

If you make your own custom-built calibration standards (for example, during in-fixture measurements), then you must characterize the calibration standards and enter the definitions into a user modified calibration-kit file. For more information on modifying calibration kit files, see [Calibration Standards](#).

**Note:** [Instrument Calibration](#) is ensuring the analyzer hardware is performing as specified. This is not the same as measurement calibration.

### Why Is Calibration Necessary?

It is impossible to make perfect hardware that would not need any form of **error correction**. Even making the hardware good enough to eliminate the need for error correction for most devices would be extremely expensive.

The accuracy of network analysis is greatly influenced by factors external to the network analyzer. Components of the measurement setup, such as interconnecting cables and adapters, introduce variations in magnitude and **phase** that can mask the actual response of the device under test.

The best balance is to make the hardware as good as practically possible, balancing performance and cost. Calibration is then a very useful tool to improve measurement accuracy.

## Conditions Where Calibration Is Suggested

Generally, you should calibrate for making a measurement under the following circumstances:

- You want the best accuracy possible.
- You are adapting to a different connector type or impedance.
- You are connecting a cable between the test device and an analyzer test port.
- You are measuring across a wide frequency span or an electrically long device.
- You are connecting an attenuator or other such device on the input or output of the test device.

If your test setup meets any of the conditions above, the following system characteristics may be affected:

- Amplitude at device input
- Frequency response accuracy
- Directivity
- Crosstalk (isolation)
- Source match
- Load match

## What Is ECal

ECal is a complete solid-state calibration solution. It makes one port (Reflection), full two and three-port calibrations fast and easy. See [Using ECal](#).

- It is less prone to operator error.
- The various standards (located inside the calibration module) never wear out because they are switched with PIN-diode or FET switches.
- The calibration modules are characterized using a TRL-calibrated network analyzer.
- ECal is not as accurate as a good TRL calibration.

For information about ordering ECal modules, see [Analyzer Accessories](#) or contact your [Agilent Support Representative](#)

## Calibration Standards

---

This following section explains the general principles and terms regarding calibration kit files. To learn **how** to modify calibration kit files, See [Modify Calibration Kits](#).

- [About Calibration Kits](#)
- [Calibration Standards](#)
- [Standard Type](#)
- [Standard Definitions](#)
- [Class Assignments](#)

[See other Calibration Topics](#)

### About Calibration Kits

A calibration kit is a set of physical devices called standards. Each standard has a precisely known or predictable magnitude and phase response as a function of frequency.

In order to calibrate the analyzer using the standards in a calibration kit, the response of each standard must be mathematically defined and then organized into standard classes that correspond to the error models used by the analyzer.

To be able to use a particular calibration kit, the known characteristics from each standard in the kit must be stored into analyzer memory. This is done for you with the PNA. All Agilent Cal Kits containing standard definitions are stored in the PNA. For a list of Agilent calibration kits, see [Analyzer Accessories](#).

### Calibration Standards

Calibration standards provide the reference for error-corrected measurements in the network analyzer. Each standard has a precisely known definition that includes electrical delay, impedance, and loss. The analyzer stores these definitions and uses them to calculate error correction terms.

During measurement calibration, the analyzer measures standards and mathematically compares the results with "ideal models" of those standards. The differences are separated into error terms that are later removed from device measurements during error correction. [See Systematic Errors](#).

### Standard Type

A standard type is one of five basic types that define the form or structure of the model to be used with that standard. The standard types are shown below:



Standard	Terminal Impedance
SHORT	zero ohms
OPEN	infinite ohms
LOAD	system impedance, Z0
THRU/LINE	no terminal impedance
<a href="#">ARBITRARY</a>	user-defined

#### Learn about other Calibration Standards:

- [Data-Based Standard](#)
- [Sliding Load](#)
- [Offset Load](#)

## Standard Definitions

Standard definitions describe the electrical characteristics of the standards and the frequencies they will be used. Standard definitions can be viewed from the [Advanced Modify Cal Kit](#) menu selection. Standard definitions include:

- **Minimum Frequency** Specifies the minimum frequency the standard is used for calibration.
- **Maximum Frequency** Specifies the maximum frequency the standard is used for calibration.
- **Z0** Specifies the characteristic impedance of the standard (not the system characteristic impedance or the terminal impedance of the standard).
- **Delay** Specifies a uniform length of transmission line between the standard being defined and the actual calibration plane.
- **Type** Specifies type of standard (SHORT, OPEN, THRU/LINE, LOAD, ARBITRARY).
- **Loss** Specifies energy loss, due to skin effect, along a one-way length of coaxial cable.

#### Loss model equation:

- The value of loss is entered as ohms/second at 1 GHz.
- To compute the loss of the standard, measure the delay in seconds and the loss in dB at 1 GHz. Then use the following formula:

$$\text{Loss} \left( \frac{\Omega}{s} \right) = \frac{\text{loss (dB)} \times Z_0(\Omega)}{4.3429(\text{dB}) \times \text{delay}(s)}$$

### Capacitance model equation:

**C0, C1, C2, C3.** Specifies the fringing capacitance for the open standard.

- $C = (C0) + (C1 \times F) + (C2 \times F^2) + (C3 \times F^3)$
- (F is the measurement frequency).
- The terms in the equation are defined when specifying the open as follows:
  - C0 term is the constant term of the third-order polynomial and is expressed in Farads.
  - C1 term is expressed in F/Hz (Farads/Hz).
  - C2 term is expressed in F/Hz<sup>2</sup>.
  - C3 term is expressed in F/Hz<sup>3</sup>.

### Inductance model equation:

**L0, L1, L2, L3.** Specifies the residual inductance for the short standard.

- $L = (L0) + (L1 \times F) + (L2 \times F^2) + (L3 \times F^3)$
- (F is the measurement frequency).
- The terms in the equation are defined when specifying the short as follows:
  - L0 term is the constant term of the third-order polynomial and is expressed in Henries.
  - L1 term is expressed in H/Hz (Henries/Hz)
  - L2 term is expressed in H/Hz<sup>2</sup>.
  - L3 term is expressed in H/Hz<sup>3</sup>.

## Class Assignments

Once a standard is characterized, it must be assigned to a standard "class". A standard class is a group of standards that are organized according to the calibration of the PNA error model.

The number of classes needed for a particular calibration type is equal to the number of error terms being corrected.

A class often consists of a single standard, but may be composed of multiple standards. These may be required for accuracy or to cover a wide frequency range.

**Example:** A response calibration requires only one class, and the standards for that class may include an OPEN, or SHORT, or THRU. A 1-port calibration requires three classes. A 2-port calibration requires 10 classes, not including two for isolation.

The number of standards assigned to a given class may vary from one to seven for unguided calibrations. Guided calibrations allow as many standards as needed.

Calibration Classes are assigned in the [Advanced Modify Cal Kit](#) menu selection.

### The different classes used in the PNA:

#### **S11A, S11B, S11C (S22A, S22B, S22C and so forth)**

These are the three classes for port 1-reflection calibrations (three classes also for S22 and S33). They are used in

the one-port calibrations and the full two-port calibration. They are required in removing the directivity, source match, and reflection tracking errors. Typically, these classes might consist of an open, a short and a load standard for each port.

#### **Transmission and Match (forward and reverse)**

These classes are used to perform a full two-port calibration. The transmission class relates primarily to the transmission tracking, while the match class refers to load match. For both of these classes, the typical standard is a thru or delay.

#### **Isolation**

The isolation classes are used to perform a full two-port and the TRL two-port calibrations. The isolation classes apply to the forward and reverse crosstalk terms in the PNA error model.

#### **TRL THRU**

These are used to perform a TRL two-port calibration. The TRL thru class should contain a thru standard or a short line. If it contains a non-zero length thru standard, then the calibration type is called LRL or LRM.

#### **TRL REFLECT**

This class is used to perform a TRL two-port calibration. The TRL reflect class should contain a standard with a high reflection coefficient, typically an open or short. The actual reflection coefficient need not be known, but its phase angle should be specified approximately correctly ( $\pm 90$  deg). The exact same reflection standard must be used on both ports in the TRL calibration process.

#### **TRL LINE or MATCH**

These are used to perform a TRL two-port calibration. The TRL line or match class should contain line standards, load standards, or both. If a line standard is used, its phase shift must differ from that of the TRL THRU standard by  $20^\circ$  to  $160^\circ$ . This limits the useable frequency range to about 8 to 1. Two or more line standards of different lengths may be specified to get broader frequency coverage. It is also common to include a load standard for covering low frequencies, where the line's length would be impractically long. When a load is used, the calibration type is called TRM or LRM.

**Note:** For more information, read [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(Application Note 1287-11\)](#)

## Calibration Wizard

---

The Calibration Wizard allows you to choose a Calibration method and then perform the calibration.

- [How to Start Calibration Wizard](#)
- [Guided Calibration: Mechanical Standards](#)
- [Unguided Calibration](#)
- [Saving a Calibration](#)

### Other Cal Topics

#### How to start Calibration Wizard

##### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Start Cal]**
3. then **[Cal Wizard]**

##### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal Wizard**

#### Programming Commands



## Calibration Wizard Begin dialog box help

Select the calibration method:

### SmartCal (Guided Calibration)

This method provides a step-by-step "wizard" interface. You describe the connectors on your DUT and the cal kits you will use; it walks you through the most accurate calibration possible.

**Note:** SmartCal DOES allow you to measure calibration standards in any order. However, you must click **Next** and **Back** without measuring standards until you get to the standard you want to measure.

- Supports ALL Cals **EXCEPT** simple open, short, and thru response Cals . See Also [TRL Calibration](#)
- Use a different Cal Kit (**including ECal**) for each port.

### Unguided Calibration

This method provides a familiar calibration interface, but with limited capability. You choose the type of cal to perform; it allows you the flexibility to measure the standards in any order.

- Supports all Cals **EXCEPT** full 3-port, full 4-port.
- TRL is NOT supported on multiport PNAs.
- Only one Cal Kit can be used.
- Can NOT use Offset Load standard.

### Use Electronic Calibration

- This method provides fast, software-controlled calibrations.
- Only one ECal module can be used. Use SmartCal when more than one ECal module is needed.

### **Save Preferences**

- When cleared, you will continue to see this page on subsequent calibrations.
- When checked, saves your calibration method choice and the dialog no longer appears.
- To make this dialog re-appear, click **Calibration**, then **Preferences**.
- Learn more about [Calibration Preferences](#).

### The Calibration Window / Channel

During a Guided Calibration, a 'Cal Window' is created for you to view the connection of calibration standards before standards are measured. This Cal Window uses a new Cal channel that is created and duplicates the settings in the channel being calibrated. [Correction is ALWAYS OFF](#) for the displayed calibration channel. At the completion of the calibration, the calibration channel and window are deleted.

With PNA Rev. 7.50.27, the measurement of calibration standards can be performed while viewing any PNA window configuration you choose. The Cal Window is appended to your Custom Cal Window setting, and all windows are visible and sweeping below the Cal Wizard before the Measure (cal standard) button is pressed.

The windows to be viewed and channels to be swept during the cal process are specified using [SCPI commands](#). [See an example](#).

The new Cal Window settings do not work in a FCA channel.

### SmartCal (Guided Calibration)

Guided Calibration automatically determines the calibration type and suggests a calibration kit that matches your DUT connectors.

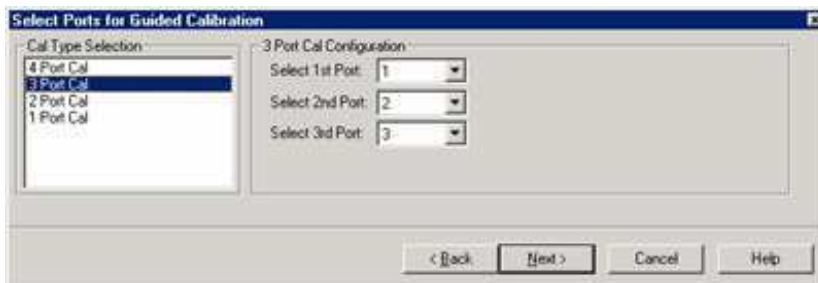
Guided Calibration can perform the following Cal Types:

- ALL Cals **EXCEPT Open, Short, and Thru Response** Cals.
- ECal on one or more ports, beginning with [PNA firmware revision 5.24](#).
- TRL - [Learn how to do TRL cals](#)

### Programming Commands

**Note:** SmartCal DOES allow you to measure calibration standards in any order. However, you must click **Next** and **Back** without measuring standards until you get to the standard you want to measure.

The PNA displays the following dialog boxes when performing a Guided calibration.



### Select Ports for Guided Calibration dialog box help

Allows you to select ports to calibrate.

**Cal Type Selection** Select the number of ports to calibrate.

**NPort Cal Configuration** If not calibrating all PNA ports, specify which ports to calibrate.

**Show Advanced Settings** (Orientation & Thru Cal Section) Available only for [ECal](#).

**Back** Return to [Cal Wizard Begin](#) dialog. If you did not see the 'Cal Wizard Begin' dialog but want to, click

**Back**, then clear the [Save Preferences](#) checkbox.

For greater than 4-port cals, see [External Test Set calibration - Select Cal Type](#).



### Select DUT Connectors and Cal Kits dialog box help

Allows you to select the connector type and Cal Kit for each DUT port to be calibrated.

**Connectors** To change selection, click the connector field for each DUT port.

If your DUT connectors are:

- **Waveguide** Change the system impedance to 1 ohm before performing a calibration. See [Setting System Impedance](#).
- **Not listed** You can create your own connector type and calibration kit file. The PNA includes the following example cal kits that can be used as a template. See [Calibration kits](#) for more information.
  - If using a gendered (male and female) connector type, select **Type A** as the connector type.
  - If using a connectorless device such as on-wafer probes., select **Type B** as the connector type.

**Cal Kits** Select the Cal Kit to be used to calibrate each test port. The list for each DUT Port displays kits having the same connector type as the DUT.

**Identical ECal models connected?** ECal modules can be distinguished by serial number. This can have implications on your remote [SCPI](#) programs.

## 85056K

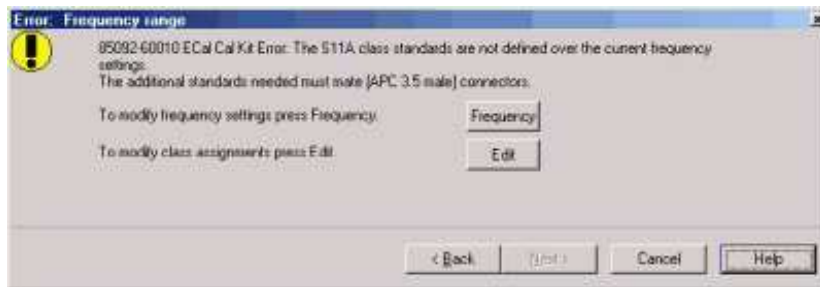
- To calibrate 2.4 mm connectors using the 85056K cal kit, select 85056D as the cal kit. The 85056K definitions in the PNA are for 2.92mm standards (2.4mm plus 2.92 adapters). The 85056D kit contains exactly the same standards WITHOUT the adapters.

## TRL

- To perform a [TRL Cal](#), assign a TRL Cal Kit to the lowest port number of each port pair.
- When selecting a TRL Cal Kit on a 4-port PNA, and a [Global Delta Match Cal](#) is not available, the Cal type will be set to SOLT and a "Could not find a Global Delta Match Cal." message is displayed on the dialog box. If the selected Cal Kit will not support SOLT, the **Next** button will not be available. Then you must select a different Cal Kit to proceed or **Cancel** and perform a Global Delta Match Cal.

**Modify Cal** Check, then click Next, to [Modify Cal](#) (Standards AND Thru Method).

For greater than 4-port cals, see External Test Set calibration - Select DUT Connectors.



### Error: Frequency Range dialog box help

The current cal kit does not cover the current frequency range of the measurement. Do one of the following to correct the problem:

**Frequency** Change the frequency range of the active channel.

**Edit** Modify the class assignments so that a different standard is selected.

**Back** Select a different Cal Kit that covers the required frequency range.

**Cancel** Exit the Cal Wizard





## Modify Cal dialog box help

### Thru #n

Lists the proposed Thru connections to be made during the calibration process. You can change these Thru connections to better suit your test setup.

- The proposed Thru connections are listed automatically.
- Additional Thru connections can be selected for higher accuracy. [Learn more](#).

### Add Thru

Click to add a Thru connection. [Learn more](#)

### Remove Thru

Select a Thru by clicking the "Thru #N" field or the "1st Port / 2nd Port" field. Then click "Remove Thru". This selection is NOT available if the selected Thru is required for the calibration.

### 1st Port / 2nd Port

Click to select the two ports to be included in the Thru connection. The order of the port numbers is not critical.

### Thru Cal Method

Lists the available Thru Cal methods for the specified port pairs.

[Learn about the Thru Cal Method choices.](#)

### Cal Type/ Stds

Click to invoke the [View / Modify Properties of Cal dialog box](#)

### Do orientation - Appears ONLY if an ECal module is selected for use.

When this box is checked (default) the PNA automatically senses the model and direction in which an ECal module port is connected to the PNA ports. If power to the ECal module is too low, it will appear as if there is no ECal module connected. If you use low power and are having this problem, clear this check box to provide the orientation manually.

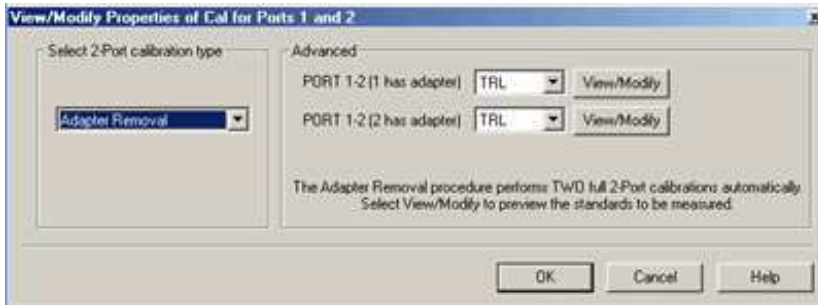
Orientation occurs first at the middle of the frequency range that you are calibrating. If a signal is not detected, it tries again at the lowest frequency in the range. If you have an **E8361A** or **E836xB** PNA and do an ECal completely within 10 - 20 MHz OR 60 - 67 GHz, you may need to do orientation manually. There may not be sufficient power to orient the ECal module at those frequencies.

**Choose delta match** - Available when a Delta Match Cal is required.

- Check, then click **Next** to invoke the [Select Cal Set for Delta Match](#) dialog box.
- Clear - The Cal Wizard uses the [Global Delta Match Cal](#) if available.

**View/Detect ECal Characterizations** - Appears ONLY if an ECal module is selected for use.

Click to invoke the [View ECal Modules and Characterizations](#) dialog box. Displays a list of ECal modules that are connected to the PNA.



#### View/Modify Properties of Cal for Ports... dialog box help

##### Select calibration type

Another chance to change the Thru method.

[Learn about the Thru Cal Method choices.](#)

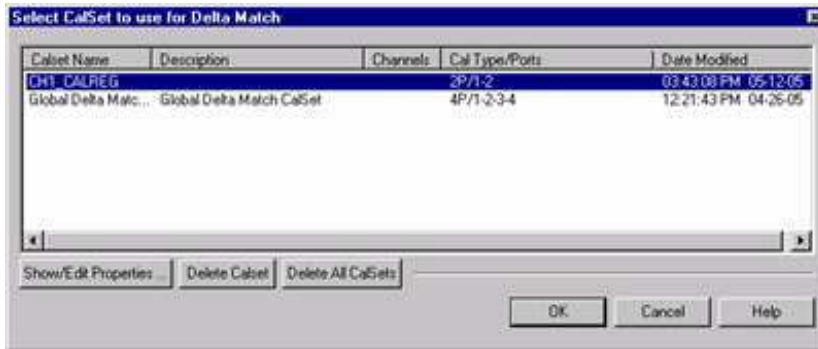
##### Advanced

Select the cal method for each connector of the Thru pair.

- [TRL](#) is only available if a TRL cal kit was selected for the lowest port number of the port pair.
- [QSOLT](#) Only available when "Defined Thru" or "Flush Thru" is selected.  
"QSOLT 2 <= 1" refers to the receive port 2 and source port 1 (where reflection standards are connected).
- [Enhanced Response](#) Only available when "Defined Thru" or "Flush Thru" is selected. "EnhResp 2 <= 1" refers to the receive port 2 and source port 1.

**View Modify** Click to invoke the [Preview and Modify Calibration Selections](#) dialog box.

**Note:** Changes made to the Cal Kit through this dialog are **temporary** that last only for this calibration. To make permanent changes to the Cal Kit, perform [Advanced Modify Cal Kits](#).



### Select Cal Set for Delta Match dialog box help

This dialog box appears when a Delta Match Cal is required and [Choose delta match](#) was selected. [Learn more.](#)

Displays the Cal Sets that meet the requirements of the Delta Match Cal.

Select either a User Cal Set or [Global Delta Match Cal](#).

If there is no suitable choice for a Delta Match Cal:

1. Click **Cancel**, then **Cancel** again to quit the Cal Wizard.
2. Perform either a [Global Delta Match Cal](#) or a SOLT cal and save the result in a User Cal Set.
3. Start the Cal Wizard to re-initiate this calibration.
4. Select the Global Delta Match Cal or User Cal Set.



### Calibration Steps dialog box help

**Note:** Beginning in PNA Rev. 6.0, calibration can be performed with External triggers. [Learn more.](#)

As each new cal step prompt appears, the traces are setup for the next standard measurement. Also, sweeps are triggered continuously until the Measure button is pressed. This way you can view the integrity of the standard connection.

Prompts for standards to be measured.

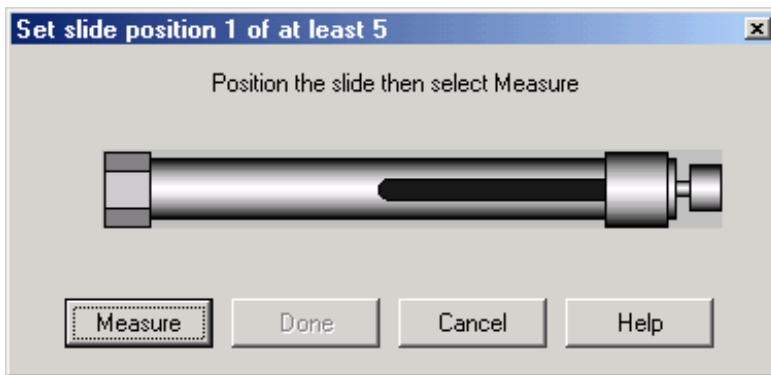
**Measure** Click to measure the standard.

**Done** Click **after** a standard is re-measured and all measurements for the calibration are complete.

**Next** Click to continue to the next calibration step. Does **NOT** measure the standard.

If a standard is NOT measured, a warning appears and **Done** will not be available after the last Cal step.

**Note:** SmartCal DOES allow you to measure calibration standards in any order. However, you must click **Next** and **Back** without measuring standards until you get to the standard you want to measure.



### Sliding Load Measurement dialog box help

Allows you to measure the sliding load standard. [Learn more about the Sliding Load standard.](#)

To ensure an accurate calibration, carefully follow the instructions that were provided with your sliding load.

#### To Measure a Sliding Load:

1. Connect the sliding load to the measurement port.
2. Position the sliding element, then click **Measure**. Do not move the sliding element until measurement is complete.
3. Measure the sliding load for at least **five** and up to **seven** positions for best accuracy.

**Note:** The positions of the sliding element should cover the full length of the slide, but be unequally spaced to reduce the possibility of overlapping data points. Most sliding loads have marks for each slide position.

4. Click **Done** after the final measurement.

5. Remove sliding load from the measurement port.
6. Measure the remaining standards.



### Specify delay dialog box help

This dialog appears ONLY when [Adapter Removal](#) or [Unknown Thru](#) calibrations are performed.

The following values were estimated from the measurement. Most of the time, they are adequate. However, for CW sweep or frequency sweep with large step sizes, the accuracy of the values may be improved.

**Adapter delay** To improve this value, measure and record the delay of the adapter with a dense step size. Enter that value here. The required precision value is the accuracy that is required to characterize the delay value.

**Nominal phase offset** (Waveguide ONLY). To improve this value, measure and record the phase offset of the Waveguide adapter with dense step size. Enter that value here.

When one connector is coax and the other connector is waveguide, the phase offset has an ambiguity of 180 degrees. For consistency, the estimate provided here is always between 0 and 180 degrees. You can change this estimate to any value between -180 degrees and +180 degrees.

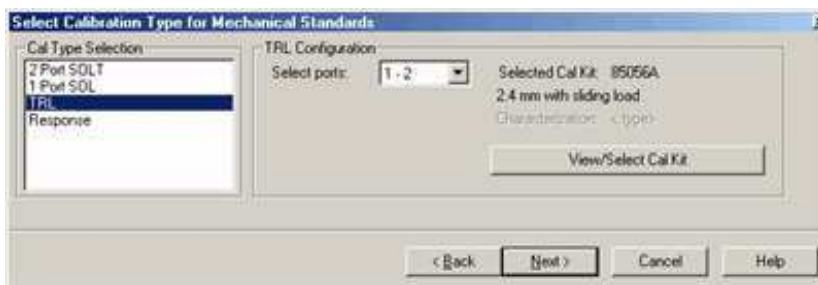
**For FCA calibrations**, this dialog box appears twice: once for the input frequencies and once for the output frequencies. The values can be slightly different.

The [Calibration Complete](#) dialog box appears after all standards are measured.

## Unguided Calibration

### Programming Commands

The PNA displays the following dialog boxes when performing an Unguided calibration:



## Select Calibration Type for Mechanical Standards dialog box help

Unguided calibration does **NOT** support cals greater than 2 ports or **E**Cal calibrations.

TRL Cal should be performed using [Guided Calibration](#).

### Calibration Type Selection

- **2-Port SOLT**
- **1-Port SOL**
- **TRL** - NOT available on PNA models with more than 2 ports.
- **Response** - Reflection and Thru (if the active measurement is transmission)

**Cal Configuration** If not calibrating all PNA ports, specify which ports to calibrate.

**Back** Return to [Cal Wizard Begin](#) dialog. If checked, you can clear the [Save Preferences](#) checkbox to see the Begin page when the Cal Wizard begins.

**View/Select Cal Kit** Click to invoke the [Select Cal Kit dialog box](#).

**Next** Click to continue to [Measure Mechanical Standards](#) dialog box.

**Note:** If the DUT connector type has an impedance other than 50 ohms (waveguide = 1 ohm), change the system impedance before performing a calibration. See [Setting System Impedance](#).



### Select Cal Type dialog box help

This dialog box only appears if the selected Cal Type is TRL in the previous dialog box.

**TRL Reference Plane** Select which standard to use to establish the position of the measurement reference plane.

**THRU Standard** Select if the THRU standard is zero-length or very short.

**REFLECT Standard** Select if the THRU standard is not appropriate AND the delay of the REFLECT standard is well defined.

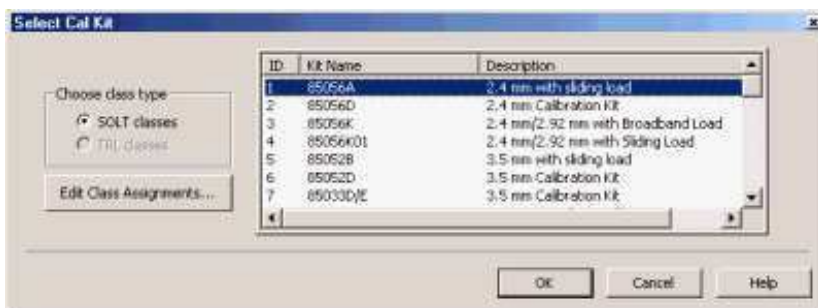
#### TRL Impedance

**LINE Standard** Specifies that the characteristic impedance of the LINE standard should be used as the system impedance. This ignores any difference between Offset Z0, Offset Loss, and System Z0.

**SYSTEM Impedance** Transforms the LINE standard impedance and loss to that of the system impedance for use with the calibration error terms. The TRL calibration will first compute the error terms assuming the LINE standard impedance is the system's characteristic impedance (same as previous LINE selection), then modify the error terms to include the impedance transformation. This should only be used with coax since the skin effect model used is a coaxial model.

[Learn how to change System Z0.](#)

To learn to substitute other calibration kits, see [Advanced Modify Cal Kits](#)



### Select Cal Kit dialog box help

Displays the calibration kit files available for Unguided calibration. Select the desired calibration kit file and click **OK**.

**Choose class type** Unguided TRL calibration is NOT available on the 4-port PNA.

**Edit Class Assignments** Allows modification of the selected Cal Kit class assignments.

To learn to substitute other calibration kits, see [Advanced Modify Cal Kits](#)

**Note:** If the Cal Kit has an impedance other than 50 ohms (waveguide=1 ohm), change the system impedance before performing a calibration. See [Setting System Impedance](#).



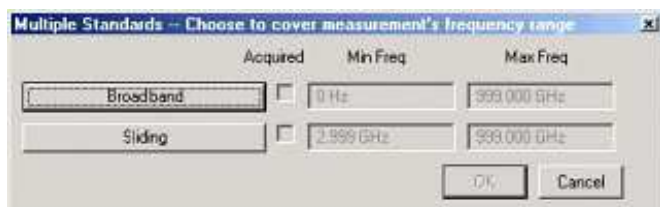
### Measure Mechanical Standards dialog box help

**Note:** Beginning in PNA Rev. 6.0, calibration can be performed with External triggers. [Learn more.](#)

Displays the calibration kit file and standards required for the calibration.

- Standards may be connected and measured in any order.
- Connect the standard to the measurement port and click its associated green button. A check mark indicates the standard has been measured.
- If a standard type contains multiple standards, the [Multiple Standards dialog box](#) opens to display the multiple standards included in the calibration kit file.
- If a sliding load is included in the calibration kit file, the [Sliding Load dialog box](#) opens to perform the measurement with the standard.
- Reflection Response** Select EITHER Open or Short standard, then click **Next**.
- Isolation** Requires one load for each test port of the PNA. [Learn more about Isolation.](#) Use when your measurement requires maximum dynamic range (> 90 dB). See also [Isolation Portion of 2-Port Calibration.](#)
- Normalize** Available when performing a response cal for any measurement. After Normalize is pressed and the Cal is complete, the data trace is flat when the same physical connections are present on the port. This is similar to [Data/Memory](#), except that the response cal is [saved with Cal data](#) and can be applied to other like measurements. Data/Memory is still available after using Normalize. You would usually connect a THRU standard when calibrating a transmission measurement, and a SHORT standard when calibrating a reflection measurement.

**Show Prompts** Check to provide a reminder for the required connection when you click on the standard.





### Multiple Standards dialog box help

Select the standards to be measured.

**Note:** You may see both male and female standards. The Unguided cal has no knowledge of the gender of your connector types. **Choose the gender of your DUT connector;** NOT the test port. Then click OK.

To modify this calibration class to show only one standard, on the Calibration menu, click **Advanced Modify Cal Kits**. Select the Cal kit and click **Edit Kit**. In **Class Assignment**, click **Edit**. Learn more about [Modify Calibration Class Assignments](#).

- Connect the standard to the measurement port and click its associated button. A check mark in the **Acquired** box indicates the standard has been measured.
- To cover the entire frequency range, you may need to measure more than one standard. The order in which the standards are measured is important. The last standard that is measured will override the others in respect to the frequency range of the standard definition. **Example:** In the case of measuring both a broadband load and a sliding load, you would measure the sliding load last. This is because the frequency range of the sliding load is a subset of the broadband load.

Learn more about [Modify Calibration Class Assignments](#)

## Saving a Calibration

SmartCal, ECal, and Unguided Calibrations end with the following dialog box:



### Calibration Completed dialog box help

**Finish** Save to the channel's calibration register.

**Save As User Cal Set** Invokes the [Save as User Cal Set dialog box](#) AND save to the channel's calibration register.

**Cancel** Calibration is NOT applied or saved.

Learn about [Calibration Registers](#).

Learn about [User Cal Sets](#)



### Save as User Cal Set dialog box help

**Existing Cal Sets** - Lists the Cal Set names saved on the PNA.

**Select Cal Set from list or type new name below** Specify a name for the new Cal Set. Either accept the suggested new name, type a new name, or select a name from the list to overwrite an existing name.

**Edit Name** If there is no keyboard, click to start the PNA typing tool that can be used from the PNA front panel.

**Save** Saves the Cal Set to the new Cal Set name.

Learn about [User Cal Sets](#)

Last modified:

3-Sep-2008	Removed legacy content
14-Apr-2008	Added note about Offset Load
4-Mar-2008	Added Cal Window feature
21-Sep-2007	Added note about no TRL on 4-port PNAs
January 20, 2007	Added note about any order for SmartCal.
18-Sept-2006	MQ Major modifications for multiport

## Select a Calibration Type

---

The following calibration types are available in the PNA.

Cal Type	Interface	Accuracy	<a href="#">Thru Methods</a> allowed
<a href="#">TRL Family</a>	SmartCal	Very High	All <b>except</b> Unknown Thru
<a href="#">SOLT</a>	Both	High	All
<a href="#">Enhanced Response</a>	SmartCal	High	Defined Thru or Flush Thru
<a href="#">QSOLT</a> (Quick SOLT)	SmartCal	Medium	Defined Thru or Flush Thru
<a href="#">1-Port Reflection</a>	Both	High	Not Applicable
<a href="#">Open/Short</a> Response	Unguided	Low	Not Applicable
<a href="#">Thru</a> Response	Unguided	Low	Known Thru or Flush Thru

[Learn how to select a default Cal Type.](#)

### Other Cal Types (Separate Topic)

- [Source and Receiver Power Cals](#)
- [FCA Scalar and Vector Mixer Cals](#)

[See other Calibration Topics](#)

### TRL Family

Application: Used to accurately calibrate any pair of ports when calibration standards are not readily available.

**Note:** A [Delta Match Cal](#) is required to cal test ports that do not have a dedicated reference receiver.

[Learn more about TRL family cal](#)

For more information on modifying standards, see [Calibration Standards](#).

---

Calibration Method: [SmartCal](#)

---

General Accuracy: Very High

---

Standards Required: THRU, REFLECT, LINE or similar combination

---

Systematic Errors Corrected:

Directivity

Source match

Isolation ([see exceptions](#))

Load match

Frequency response transmission tracking  
Frequency response reflection tracking

---

## SOLT

Application: Used to accurately calibrate any number of ports.

---

General Accuracy: High

---

Calibration Method: [SmartCal](#), [Unguided Calibration](#), [ECal](#)

---

Standards Required: (SHORT, OPEN, LOAD, THRU) or ECal module

---

Systematic Errors Corrected (on all ports):

- Directivity
- Source match
- Isolation ([see exceptions](#))
- Load match
- Frequency response transmission tracking
- Frequency response reflection tracking

---

## Enhanced Response

Application: Used to calibrate two ports when only measurements in one direction (forward OR reverse) are required. Measurements are faster because a second sweep is NOT required.

Reflection Standards (OPEN, SHORT, LOAD) are connected to the source port to be calibrated.

[Defined THRU](#) or [Flush THRU](#) standard is connected between port pairs.

Much quicker than SOLT when using a mechanical cal kit. ECal can also be used.

### To select Enhanced Response:

For a standard S-parameter Cal, select **SmartCal** in the Cal Wizard.

Then, for all cals:

1. At the 'Select DUT Connectors page', check [Modify Cal](#), then click **Next**.
2. Under 'Cal Type', select **Enhanced Response**.

Enhanced Response cal also be selected as the default Cal Type using [Cal Preferences](#).

---

General Accuracy: High

---

Calibration Method: [SmartCal](#), [ECal](#)

---

Standards Required: (SHORT, OPEN, LOAD, [Defined THRU](#) or [Flush THRU](#))

---

Systematic Errors Corrected:

- Directivity (source port)
- Source match (source port)
- Isolation ([see exceptions](#))
- Load match (receiver port) - used only to produce transmission tracking term.

Frequency response transmission tracking (receiver port).  
Frequency response reflection tracking (source port).

---

## QSOLT (Quick SOLT)

Application: Used to quickly calibrate any number of ports. Developed specifically for use with [external multiport test sets](#).

**Note:** A [Delta Match Cal](#) is required to cal test ports that do not have a dedicated reference receiver.

Reflection Standards (OPEN, SHORT, LOAD) are connected to only ONE of the ports to be calibrated. The lower port number of the ports to be calibrated is selected by default. This can be changed through the [Modify Cal / Cal Type](#) setting.

[Defined THRU](#) or [Flush THRU](#) standards are connected from the reflection standard port to the remaining ports to be calibrated.

Much quicker than SOLT when using a mechanical cal kit.

Based on TRL math.

---

General Accuracy: Not as high as SOLT

---

Calibration Method: [SmartCal](#), [ECal](#)

---

Standards Required: (SHORT, OPEN, LOAD, [Defined THRU](#) or [Flush THRU](#))

---

Systematic Errors Corrected:

Directivity

Source match

Isolation ([see exceptions](#))

Load match

Frequency response transmission tracking

Frequency response reflection tracking

---

## 1-Port (Reflection)

Application: Used to accurately calibrate any single test port for reflection measurements only.

---

Calibration Method: [SmartCal](#), [Unguided Calibration](#), [ECal](#)

---

General Accuracy: High

---

Standards Required: (SHORT, OPEN, LOAD) or ECal module

---

Systematic Errors Corrected:

Directivity

Source match

Frequency response reflection tracking

---

## Open / Short Response

Application: Used to quickly calibrate any single test port for reflection measurements only.

Calibration Method: [Unguided Calibration](#)

General Accuracy: Low

Standards Required: OPEN or SHORT

Systematic Errors Corrected:

Frequency response reflection tracking

## Thru Response (Isolation Optional)

Application: Used to quickly calibrate any pair of test ports for transmission measurements only.

Isolation is not usually recommended. Learn more about [Isolation](#)

Calibration Method: [Unguided Calibration](#)

General Accuracy: Low

Standards Required: THRU

Isolation: One LOAD for each PNA test port.

Systematic Errors Corrected:

Frequency response reflection tracking

Isolation

Last modified:

5-Sep-2008	Added note for ER Load Match
16-Apr-2008	Removed AR for TRL limitation
23-Feb-2007	Added Enhanced Response
12-Sept-2006	Added QSOLT

## Using Calibration Sets

---

- [What are PNA Cal Sets](#)
- [Cal Registers and User Cal Sets](#)
- [How to Manage and Apply Cal Sets](#)
- [Examples of Cal Set Usage](#)
- [Archiving Cal Sets using .cal files](#)

[See other Calibration Topics](#)

### What are PNA Cal Sets

At the completion of a calibration, all calibration data is stored to a Cal Set. The Cal Set can be applied later to any channel that has the same stimulus settings as the Cal Set, thereby saving the time it takes to perform another calibration. The following data is saved to a Cal Set:

- Name
- Cal Set Description
- Cal Set Attributes - stimulus settings, cal type, port association
- Standards data
- Error term data
- GUID (**G**lobally **U**nique **I**dentifier)

### Cal Registers and User Cal Sets

There are two types of Cal Sets:

- **Cal Registers** (channel specific)
- **User Cal Sets**

Calibration data is automatically saved to a Cal Register at the end of every calibration. You can also choose to save the cal data to a User Cal Set.

### Calibration Registers

New with PNA Release 5.0, Calibration Registers are designed to simplify calibrations for most users. When a calibration is complete, the data is automatically saved to the channel's Cal Register, overwriting (or [appended to](#)) the previous cal data stored in that register. This concept is similar to previous Agilent Vector Network Analyzers.

- Every channel has ONE dedicated Cal Register. They are named CHn\_CALREG, where **n** is the channel number. The name cannot be changed.
- Cal Registers are more volatile because they are overwritten (or [appended](#)) each time a calibration is performed on that channel. The Cal data is always saved, but only temporarily.
- Cal Registers can be applied to other measurements, but ONLY on the same channel as the Cal Register.

## User Cal Sets

At the end of a calibration, you can choose to also save cal data to an existing or new User Cal Set.

- User Cal Sets can be applied to any number of channels simultaneously.
- User Cal Sets are named by you for easy identification.
- You can have an unlimited number of User Cal Sets.
- At any time, you can copy Cal Register data to create a User Cal Set. See [Cal Set Properties](#).

**Note:** You can [set a Preference](#) to revert the pre-5.0 behavior - Cal data was automatically saved to a User Cal Set instead of a Cal Register.

## Appending Data in a Cal Set

At the end of a calibration, data is saved to the channel's Cal Register and, if you choose, to a User Cal Set. When you choose to save to an existing User Cal Set, the PNA attempts to append the new error terms to the existing User Cal Set. The existing Cal Set data is completely overwritten UNLESS the new data can coexist with the existing data according to the following two rules:

- The stimulus settings of the new data must exactly match the existing data.
- The new cal must involve different ports from the existing cal.

For example:

**Case 1** - An existing Cal Set contains a full 2-port cal between ports 1 and 2. Using the same stimulus settings, you perform a 1-port cal on port 3. At the end of the cal, you click [Save As User Cal Set](#) and select the existing full 2-port User Cal Set.

**Result:** The 1-port cal is appended to the 2-port User Cal Set. There is NO overlap between them.

**Case 2** - Same situation as Case 1, except the 1-port cal is performed on port 1.

**Result:** The Cal Set will contain a 1 port cal on port1 and a 1 port cal on port 2. The overlapping tracking terms are removed rendering the original full 2 port cal invalid.

## How to Manage and Apply Cal Sets and Cal Types

The PNA attempts to apply a Cal Set, and turn error correction ON, for ALL of the measurements on the active channel. This may not always be possible. For example, suppose a channel contains both S11 (reflection) and S21 (transmission) measurements. If a Cal Set that contains only an S11 **Cal Type** is applied to that channel, the Cal



Set does not contain the error terms to correct the S21 measurement. Error correction is turned ON for the S11 measurement and NOT turned on for the S21 measurement.

There are two ways to apply an existing Cal Set (Cal Register or User Cal Set) to a measurement:

1. Recalling an Instrument State with Cal data ([.cst file](#)) - A .cst file contains an Instrument State with all measurement attributes AND a 'pointer' to the Cal Set that was used to calibrate the measurement. Before saving a .cst file, be sure that a User Cal Set (NOT a Cal Register) is being used for the measurement. Because Cal Registers are automatically overwritten when a new calibration is performed, it is likely that the Cal Register data will change before the .cst file is recalled.
2. Create a new measurement and select a Cal Set to apply to the active channel.

**Note:** NEVER copy or modify Cal Sets from Windows Explorer or other applications. Cal Sets should only be accessed through the PNA Application.

### How to select and apply a Cal Set to the active channel

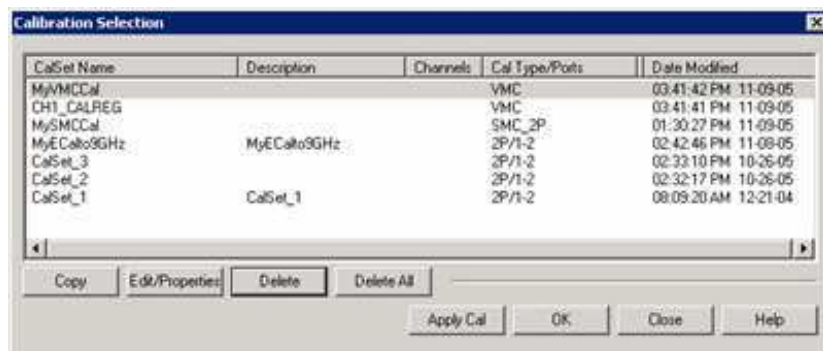
#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Manage Cals]**
3. then **[Cal Set]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Manage Cals**
4. then **Cal Sets**

### Programming Commands



### Calibration Selection dialog box help

This dialog box allows you to manage and apply Cal Sets.

Although the number of Cal Sets you can have is limited only by the amount of PNA memory, it is considered unusual to have more than about 10 existing Cal Sets, or one current Cal Set for every unique channel setup. Old Cal Sets (with 'stale' data) should be deleted or overwritten.

The active channel's Cal Register always appears, even if empty. Cal Registers that belong to other channels

appear in the list of Cal Sets only if the channel exists, and only if they contain data.

- Learn about [Cal Registers](#).
- Learn how to [View the Error Terms of a Cal Set](#).

**To apply a Cal Set** to the active channel, click a row to select that Cal Set, then click Apply Cal.

**Note:** A Cal Set must have been generated from the same [measurement class](#) as the active channel in order for it to be Applied.

**Columns** click a heading to sort by that column

**Cal Set Name** Name to identify the Cal Set.

**Description** User-settable text to further identify the Cal Set.

**Channels** Channel numbers that are currently using this Cal Set. A blank entry means it is not currently in use.

**CalType / Ports** Type of Cal contained in the Cal Set. [Learn about applying appropriate Cal Types](#).

**Cal Type Abbreviations:**

**1P, 2P, 3P, 4P...** - Full n-Port calibrations

**R** - Response (instead of ports, shows the measurement type that it corrects.)

**ER/x-y** [Enhanced Response](#), where **x** is the receive port; **y** is the source port.

**VMC** [Vector Mixer Cal](#)

**SMC** [Scalar Mixer Cal](#)

**Modified** Date and time the Cal Set was last modified.

## Buttons

**Copy** Invokes the [Save as User Cal Set](#) dialog box. Type a name for the copy of the selected Cal Set data.

**Show / Edit Properties** Invokes the Cal Set Properties dialog box. This allows you to view all of the Cal Set properties and create a **duplicate** User Cal Set from an existing User Cal Set or Cal Register.

**Delete** Permanently deletes the Cal Set after you choose OK to a warning prompt.

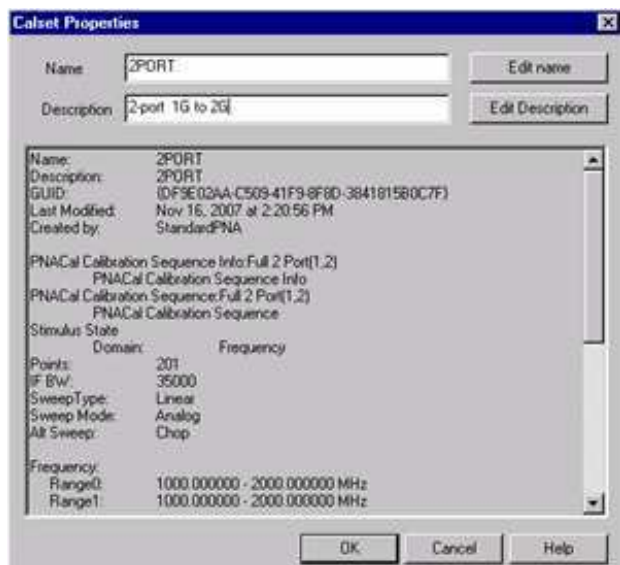
**Delete All** Permanently deletes ALL listed Cal Sets and Cal Registers after you choose OK to a warning prompt.

**Apply Cal** Applies the selected Cal Set to the active channel. If the stimulus settings of the Cal Set and channel are different, [a choice must be made](#).

**Unselect** Available ONLY if the selected Cal Set is being used by the active channel. Click to 'Un-apply' the Cal Set, then click **Close** to exit with the Cal Set un-applied.

**OK** Always APPLIES THE SELECTED CAL SET to the active channel, then closes the dialog box.

**Close** Exit the dialog box. Performs no further action.



### Cal Set Properties dialog box help

Allows you to view all of the Cal Set properties and create a **duplicate** User Cal Set from an existing User Cal Set or Cal Register.

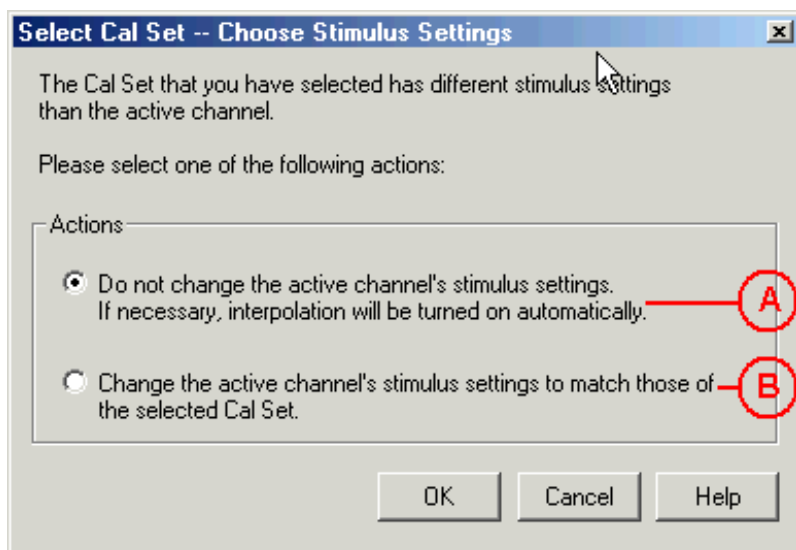
**Name** Edit name of the User Cal Set. You can NOT change the name of a Cal Register.

**Description** Descriptive text to further identify the Cal Set.

**Cal Set Properties** Lists descriptive information and stimulus conditions of the Cal Set.

Learn how to [View the Error Terms of a Cal Set](#).

### Stimulus Setting Different between Cal Set and Measurement



### Select Cal Set -- Choose Stimulus Settings dialog box help

The Cal Set contains the channel stimulus settings that were in place when the Cal Set was saved. This dialog appears when the Cal Set channel settings are different than those of the channel to which the Cal Set is being applied. Choose between the following options.. (See above image).

- A. Keep the Active Channel Stimulus settings. Interpolate if possible.
  - If the Cal Set frequency range is greater the active channel, then Interpolation will be turned ON. Learn more about [Interpolation Accuracy](#)
  - If the Cal Set frequency range is less than the active channel, then this option is not available.
- B. Keep the Cal Set Stimulus settings. The Active Channel stimulus setting are changed.

**OK** Make the change.

**Cancel** Cal Set will NOT be applied.

## Examples of Cal Set Usage

The following examples show how Cal Sets increase flexibility and speed in making analyzer measurements.

- Using one User Cal Set with many Channels
- [Using one Measurement with many Cal Sets](#)

### Using one User Cal Set with many Channels

It is possible to do one calibration, then apply it to several channels.

#### An example:

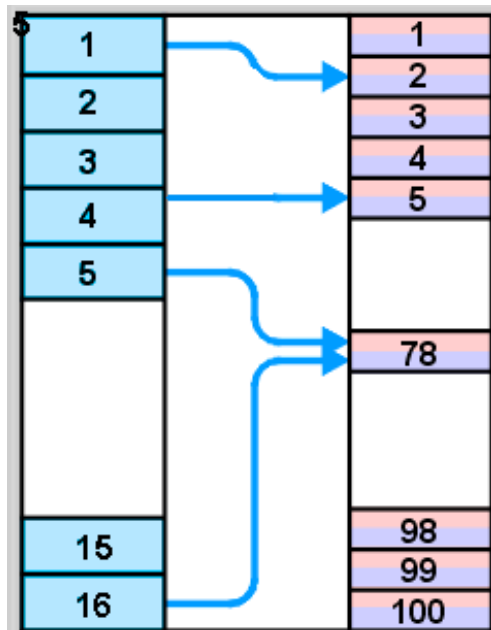
During a manufacturing process, you may have many calibrated channels. You may wish to continuously cycle through the measurements and examine them individually. Occasionally, you may wish to refresh the calibration without having to recreate all the measurement state files.

**Here is how:** Examine the stimulus settings for each channel. Then make the User Cal Set stimulus range a super-set of the whole group. Each channel can then use the same User Cal Set. Some calibrations will be interpolated. **Note:** Make sure that [interpolation](#) is turned on.

Notice in the following image, Cal Set 78 is used on more than one channel, in this case Channel 5 and 16 .

**Channel**

**Cal Set**



## Using one Measurement with many Cal Sets

The drawback with having one very large User Cal Set associated with many instrument states could be a loss of accuracy due to interpolation. In such cases, consider using one User Cal Set for each stimulus setting.. The stimulus conditions can then be changed for a channel by applying different User Cal Sets. Other settings (window setups, measurement definitions, scaling, limits, markers) will not change. This may result in faster state changes than if you saved and recalled \*.cst files for each set of stimulus conditions.

**Example #1:** An amplifier needs to be measured at several input power levels. Calibrate at several power levels and save each calibration in a separate User Cal Set. Then, apply the User Cal Sets to the single measurement consecutively.

**Example #2:** Making an S21 Measurement, you need to measure both wide span and narrow span characteristics of the device. One Cal Set covers the wide span setup; another the narrow span setup.

## Archiving Cal Sets using .cal or .csa files

Because User Cal Sets can easily be deleted, provide extra backup by also saving your calibration as a .cal or .csa file ([see saving a .cal file](#)).

### Example:

One person performs a calibration, names and saves it as a User Cal Set. This Cal Set is available for any other person to use. A second user could accidentally delete or modify the User Cal Set requiring the originator to repeat the calibration.

Security can be provided for calibration data by saving the Cal Set to a **.cal** file or .csa file. At a later time, the file could be recalled and the original calibration restored.

---

Last modified:

17-Sep-2009	Edit wording in Appending...
3-Sep-2008	Removed legacy content
22-Apr-2008	Added link to UI Cal Set preference
9/12/06	Added link to programming commands

## Error Correction and Interpolation

---

Error Correction and Interpolation settings work together to provide you with the highest level of calibration accuracy possible.

- [How to set Error Correction](#)
- [Error Correction](#)
- [Viewing Correction Levels](#)
- [How to set Interpolation](#)
- [Interpolation and Accuracy](#)

### See other Calibration Topics

#### How to set Error Correction

##### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Correction ON/off]**

##### Using a mouse with PNA Menus

1. Click **Cal**
2. then **Correction ON/off**

◀ **Programming Commands** ▶

## Error Correction

The Error Correction ON setting means that the calibration error terms are applied to the measurement. Error Correction is automatically turned ON when a calibration is performed or if a Cal Set is applied to a measurement. The PNA attempts to turn error correction ON for ALL of the measurements on the active channel. This may not always be possible when applying Cal Sets. For more information, see [Applying Cal Sets](#).

When full 2-port error correction is ON, both forward and reverse sweeps are required to gather all 12 error terms, even if only one reflection measurement is displayed. This may result in a higher measurement speed than expected. [Learn more](#).

You can always turn Error Correction OFF for the active measurement by clicking Correction OFF. The PNA will turn Error Correction OFF automatically when making stimulus changes [under some conditions](#). To turn correction back ON, click **Correction ON**. Then:

- If Interpolation can NOT be performed, a dialog box will ask if you would like to [change the stimulus settings](#) to those of the applied calibration. Click OK or Cancel.

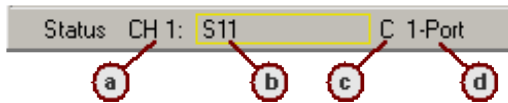
- If Interpolation can be performed, the stimulus setting will change and correction turned ON.

## Viewing Correction Level

The correction level provides information about the accuracy of the active measurement. Correction level notation is displayed on the status bar for different calibration types like response, full 2-port, TRL, or power calibration.

### To View Correction Levels:

In the **View** menu, click **Status Bar**. The status bar appears and displays the following items:



- Active Channel
- Measurement parameter
- Correction Level (see description below)
- Calibration type

Correction Level		Accuracy
C	Full	Highest
C*	Interpolated	Uncertain
C	Changed	Uncertain
No Cor	No Correction	Lowest

### C Full Correction

Full Correction level is displayed immediately after a calibration is performed or when a valid Cal Set is applied. If you require optimum accuracy, avoid adjusting analyzer settings after calibration so your measurement remains at this level.

### C\* Interpolated Correction

"C star" appears in the status bar when a measurement is being interpolated. See Interpolation (above) and [Interpolation Accuracy](#).

### C Changed Settings

"C-delta" appears in the status bar when one or more of the following stimulus settings change. The resulting measurement accuracy depends on which parameter has changed and how much it has changed. For optimum accuracy, recalibrate using the new settings.

- [Sweep time](#)



- [IF Bandwidth](#)
- [Port power](#)
- [Stepped sweep enabled/disabled](#)

## No Corr No Correction

The following will cause the PNA to turn Error Correction OFF for the channel:

- Decrease the start frequency
- Increase the stop frequency
- Change start frequency, stop frequency, or number of points with Interpolation OFF.
- [Change sweep type](#)

### How to set Interpolation

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[More]**
3. then **[Interplotion ON/off]**

#### Using a mouse with PNA Menus

1. Click **Cal**
2. then **More**
3. then **Interplotion ON/off**

◀ Programming Commands ▶

## Interpolation

Calibration interpolation adjusts calibration error terms to match changes to the following settings that you make AFTER a calibration is performed or a [Cal Set applied](#).

The Interpolation **ON** setting means that interpolation is **enabled** for the active measurement. This does not necessarily mean that the measurement is interpolated. When enabled (ON), if interpolation becomes necessary because you change any of the following stimulus settings, **then** interpolation will be applied. When stimulus settings change while interpolation is OFF, interpolation is NOT applied but instead, error correction is turned OFF.

Interpolation occurs (if enabled) when you change any of the following settings:

- Start frequency increased
- Stop frequency decreased
- Number of points

**Note:** Decreasing the start frequency, or increasing the stop frequency will always turn correction **OFF**. (Exception: [Power Calibration](#) DOES extrapolate to the start and stop frequencies.)

## Interpolation Accuracy

When a measurement is interpolated, the accuracy of the measurements cannot be predicted. It may be affected significantly or not at all. Identifying measurement errors in these cases must be determined on a case-by-case basis.

Significant measurement inaccuracy **WILL** occur when the phase shift between measurement points increases more than 180 degrees. The PNA will incorrectly interpolate the new phase data. For more information, see [phase accuracy](#).

In general, the chances of significant inaccuracy increase when interpolating measurements under the following conditions:

- when increasing, rather than decreasing, the frequency span between measurement points.
- when frequency span between measurement points becomes much greater.
- when measurement frequencies are very high, especially above 10 GHz.

---

Last modified:

3-Sep-2008    Removed legacy content

Sept 12, 20/06    Added link to programming commands

## Calibration Thru Methods

---

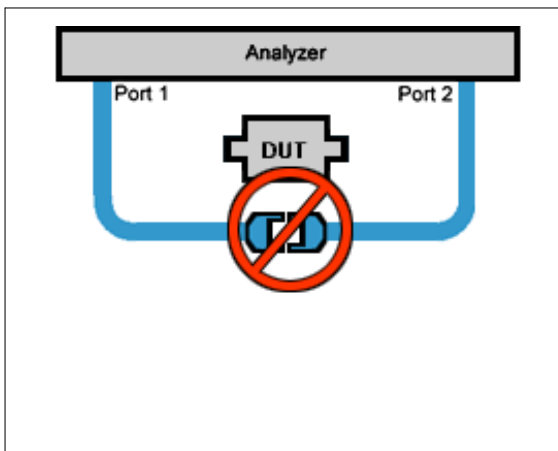
- [What is a Non-Insertable Device](#)
- [Choosing a Thru Method](#)
- [Flush Thru](#)
- [Adapter Removal](#)
- [Defined Thru](#)
- [Unknown Thru](#)
- [ECal Thru Method Choices](#)

### Other Cal Topics

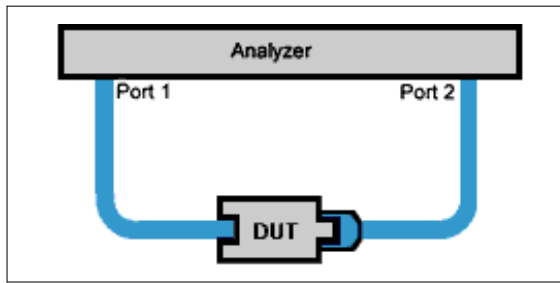
## What is a Non-Insertable Device

To understand the Thru method choices, you must first understand what is meant by "Non-Insertable device". These definitions also apply to ECal modules. Substitute "ECal module" for "device". Then see [ECal Thru Method Choices](#).

A **non-insertable device** is one whose connectors could NOT mate together. They either do not have the same type of connector or they have the same gender. This also means that the test port cables would not mate together, as in the following diagram.



An **insertable** device is one whose connectors could mate together. They have the same type of connector and opposite, or no, gender. This also means that the test port cables would mate together, as in the following diagram.



## Choosing a Thru Method of Calibration

The Thru method is selected from the Cal Wizard. Select the **Modify** checkbox in the [Select DUT Connectors and Cal Kits](#) dialog box.

### Notes:

For ECal, the following choices have different meanings. See [THRU methods for ECal](#).

For 4-port calibration, see [How can we measure only 3 THRU connections?](#)

### Choice for Insertable Devices: FLUSH Thru (also known as Zero-length Thru)

When calibrating for an insertable device, the test ports at your measurement reference plane connect directly together. This is called a zero-length THRU, or Flush THRU meaning that the THRU standard has zero-length: no delay, no loss, no capacitance, and no inductance. Your calibration kit may not have a physical THRU standard because it is assumed you have an insertable device and will be using a zero-length THRU.

### Choices for Non-Insertable Devices

The following methods calibrate for a non-insertable device:

- [Adapter Removal](#) Accurate, but least convenient.
- [Defined Thru](#)
- [Unknown Thru Cal](#) **Preferred method.**
- Swap-Equal-Adapters Method is a valid choice, but NOT included in the PNA firmware.

### Adapter Removal Calibration

This method is potentially very accurate. However, it requires many connections which increases the chances of inaccurate data.

Two full 2-port calibrations are performed: one with the adapter connected at port 1, and the other with the adapter connected to port 2. The result of the two calibrations is a single full 2-port calibration that includes accurate characterization and removal of the mismatch caused by the adapter.

Performing an Adapter Removal Cal requires:

- a THRU adapter with connectors that match those on the DUT.

- calibration standards for both DUT connectors.

To select Adapter Removal during a SmartCal, select the **Modify** checkbox in the [Select DUT Connectors and Cal Kits](#) dialog box. The Cal Wizard will guide you through the steps.

Learn how to perform an [Adapter Removal Cal using ECal](#).

### Defined Thru (also known as **Known Thru**, **Cal Kit Thru**, **ECal Thru**, **Characterized Thru**)

Defined Thru uses the THRU definition that is stored in the Cal Kit file or ECal module. The THRU standard may have worn over time, making it not as accurate as when it was new. Defined Thru is usually more accurate than Adapter Removal, but not as accurate as [Unknown Thru](#) method.

#### Notes

- If performing an ECal, this is the THRU standard in the ECal Module.
- If Defined Thru appears as a potential THRU method in the [SmartCal Wizard](#), this means that there is a defined THRU standard in the selected Cal Kit. This could be a [Zero-length Thru](#). The SmartCal Wizard will prompt you to connect the required standard when appropriate.

To define a THRU standard in a Cal Kit (not ECal module):

1. From the PNA Menu, click Calibration, [Advanced Modify Cal Kits](#).
2. Select the Cal Kit
3. Click Edit Kit
4. Click Add
5. Select THRU
6. Complete the dialog box.

The next time you perform a Guided Cal, this Defined THRU standard will be available if the DUT connector types match the THRU standard.

### Unknown Thru Cal

Unknown Thru Cal is the **preferred** THRU method of calibrating the PNA to measure a non-insertable device.

The Unknown Thru calibration is also known as **Short-Open-Load-Reciprocal Thru** (SOLR) calibration.

- Very easy to perform.
- Better accuracy than [Defined Thru](#) and usually better than [Adapter Removal](#).
- Does not rely on existing standard definitions that may no longer be accurate.
- Causes minimal cable movement if the THRU standard has the same footprint as the DUT. In fact, the DUT can often BE the THRU standard.

## About the Unknown Thru Process

SmartCal guides you through the process. Although the following process describes ports 1 and 2, Unknown Thru can be performed on any two ports when using a multiport PNA.

1. Perform 1-port cal on port 1.
2. Perform 1-port cal on port 2.
3. Connect Unknown Thru between ports 1 and 2.
4. Measure Unknown Thru.
5. [Confirm Estimated Delay](#). This estimate may be wrong if there are too few frequency points over the given frequency span. You can measure the delay value independently and enter that value in the dialog box.

## The Unknown Thru Standard

- Can have up to about 40 dB of loss and long electrical length.
- Must be reciprocal:  $S_{21}=S_{12}$ .
- Must know the phase response to within 1/4 wavelength (see step 5 above).
- Can be the DUT if it meets these conditions.

## Unknown Thru Limitations

- Unknown Thru is NOT supported during a TRL calibration.
- Beginning with PNA code release 5.25, Unknown Thru CAN be performed using a 4-port PNA-L that does NOT have a [reference receiver for each test port](#). However, a [Delta Match Calibration](#) is usually required before the Unknown Thru is measured.
- Unknown Thru is NOT supported on E8801A, E8802A, and E8803A.

## ECal Thru Method Choices

When the ECal module connectors exactly match the DUT connectors, choose from the following THRU methods:

**ECal Thru as Unknown Thru** [Learn more about Unknown Thru](#).

- Measures the THRU state of the ECal module as an Unknown Thru.
- The default method when the ECal module connectors match the DUT.
- Very accurate and easy.
- May require a [Delta Match Cal](#).

### Flush Thru (zero-length Thru) [Learn more about Flush Thru](#)

- Requires an insertable ECal module / DUT.
- Remove the ECal module and connect the two reference planes directly together for a zero-length thru.
- Accurate, but not as easy as 'ECal Thru as Unknown Thru'.

### ECal (Defined Thru)

- Measures the THRU state of the ECal module.
- Very easy, but not very accurate.

### Unknown Thru

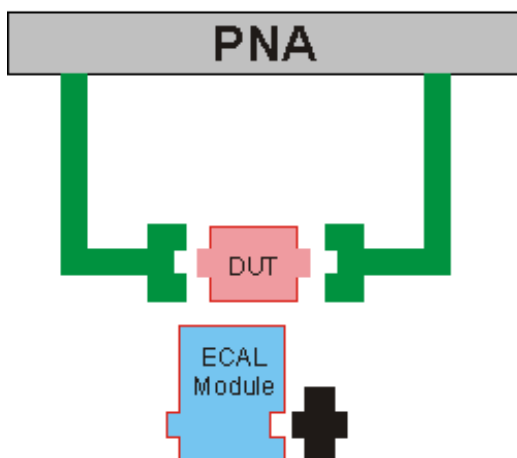
- Remove the ECal module.
- Then connect a Thru adapter to be measured as Unknown Thru.
- May require a [Delta Match Cal.](#)

When the ECal module connectors do NOT exactly match the DUT connectors, choose from the following two methods:

### Adapter Removal

- Can be used with ECal when your DUT is [NON-insertable](#). However, the ECal module MUST be insertable, and the adapter connectors must exactly match the connectors of the DUT as in the following diagram.

**Note:** With PNA release 4.8, adapter removal now performs 2-port measurements on both sides of the adapter. It previously performed 2-port measurements on one side and 1-port measurements on the other. This improves the accuracy of the adapter removal calibration.



## ECal User Characterization

In cases when adapter removal cannot be performed, ECal [User Characterization](#) is ALWAYS possible if you have the right adapters. A User Characterization is performed once and stored in the ECal module. However, accuracy is compromised every time you remove, then reconnect, the adapter with the ECal module.

---

Last Modified:

20-Feb-2008    Added bullet for default ECal method



## Accurate Measurement Calibrations

---

Calibration accuracy is affected by the type of calibration, quality of the calibration standards, and the care with which the calibration is performed. This section provides additional information about how to make accurate calibrations.

- [Measurement Reference Plane](#)
- [Effects of Using Wrong Calibration Standards](#)
- [Data-based versus Polynomial Calibration Kits](#)
- [Accuracy Level of Interpolated Measurement](#)
- [Effects of Power Level](#)
- [Using Port Extensions](#)
- [Isolation Portion of 2-Port Calibration](#)
- [Choosing a Thru Method](#)

Learn how to [determine the validity of your calibration](#).

---

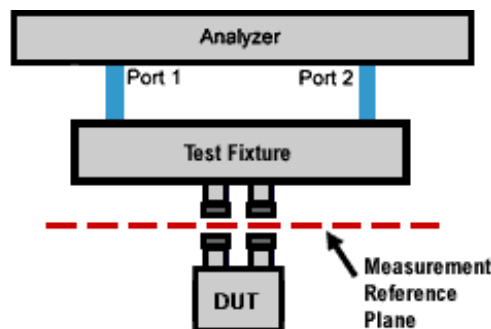
[See other Calibration Topics](#)

---

### Measurement Reference Plane

Most measurement setups will NOT allow you to connect a device under test (DUT) directly to the PNA front panel test ports. More likely, you would connect your device to test fixtures, adapters, or cables that are connected to the PNA.

A calibration takes place at the points where calibration standards are connected during the calibration process. This is called the measurement reference plane (see graphic). For the highest measurement accuracy, make the calibration reference plane the place where your DUT is connected. When this occurs, the errors associated with the test setup (cables, test fixtures, and adapters used between the analyzer ports and the reference plane) are measured and removed in the calibration process.



## Effects of Using Wrong Calibration Standards

Normally, a calibration is performed using a calibration kit that contains standards with connectors of the same type and sex as your device under test.

However, your calibration kit may not always have the same connector type and gender as your device. For example, suppose your device has 3.5mm connectors, but you have a Type-N calibration kit. If you use an adapter to connect the Type-N standards to the 3.5mm test port, then the adapter becomes part of the calibration and NOT part of the test setup. This will result in significant errors in your reflection measurements.

## Data-based versus Polynomial Calibration Kits

The [Select DUT Connectors and Cal Kits](#) dialog box offers a data-based model and a polynomial model for the newest high-frequency cal kits. See PNA Accessories. The data-based models provide higher accuracy for describing calibration standards than the polynomial models. It is RECOMMENDED that the data-based model be used if the most accurate results are desired.

	Data-Based Model	Polynomial Model
How accurate is the model?	Provides highest calibration accuracy. Eliminates the errors that can be the result of polynomial model approximations.	Provides high calibration accuracy.
How does the model define calibration standards?	Uses S-Parameter measurements.	Uses traditional four-term polynomial calibration standard modeling parameters.
How do I manually edit the definitions of the calibration standards when using the model?	Use the <a href="#">Advanced Modify Cal Kit</a> function.	Use the <a href="#">Advanced Modify Cal Kit</a> function.
How do I use the Calibration Wizard with the model?	Use only the SmartCal (Guided) Calibration method.	Use the SmartCal (Guided) or the Unguided Mechanical Calibration methods.

Learn about the ["Expanded Math"](#) feature.

## Effects of Power Level

To attain the most accurate error correction, do NOT change the power level after a calibration is performed. However, when changing power within the same attenuator range at which the measurement calibration was performed, S-parameter measurements can be made with only a small degradation of accuracy. If a different attenuator range is selected, the accuracy of error correction is further degraded.

To check the accuracy of a calibration, see [Validity of a Calibration](#).

## Using Port Extensions

Use the port extensions feature after calibration to compensate for phase shift of an extended measurement

reference plane due to additions such as cables, adapters, or fixtures.

Port extensions is the simplest method to compensate for phase shift, mismatch, and loss of the path between the calibration reference plane and the DUT.

Learn how to apply [port extensions](#).

Learn about [characterizing a test fixture](#).

## Isolation Portion of 2-Port Calibration

The isolation portion of a calibration corrects for crosstalk, the signal leakage between test ports when no device is present. When performing an UNGUIDED 2-port calibration, you have the option of omitting the isolation portion of the calibration.

**Note:** Isolation is never performed on a Smart (Guided) Calibration.

The uncorrected isolation between the test ports of the PNA is exceptional (typically >100dB). Therefore, you should only perform the Isolation portion of a 2-port calibration when you require isolation that is better than 100dB. Perform an isolation calibration when you are testing a device with high [insertion loss](#), such as some filter stopbands or a switch in the open position.

The isolation calibration can add noise to the error model when the measurement is very close to the [noise floor](#) of the analyzer. To improve measurement accuracy, set a narrow IF Bandwidth.

### How to perform an isolation calibration

Isolation is measured when the Load standards are connected to the PNA test ports. For best accuracy, connect Load standards to BOTH test ports each time you are prompted to connect a load standard. If two Loads are not available, connect the untested PNA port to any device that will present a good match.

## Choosing a Thru Method

When calibrating for a non-insertable device, you must choose a method to calibrate for the THRU error terms. This can have a significant effect on measurement accuracy. Learn more about [choosing a thru method](#).

## Validity of a Calibration

---

This section helps you determine if your calibration is valid and how the analyzer displays correction level information for your measurement.

- [Frequency Response of Calibration Standards](#)
- [Validating a Calibration](#)
- [Quick Check](#)
- [ECal Confidence Check](#)
- [Verification Kit](#)

### See other Calibration Topics

#### Frequency Response of Calibration Standards

In order for the response of a calibration standard to show as a dot on the [smith chart display format](#), it must have no phase delay with respect to frequency. The only standards that exhibit such "perfect" response are the following:

- 7-mm short (with no offset)
- Type-N male short (with no offset)

There are two reasons why other types of calibration standards show phase delay after calibration:

1. The reference plane of the standard is electrically offset from the mating plane of the test port. Such devices exhibit the properties of a small length of transmission line, including a certain amount of phase shift.
2. The standard is an open termination, which by definition exhibits a certain amount of fringe capacitance and therefore phase shift. Open terminations which are offset from the mating plane will exhibit a phase shift due to the offset in addition to the phase shift caused by the fringe capacitance.

The most important point to remember is that all standards are measured in order to remove [systematic errors](#) from subsequent device measurements. As a result, if calibration standards with delay and fringe capacitance are measured as a device after a calibration, they will NOT appear to be "perfect". This is an indication that your analyzer **is calibrated accurately and working properly**.

## Validating a Calibration

At the completion of a calibration or selection of a stored Cal Set, validation can accomplish the following:

**Improve Measurement Accuracy** – Once a measurement calibration has been performed, its performance should be checked before making device measurements. There are several sources of error that can invalidate a calibration: bad cables, dirty or worn calibration standards that no longer behave like the modeled standards, and operator error.

**Verify Accuracy of Interpolation** – You should validate the calibration if you are testing a device and the measurements are uncertain because of interpolation. For more information see [Interpolation Accuracy](#).

**Verify Accuracy of Cal Standards** – To check accuracy, a device with a known magnitude and phase response should be measured.

## Quick Check

For this test, all you need are a few calibration standards. The device used should not be one of the calibration standards; a measurement of one of these standards is merely a measure of repeatability.

The following reflection and transmission Quick Check tests can be applied to all test ports.

### To verify reflection measurements, perform the following steps:

1. Connect either an OPEN or SHORT standard to port 1. The magnitude of S11 should be close to 0 dB (within a few tenths of a dB).
2. Connect a load calibration standard to port 1. The magnitude of S11 should be less than the specified calibrated directivity of the analyzer (typically less than -30 dB).

### To verify transmission measurements:

1. Connect a THRU cable (or known device representative of your measurement) from port 1 to port 2. Verify the loss characteristics are equivalent to the known performance of the cable or device.
2. To verify S21 isolation, connect two loads: one on port 1 and one on port 2. Measure the magnitude of S21 and verify that it is less than the specified isolation (typically less than -80 dB).

**Note:** To get a more accurate range of expected values for these measurements, consult the analyzer's specifications.

## ECal Confidence Check

ECal Confidence Check is a method to check the accuracy of a calibration performed with mechanical standards or an ECal module. The confidence check allows you to measure an impedance state in the ECal module (called the confidence state), and compare it with factory measured data stored in the module.

In order for this test to be valid, the test ports of the ECal module must connect directly to the calibration reference plane (without adapters).

**Note:** In the **N469x** series of 2-port ECal modules, from the module minimum frequency up to approximately 2 GHz, the confidence state has a very high amount of transmission loss. In this frequency range, calibrated measurements of transmission S-parameters for the confidence state may vary much more than expected from the Agilent-characterized data in the measurement memory trace. When comparing the measurement trace and memory trace you, ignore the data for frequencies up to 2 GHz.

## How to Perform ECal Confidence Check:

1. Connect ECal module to the analyzer with the USB cable. See [Connect ECal Module to the PNA](#). **Note:** Terminate any unused ECAL ports with a 50 ohm load.
2. Allow the module to warm up for 15 minutes or until the module indicates **READY**.
3. Do one of the following to start ECal Confidence Check

### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[More]**
3. then **[ECal]**
4. then **[Confidence Check]**

### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **More**
4. then **ECal**
5. then **Confidence Check**

## Programming Commands

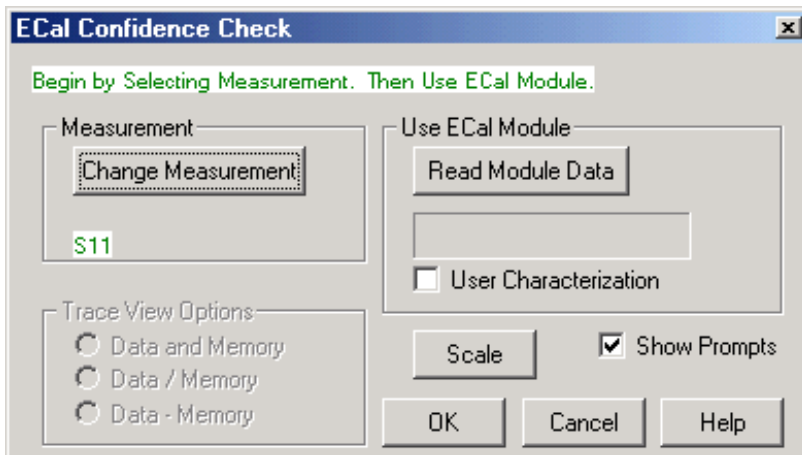
On the following [ECal Confidence Check dialog box](#):

4. Click **Read Module Data**. The following occurs:
  - ECal module is set to "confidence state".
  - PNA reads and displays stored data.
  - PNA measures and displays "confidence state".
5. To view a different parameter, select **Change Measurement** and select the check box for the desired parameter. The default is the active channel parameter.
6. Select the viewing option in the Trace View Options block.
7. Compare the stored and measured data for each measurement parameter.

### Notes:

- After exiting ECal Confidence Check, the ECal module remains in the same impedance state and the factory (or user-characterized) data is still stored in the memory trace. Therefore, you can save both the data and memory trace as a \*.csv files and import them to a spreadsheet. [Learn how.](#)
- If the two traces show excessive difference, there may be a loose or dirty connection at the test ports or damage to the test cables. Carefully inspect the cables and connections. Then clean and gage each connector, and re-calibrate if needed.

- The User Characterization setting selects the user-characterization data instead of the factory characterization data (available when a User-Characterization is stored in the ECal module).



### ECal Confidence Check dialog box help

Compares the accuracy of corrected (calibrated) data with stored data in the ECal module. For the check to be valid, the module test ports must connect directly to the calibration reference plane (without an adapter). [Learn more about ECal Confidence Check.](#)

#### Measurement

**Change Measurement** Opens the Measure dialog box.

#### Use ECal Module

##### Read Module Data

- Copies stored data from the ECal module to Memory.
- Changes state of ECal module to confidence state.
- Measures and displays confidence state and Memory trace.

**User Characterization** Selects the user-characterization data (stored in the module) instead of the factory characterization data (available when a User-Characterization is stored in the ECal module).

**Scale** Opens the Scale dialog box.

**Show Prompts** Check to show a reminder for the connection (default).

#### Trace View Options

**Data and Memory Trace** Displays current measurement data and Memory trace.

**Data / Memory** Performs an operation where the current measurement data is divided by the data in memory.

**Data + Memory** Performs an operation where the current measurement data is added to the data in memory.

## Verification Kit

Measuring known devices, other than calibration standards, is a straightforward way of verifying that the network analyzer system is operating properly. Verification kits use accurately known verification standards with well-defined magnitude and phase response. These kits include precision airlines, mismatch airlines, and precision fixed attenuators. Traceable measurement data is shipped with each kit on disk and verification kits may be re-certified by Agilent.

See [Analyzer Accessories](#) for a list of Agilent verification kits.

---

Last modified:

5-May-2010	Added Note
26-May-2009	Added ECal conf check save
3-Sep-2008	Removed legacy content
9/12/06	Added link to programming commands



## Using ECal

---

This topic discusses all aspects of performing an ECal:

- [ECal Overview](#)
- [Connect ECal Module to the PNA](#)
- [How to Perform a Calibration Using ECal](#)

### See Also:

[ECal User-Characterization](#)

[Restore ECal Module Memory](#)

[See other Calibration Topics](#)

## ECal Overview

ECal is a complete solid-state calibration solution. Every ECal module contains electronic standards that are automatically switched into position during a PNA measurement calibration. These electronic standards have been measured at the factory and the data stored within the memory of the ECal module. The PNA uses this stored data, along with the PNA-measured data, to calculate the error terms for a measurement calibration.

ECal modules are available in 2-port and 4-port models and a variety of connector types, covering many frequency ranges. See [Analyzer Accessories](#) for more about available ECal modules and ordering information.

You can perform the following calibrations with ECal:

- 1-Port Reflection calibration
- Full 2-Port calibration
- Full 3-Port calibration
- Full 4-Port calibration

Verify the validity of a mechanical or ECal calibration with [ECal confidence check](#).

### Care and Handling of ECal Modules

You can improve accuracy, repeatability, and avoid costly repair of equipment in the following ways.

- Practice proper connector care. See [Connector Care](#).
- Protect equipment against ESD damage. Read [Electrostatic Discharge Protection](#).

### Power Level into an ECal module

- NEVER exceed the following Damage levels to the ECal module.
- For highest accuracy, do not exceed the following ECal Compression levels when calibrating:

Model	Compression level	Damage level
N469x series	-5 dBm	+10 dBm
N4432A series N4433A series	-7 dBm	+20 dBm
N4431x series	+7 dBm	+20 dBm
8509x series	+9 dBm	+20 dBm

When using the PNA-X, the power level can be increased after calibration with minimal impact on measurement accuracy.

## Connect ECal Module to the PNA

ECal modules are controlled and powered through a USB connection to the PNA. When you connect the module, the PNA automatically recognizes the type of module, frequency range, and connector type.

### "First Time" Note

When you connect an ECal module that has a serial number never before seen by that PNA, the **Welcome to the Found New Hardware Wizard** will appear.

Click **"No, not this time"**, then click **Next** or **Finish** at the remaining dialog prompts.

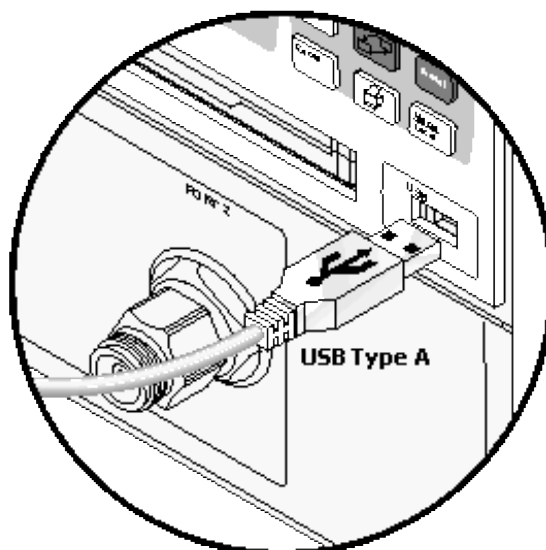
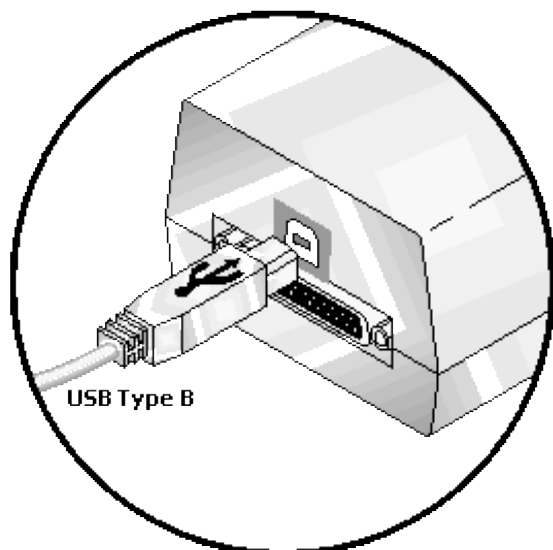
You must be logged on to the PNA with [Administrator privileges](#) to complete the first time registration process. This occurs automatically unless you change your default User Account.

ECal modules connect to the USB port on the front or rear panel of the PNA.

1. Wear a grounded wrist strap when making connections.
2. Connect the USB cable **Type B** connector to the ECal module and the USB cable **Type A** connector to the front or rear panel USB connector of the analyzer, as shown in the following graphics.

**ECal Module USB Port**

**Analyzer Front Panel USB Port**



#### Notes:

- Unused ECal modules that have completed a calibration may remain connected to the USB port.
- You can connect and disconnect the ECal module while the analyzer is operating. However, DO NOT connect or disconnect the module while data transfer is in progress. This can result in damage or at least corrupted data.
- A USB hub may be used to connect more than one USB device to the analyzer. See [Analyzer Accessories](#) for more information about USB peripheral equipment.

#### How to Perform a Calibration Using ECal

Select an ECal module that has connectors of the same type and gender as the DUT. If such an ECal module is not available, a module with connectors different from the DUT can be used by using [Advanced Settings](#) or [User Characterization](#).

Connect the ECal module ports to the PNA ports. During the calibration process the PNA can either automatically detect how the ECal module is connected, or the orientation can be performed manually.

1. Connect the ECal module USB cable to the analyzer USB. See [Connect ECal Module USB to PNA USB](#).
2. Allow the module to warm up until it indicates **READY**.
3. Enter the analyzer settings. See [Set Up Measurements](#).
4. Do one of the following to start the [Calibration Wizard](#)

Using front-panel  
HARDKEY [softkey] buttons

Using a mouse with PNA Menus

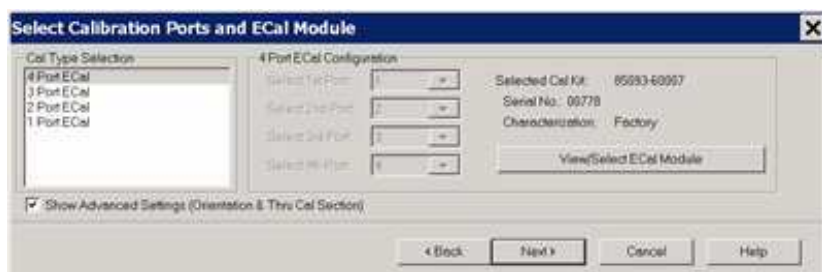
1. Press **CAL**
2. then **[Start Cal]**
3. then **[Cal Wizard]**

1. Click **Response**
2. then **Cal**
3. then **Cal Wizard**

### Programming Commands

5. In the [Calibration Wizard Begin](#) dialog box, click **Use Electronic Cal (ECal)**.

**Note:** To calibrate with more than one ECal module, select [SmartCal](#), then choose the ECal modules as your Cal Kits.



### Select Calibration Ports and ECal Module dialog box help

Allows you to select calibration type and settings.

**Cal Type Selection / Configuration** Select the number of ports to calibrate. Then select the port number configuration.

**4 Port ECal** Available only if using a 4-port PNA. No additional configuration necessary.

**3 Port ECal** Available only if using a 4-port or 3-port PNA.

**2 Port ECal**

**1 Port ECal- (Reflection)** Advanced Settings are not available.

**View/Select ECal Module** Click to [Select the ECal module](#) if more than one ECal module is connected to the PNA. Also, [Select the User Characterization](#) within the module. Learn more about [User Characterization](#).

**Show Advanced Settings** Check to display the [Advanced Settings](#) when **Next** is clicked.

**Back** Return to [Cal Wizard Begin](#) dialog. If checked, you can clear the [Save Preferences](#) checkbox to see the Begin page when the Cal Wizard begins.

**Note:** The PNA no longer allows ECal isolation to be performed. The inherent isolation of the PNA is better than that attained with correction using an ECal module.

Terminate any unused ECal ports with a 50 ohm load.



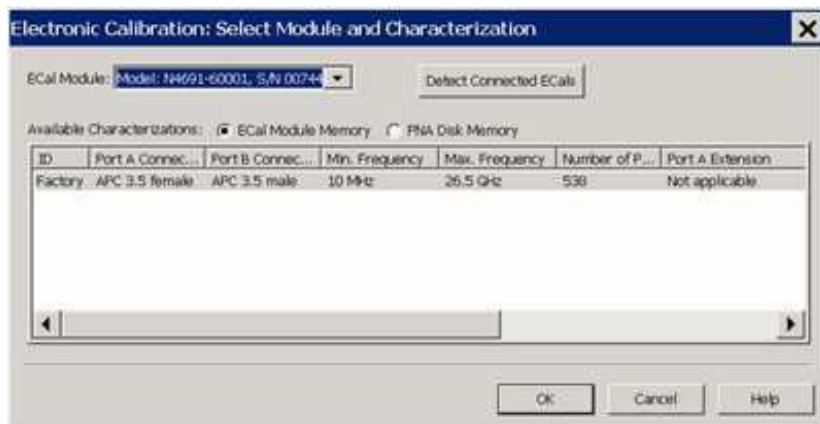
### ECal module not found dialog box help

Displays an error message indicating the ECal module is not connected or has not been recognized by the network analyzer.

**Retry** Check the USB connections and click to continue.

#### Notes:

- If your ECal module is not detected, try to unplug, then reconnect the USB connector to the PNA.
- When the ECal module is connected to the network analyzer for the first time, it may take approximately 30 seconds for the analyzer to recognize the module and make it available for calibration.
- For best accuracy, allow the ECal module to warm-up until it indicates READY.
- Agilent 8509x and N4431 ECal modules, when first connected, draw significantly more current than other modules. This could cause the USB to stop working in certain situations. [See USB limitations.](#)
- See [Connect ECal Module USB to PNA USB.](#)



## Select Module and Characterization dialog box help

**ECal Module** Select one of the ECal modules that are connected to the PNA.

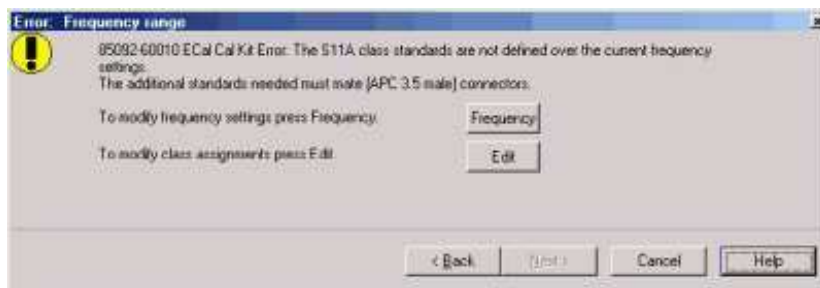
**Detect Connected ECals** Click to rescan the USB for ECal modules.

### Available Characterizations

**ECal Module Memory** - Displays the factory and [user characterizations](#) that are stored in the ECal module.

**PNA Disk Memory** - Displays the user characterizations that are stored in PNA Disk Memory. [Learn more User Characterizations in PNA Disk Memory.](#)

Select either the characterization data to use for the calibration. Once selected, that characterization becomes the default selection until the PNA is turned OFF and restarted. When restarted, **Factory** again becomes the default selection.



## Error: Frequency Range dialog box help

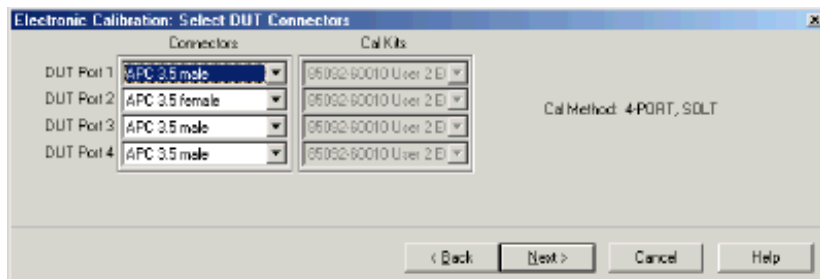
The current cal standards (or ECAL module) does not cover the current frequency range of the measurement. Do one of the following to correct the problem:

**Frequency** Change the frequency range of the active channel.

**Edit** This selection is not relevant to ECal modules.

**Back** Select a different characterization that covers the required frequency range.

**Cancel** Re-characterize the module with an increased frequency range.



## Select DUT Connectors and Cal Kits dialog box help

If the ECal module or selected User Characterization has more than one connector type, then the following dialog box is presented which allows you to describe the DUT connector type. Otherwise, click next to proceed to [Advanced Settings](#) (if checked) or [ECal Steps](#).

### Connectors

The available connectors are listed for each DUT port.



## Advanced Settings dialog box help

### Thru #n

Lists the proposed Thru connections to be made during the calibration process. You can change these Thru connections to better suit your test setup.

- The proposed Thru connections are listed automatically.
- Additional Thru connections can be selected for higher accuracy. [Learn more](#).

### Add Thru

Click to add a Thru connection. [Learn more](#)

### Remove Thru

Select a Thru by clicking the "Thru #N" field or the "1st Port / 2nd Port" field. Then click "Remove Thru". This selection is NOT available if the selected Thru is required for the calibration.

### 1st Port / 2nd Port

Click to change the two ports to be included in the Thru connection. The order of the port numbers (1st or 2nd) is not critical.

### Thru Cal Method

Lists the available Thru Cal methods for the specified port pairs.

[Learn about ECal Thru Methods](#)

### Cal Type/ Stds

Click to invoke the [View / Modify Properties of Cal dialog box](#)

### Do orientation

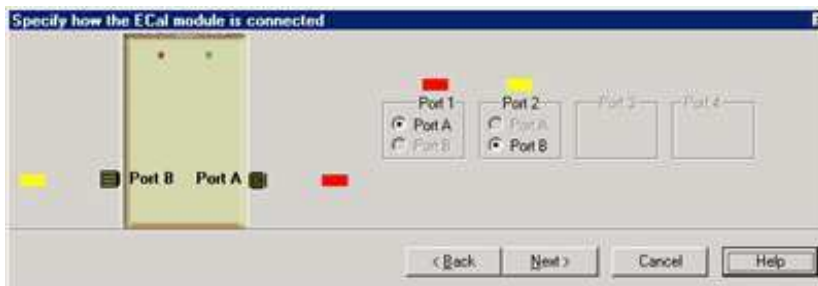
When this box is checked (default) the PNA automatically senses the model and direction in which an ECal module port is connected to the PNA ports. If power to the ECal module is too low, it will appear as if there is no ECal module connected. If you use low power and are having this problem, clear this check box to provide the orientation manually.

Orientation occurs first at the middle of the frequency range that you are calibrating. If a signal is not detected, it tries again at the lowest frequency in the range. If you have an **E8361A** or **E836xB** PNA and do an ECal completely within 10 - 20 MHz OR 60 - 67 GHz, you may need to do orientation manually. There may not be sufficient power to orient the ECal module at those frequencies.

### Choose delta match

Available only when a Delta Match Cal is required.

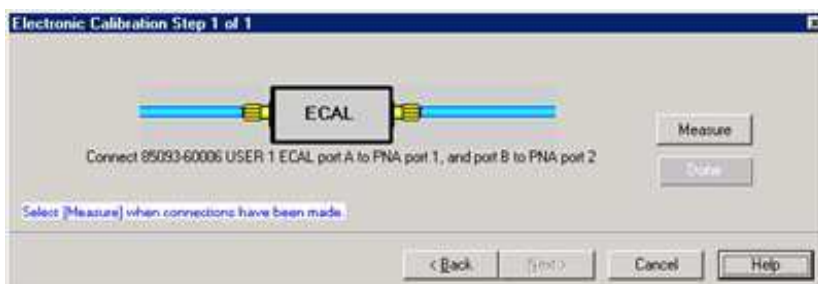
- Check, then click **Next** to invoke the [Select Cal Set for Delta Match](#) dialog box.
- Clear - The Cal Wizard uses the [Global Delta Match Cal](#) if available.



### Specify how the ECal module is connected dialog box help

This dialog box appears when the **Do orientation** checkbox in the previous dialog box is cleared.

Click the ECal Port that is connected to each PNA port.





## Electronic Calibration Steps dialog box help

**Note:** Beginning in PNA Rev. 6.0, ECal can be performed with External triggers. [Learn more.](#)

Displays the instructions for each measurement required for calibration.

**Measure** Measures the ECal standards.

**Done** Click when last standard has been measured.

## Saving an ECal Calibration

When complete, you can save the new calibration. [Learn how.](#)

---

Last modified:

9-Jan-2009	Updated compression level
3-Sep-2008	Removed legacy content
27-Jun-2008	Added compression levels
4-Sep-2007	Added First time note
14-Sep-2007	MX Added UI
Sept. 12, 2006	MQ Modified images for multiport

## ECal User Characterization

---

- [Overview](#)
- [How to Perform a User Characterization](#)
- [Manage Disk Memory](#)
- [Restore ECal Module Memory](#)

See also [Using ECal](#)

### Other [Calibration Topics](#)

## Overview

A user-characterized ECal module allows you to add adapters to the ECal module, re-measure the standards in the ECal module, INCLUDING the adapters, then add that data to ECal memory or save it to the PNA hard drive. This extends the reference plane from the module test ports to the adapters.

### Why perform a user characterization?

- If you need to use adapters with your ECal module, you could characterize your ECal module with the adapters attached and perform subsequent ECals in a single step.
- If you have a 4-port ECal module, you could configure the module with adapters of different connector types, then perform a user characterization of the module. When you need to test a DUT with a pair of the connector types on your module, calibrate the PNA with a 1-step ECal using the same two connectors on the User-characterized module.
- If you test devices in a fixture, you could embed the characterization of the fixture in the characterization of the module. To do this, during the mechanical calibration portion of the user characterization, calibrate at the reference plane of the device as you would normally calibrate. Then remove the fixturing to be embedded and insert the ECal module to be characterized. When measuring the ECal module, the PNA removes the effects of the fixturing and stores the measurement results in the user characterized ECal module. Subsequent calibrations with that user-characterized module will also remove the fixture effects.

### Notes:

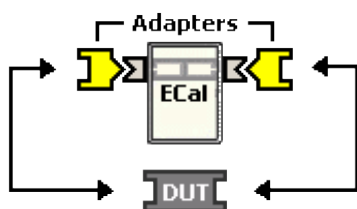
- Both 2-port and 4-port ECal modules support user characterization.
- User Characterization does not delete the factory characterization data. The factory data is saved in the ECal module in addition to the user characterization data.
- Beginning with A.06.03, a User Characterization can be performed beyond the frequency range of the ECal module. Although this practice is allowed, calibration accuracy with the extended user characterization is likely to be degraded. To determine the level of degradation, compare measurements of a variety of devices using a PNA with a mechanical cal kit calibration versus an ECal extended user characterization calibration.

- Beginning with A.08.33, you can save up to 12 User Characterizations in a single ECal module. Previous releases allowed up to 5. There are memory limitations. The PNA will determine if the contents of a user characterization will fit inside the module before it is performed.
- Saving a new User Characterization with PNA releases **before** A.08.33 will erase any User Characterizations greater than 5. This is because the PNA completely rewrites ALL ECal User Characterizations when saving a new one in order to free storage space in the ECal module.
- Beginning with A.08.33, a User Characterization can be performed remotely. [See programming commands.](#)
- Beginning with A.09.00, a User Characterization can be saved to the PNA hard drive. [Learn how.](#)

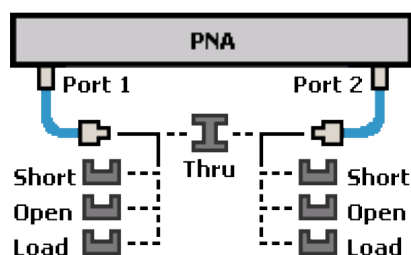
## How to Perform a User Characterization

**SUMMARY** (A [detailed procedure](#) follows.)

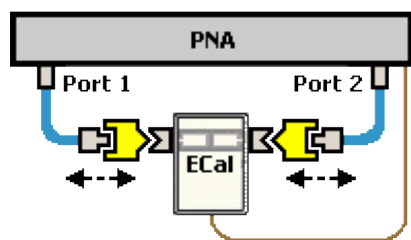
1. Select adapters for the module to match the connector configuration of the DUT.



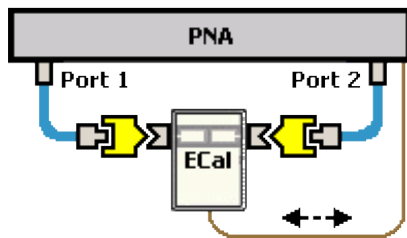
2. Either calibrate the PNA using mechanical standards or recall an existing Cal Set.



3. Measure the ECal module, including adapters, as though it were a DUT.



4. The measurement results are the characterization data that then gets stored inside the module.



#### Note

**A 2-port PNA can be used to perform a User Characterization on a 4-port ECal module.** However, a 4-port ECal module has SIX different port pairs. The PNA must be recalibrated for each port pair that uses unique connector types or gender.

- If all 4 ECal module ports have the same connector type and gender, then only one PNA calibration is required to measure all six port pairs.
- If all 4 ECal module ports have different connector types or gender, then 6 calibrations are required.

When more than one PNA calibration is required during a User Characterization, then ALL calibrations must be performed using the standard Cal Wizard, saved to Cal Sets, and then [recalled from Cal Sets](#) DURING the User Characterization.

#### Detailed steps to Perform a User Characterization

1. Connect the ECal module to the network analyzer with the USB cable. See [Connect ECal Module USB to PNA USB](#).
2. Allow the module to warm up until it indicates **READY**.
3. **Preset** the analyzer.
4. [Set up the measurement](#). For best accuracy, the **IF bandwidth** should be set to **1 kHz** or less.
5. Start and complete the **Characterize ECal Module** Wizard:

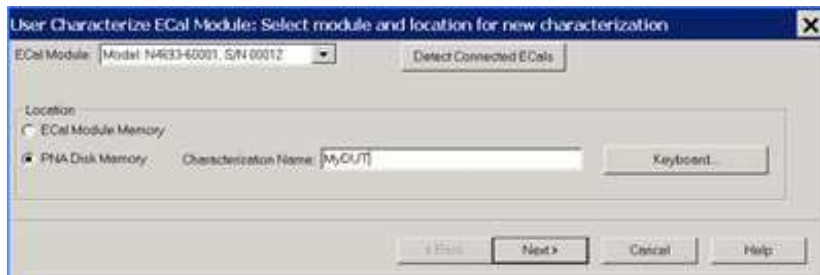
##### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[More]**
3. then **[ECal]**
4. then **[Characterize ECal Module]**

##### Using a mouse with PNA Menus

1. Click **Response**
2. the **Cal**
3. then **More**
4. then **ECal**
5. then **Characterize ECal Module**

## Programming Commands



### Select Module and Location dialog box help

**ECal Module** Select one of the ECal modules that are connected to the PNA.

**Detect Connected ECals** Click to rescan the USB for ECal modules.

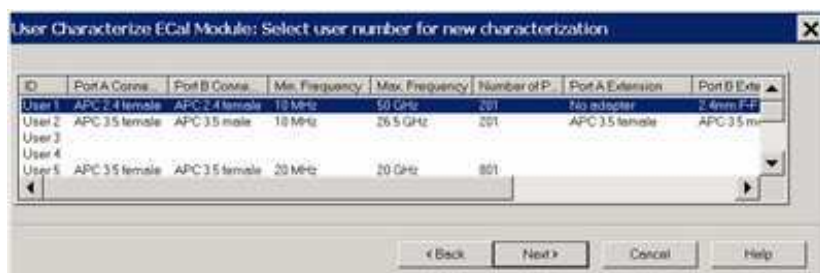
#### Location

- **ECal Module Memory** Click Next to see the following dialog.
- **PNA Disk Memory** Enter a Characterization Name. This name appears when selecting a user characterization to used with subsequent calibrations. [Learn how to manage characterizations that are stored to PNA disk memory.](#)

**Keyboard** Launches a keypad that can be used to type a characterization name from the PNA front panel.

**Next** Click to continue to the [Select Connectors for the Characterization](#) dialog box.

[See note regarding extended frequency use.](#)



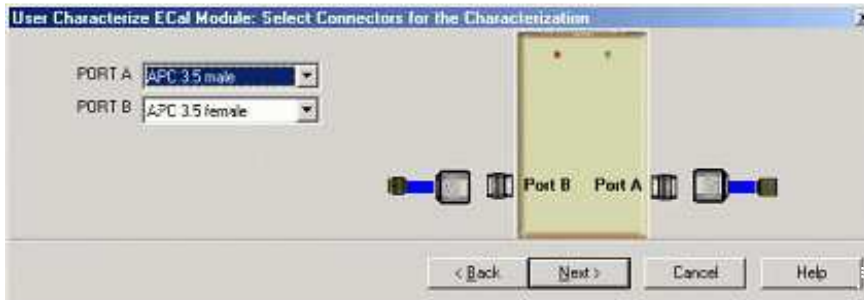
### Select User Number for new characterization dialog box help

Scroll to view all of the parameters of the stored characterizations. Select an empty location or select to overwrite an existing characterization.

Data can NOT be deleted from the ECal module.

**Next** Click to continue to the [Select Connectors for the Characterization](#) dialog box.

[See note regarding extended frequency use.](#)



### Select Connectors for the Characterization dialog box help

**Note:** When performing an ECal User Characterization, do NOT use a [custom connector name](#) that you added to this list. If you need to use a custom-defined connector type, select "Type B", or one of the "Type A" variations from the list of connectors for each port.

Allows you to select the adapters for the ECal module test ports. Select **No adapter** if no adapter is used on a port.

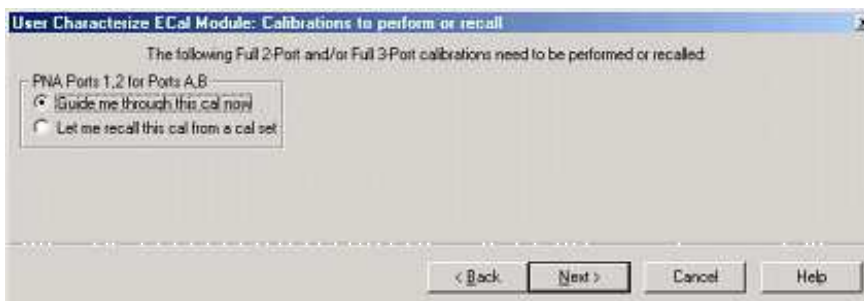
**PORT A** Lists the connector types available for Port A.

**PORT B** Lists the connector types available for Port B.

**PORT C** Lists the connector types available for Port C (available with a 4-port ECal module).

**PORT D** Lists the connector types available for Port D (available with a 4-port ECal module).

**Next** Click to continue to the [Calibrations to perform or recall](#) dialog box.



## Calibrations to perform or recall dialog box help

The PNA must be calibrated before measuring the ECal module and necessary adapters. This dialog box displays the number and types of mechanical calibrations required for the characterization.

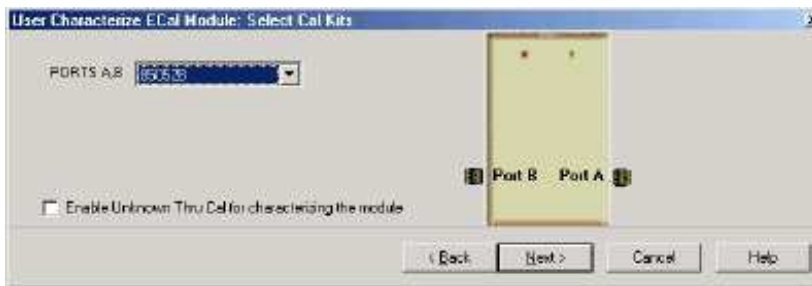
**Guide me through this cal now** Click to perform a Guided calibration. A calibration kit is required for each connector type.

**Note:** TRL calibrations can NOT be performed on a 4-port PNA during the calibration portion of a User Characterization. However, this type of Cal can be performed using the Cal Wizard, saved to a Cal Set, then recalled at this point in the User Characterization.

If more than one calibration is required, this selection is not available. [See Note.](#)

**Let me recall this cal from a cal set** Click to select an existing Cal Set. You cannot select a Cal Set that is currently in use. Learn more about [Using Cal Sets](#).

**Next** Click to continue to either the [Select Cal Kits](#) or the [Select Cal Set](#) dialog box.



## Select Cal Kits dialog box help

Provides a list of calibration kits to perform the calibration. Select the Cal Kit you will use for each port.

**Enable Unknown Thru for characterizing the module** Check to enable. This reduces the number of steps required to characterize the THRU standard. This setting is available only on PNA models with [one reference receiver per test port](#).

**Next** Click to continue to the [Select Cal Set](#) dialog box.



### Select Cal Set dialog box help

The calibration that you perform will be written to a Cal Set. This dialog box allows you to select a Cal Set to overwrite, or to write to a new Cal Set. The current choice is visible below the **Select Cal Set** button.

**Select Cal Set** Click to open the **Select A Cal Set** dialog box.

**Create new Cal Set** Check to create a new Cal Set to store the calibration. Clear to select and overwrite a stored Cal Set.

**Next** Click to continue to the [Guided Calibration Steps](#) dialog box.

**Note:** Remember the Cal Set name for future reference.



### Guided Calibration Steps dialog box help

Instructs you to connect each calibration standard to the measurement port.

**Measure** Click to measure the standard.

**Back** Click to repeat one or more calibration steps.

**Done** Click **after** a standard is re-measured and all measurements for the calibration are complete.

**Next** Click to continue to the next calibration step. (Does **not** measure the standard.)

**Cancel** Exits Calibration Wizard.

The **Specify nominal delay** or **Guided Calibration completed** dialog box appears when the steps are completed.





### Specify nominal delay dialog box help

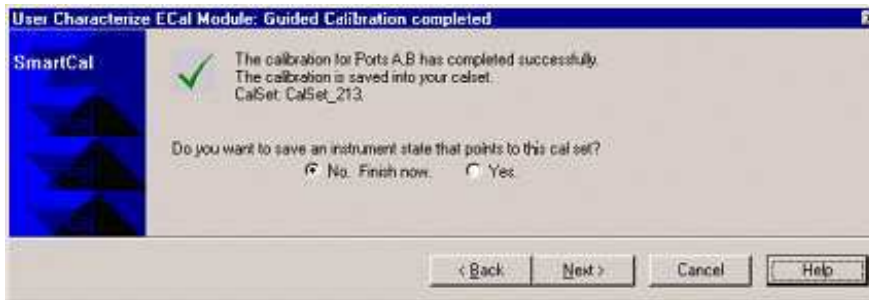
This dialog ONLY appears when [Adapter Removal](#) or [Unknown Thru](#) calibrations are performed.

The following values were estimated from the measurement. Most of the time, they are adequate. However, for CW sweep or frequency sweep with large step sizes, the accuracy of the values may be improved.

**Nominal adapter delay** To improve this value, measure and record the delay of the adapter with a dense step size. Enter that value here.

**Nominal phase offset** (Waveguide ONLY). To improve this value, measure and record the phase offset of the Waveguide adapter with dense step size. Enter that value here.

When one connector is coax and the other connector is waveguide, the phase offset has an ambiguity of 180 degrees. For consistency, the estimate provided here is always between 0 and 180 degrees. You can change this estimate to any value between -180 degrees and +180 degrees.



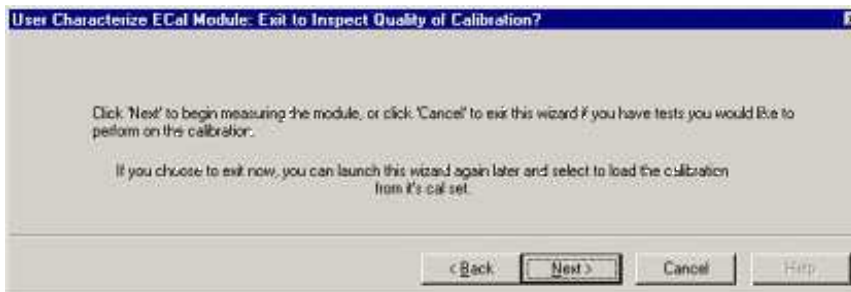
### Guided Calibration completed dialog box help

Allows you to finish the calibration and continue to the next characterization steps.

**No. Finish now** Select to save Cal Set data.

**Yes** Allows selection of Save options.

**Next** Click to continue to the [Exit to Inspect Quality of Calibration](#) dialog box.



### Exit to Inspect Quality of Calibration dialog box help

Allows you to exit User Characterization to [validate the calibration](#) before proceeding with the characterization.

**Back** Allows you to repeat calibration.

**Next** Click to continue to the [Characterization Steps](#) dialog box.

**Cancel** Exits the Calibration.

To return to the current step:

1. Start User Characterization.
2. In the **Select user number for new characterization** dialog box, click **Next**.
3. In the **Select Connectors for Characterization** dialog box, click **Next**. (Previous entry is stored in memory.)
4. In the **Calibrations to perform or recall** dialog box, recall the Cal Set that you just performed.



### Characterization Steps dialog box help

Describes the instructions for each measurement required for characterization.

**Measure** Measures the ECal module.

**Next** Click to continue to the [Information for the New Characterization](#) dialog box when measurements are complete.

### Information for the New Characterization dialog box help

Allows you to describe the properties of the User Characterization.

**Next** Click to continue to the [Write Characterized Data to the ECal module](#) dialog box.

To minimize the number of characters, use the following 3-character codes to describe the connectors listed.

Connector Type	3-Character Code
1.0 mm female	10F
1.0 mm male	10M
1.85 mm female	18F
1.85 mm male	18M
2.4 mm female	24F
2.4 mm male	24M
2.92 mm female	29M
2.92 mm male	29F
3.5 mm female	35F
3.5 mm male	35M
7-16 female	16F
7-16 male	16M
Type F female	F7F
Type F male	F7M
N50 female	N5F
N50 male	N5M

N75 female	N7F
N75 male	N7M
APC 7	7MM
K-band waveguide	KBW
P-band waveguide	PBW
Q-band waveguide	QBW
R-band waveguide	RBW
U-band waveguide	UBW
V-band waveguide	VBW
W-band waveguide	WBW
X-band waveguide	XBW



#### Write Characterized Data dialog box help

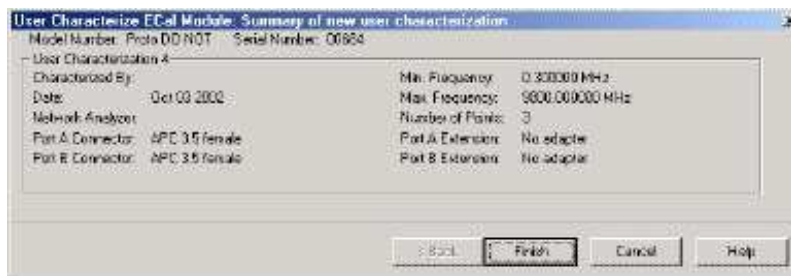
The PNA writes User Characterization and factory characterization data to either the PNA disk memory or the ECal module memory.

For more information, see [Restore ECal module memory](#).

**Write** Click to write data into the ECal module.

The [Summary of new user characterization](#) dialog box opens after data is saved to module.

Data can NOT be deleted from the ECal module.



### Summary of new user characterization dialog box help

Verify the status of the ECal User Characterization.

- ECal module model number
- summary from user characterization

**Cancel** Click to exit (characterization complete).

**Finish** Click to exit (characterization complete).

## Manage ECal User Characterizations in Disk Memory

Normally, user characterizations that are stored in PNA disk memory can be used indefinitely without needing them to be managed. However, this dialog allows you to backup the characterizations in case they are accidentally erased, or to save them to a file that can be moved to another PNA.

### How to Manage ECal User Characterizations in Disk Memory

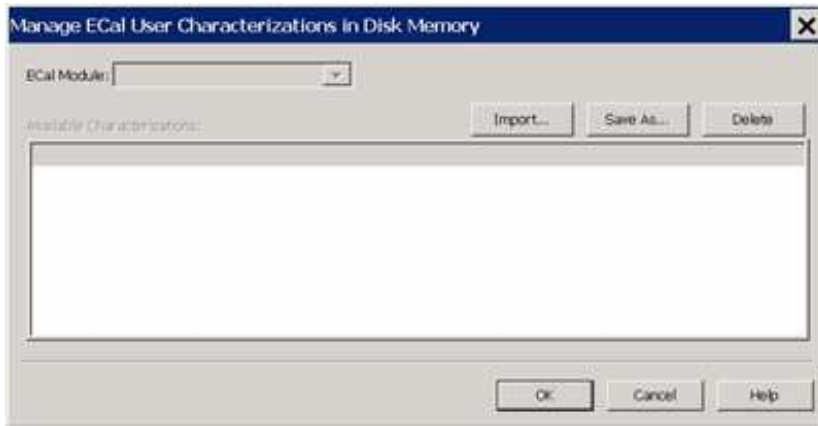
#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[More]**
3. then **[ECal]**
4. then **[Manage ECal Disk Memory]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **More**
4. then **ECal**
5. then **Manage ECal Disk Memory**

◀ **Programming Commands** ▶



#### Manage ECal User Characterizations in Disk Memory dialog box help

This dialog allows you to save user characterizations to files in case they are accidentally erased, or so they can be moved to another PNA.

**ECal Module** Select an ECal Module from the list for which user characterizations are currently stored in PNA disk memory.

**Save As** Saves a User Characterization that is currently in PNA disk memory to a \*.euc file. This file can be used as a backup in case the archive file is accidentally deleted, or allows you to move the file to another PNA to be used with the selected ECal Module.

**Import** Loads a previously saved \*.euc file for use on the PNA with the specified ECal module.

**Delete** Removes a User Characterization from PNA disk memory.

### Restore ECal Module Memory

When user-characterized data is written to the ECal module, the entire contents of ECal memory is also written to the PNA hard drive, including the factory ECal data. In the unlikely event that your ECal module memory is lost, you can restore all ECal data to ECal memory.

**Caution:** If a new factory cal was performed **after** the ECal memory was written to the PNA, the new factory cal data will also be overwritten.

**Note:** An ECal Data Wipe Utility destroys all user data per US DoD 5220.22-M. Learn more at <http://na.tm.agilent.com/pna/apps/applications.htm>

## How to Restore ECal Module Memory

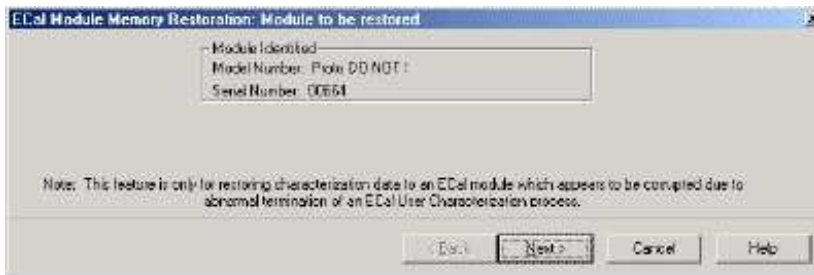
### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Service]**
3. then **[Utilities]**
4. then **[Restore...]**

### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Service**
4. then **Utilities**
5. then **Restore...**

No Programming commands are available for this feature.



### Module to be restored dialog box help

Verify the Serial number of the module to be restored. If two modules are connected to the PNA , choose the one to have data restored.

**Next** Click to write data to the module.

Last modified:

- |              |                                       |
|--------------|---------------------------------------|
| 24-Aug-2009  | Added PNA Disk Memory (9.0)           |
| 3-Nov-2008   | Added 12 chars and remote link (8.33) |
| 3-Sep-2008   | Removed legacy content                |
| 28-Aug-2008  | Clarified note about restore          |
| 13-Aug-2008  | Added note about data wipe            |
| 25-Apr-2007  | Added note about can NOT delete data. |
| 20-Sept-2006 | MX Modified for cross-browser         |

## TRL Calibration

---

TRL (Thru, Reflect, Line) represents a **family** of calibration techniques that measure two transmission standards and one reflection standard to determine the 2-port 12-term error coefficients. For example, **TRM** (Thru, Reflect, Match), **LRL** (Line, Reflect, Line), **LRM** (Line, Reflect, Match) are all included in this family.

The traditional SOLT calibration measures one transmission standard (T) and three reflection standards (SOL) to determine the same error coefficients.

- [Why Perform a TRL Cal?](#)
- [The TRL Calibration Process](#)
- [TRL Cal Kits](#)
- [Cal Standards Used in TRL](#)
- [TRL in 4-port PNA a](#)

[See other Calibration Topics](#)

### Why Perform a TRL Cal?

TRL calibration is extremely accurate, in most cases more accurate than an SOLT cal. However, very few calibration kits contain TRL standards. TRL Cal is most often performed when you require a high level of accuracy and do not have calibration standards in the same connector type as your DUT. This is usually the case when using test fixtures, or making on-wafer measurements with probes. Therefore, in some cases you must construct and [characterize standards](#) in the same media type as your DUT configuration. It is easier to manufacture and characterize three TRL standards than the four SOLT standards.

Another advantage of TRL calibration is that the TRL standards need not be defined as completely and accurately as the SOLT standards. While SOLT standards are completely characterized and stored as the standard definition, TRL standards are modeled, and not completely characterized. However, TRL cal accuracy is directly proportional to the quality and repeatability of the TRL standards. Physical discontinuities, such as bends in the transmission lines and beads in coaxial structures, will degrade the TRL calibration. The connectors must be clean and allow repeatable connections.

To learn more about Cal Standard requirements, see [Cal Standards Used in TRL](#).

### The TRL Cal Process

Although TRL can be performed using the Cal Wizard Unguided Cal selection, the following process uses the easier [SmartCal](#) selection. Both selections require that you already have [TRL calibration standards defined](#) and included in a PNA cal kit.

1. Preset the PNA
2. Set up a S-parameter measurement and the desired stimulus settings.
3. Click **Calibration / Calibration Wizard**



4. Click **SmartCal (Guided Cal)**.
5. [Select the DUT connectors and Cal Kit](#) for each port. The LOWEST port number of each [port pair](#) MUST include TRL standards. TRL appears as the Cal Method.
6. Check **Modify Cal, Next**, then **View/Modify** to change [default TRL options](#) if necessary.
7. Follow the prompts to complete the calibration.
8. [Check the accuracy](#) of the calibration

## TRL Cal Kits

Agilent Technologies offers two cal kits that include the required standards to perform a TRL calibration: 85050C (APC 7mm) and 85052C (3.5mm). Both kits include the traditional Short, Open, and Load standards. (The Thru standard, not actually supplied, assumes a [zero-length Thru](#)). In addition, the kits include an airline which is used as the LINE standard. To use the airline, the kits include an airline body, center conductor, and insertion / extraction tools. The APC 7 kit includes an adapter to connect the airline to the APC connector.

## Cal Standards Used in TRL

These standards must be defined in your TRL cal kit:

### THRU

**Note:** All [THRU calibration methods](#) are supported in a TRL Cal **EXCEPT** Unknown Thru.

- The THRU standard can be either a zero-length or non-zero length. However, a zero-length THRU is more accurate because it has zero loss and no reflections, by definition.
- The THRU standard cannot be the same electrical length as the LINE standard.
- If the insertion phase and electrical length are well-defined, the THRU standard may be used to [set the reference plane](#).
- Characteristic impedance of the THRU and LINE standards defines the reference impedance of the calibration.
- If a THRU standard with the correct connectors is NOT available, an adapter removal cal can be performed.

### REFLECT

- The REFLECT standard can be anything with a high reflection, as long as it is the same when connected to both PNA ports.
- The actual magnitude of the reflection need not be known.
- The phase of the reflection standard must be known within 1/4 wavelength.
- If the magnitude and phase of the reflection standard are well-defined, the standard may be used to [set the reference plane](#).

## LINE

The LINE and THRU standards establish the reference impedance for the measurement after the calibration is completed. TRL calibration is limited by the following restrictions of the LINE standard:

- Must be of the same impedance and propagation constant as the THRU standard.
- The electrical length need only be specified within 1/4 wavelength.
- Cannot be the same length as the THRU standard.
- A TRL cal with broad frequency coverage requires multiple LINE standards. For example, a span from 2 GHz to 26 GHz requires two line standards.
- Must be an appropriate electrical length for the frequency range: at each frequency, the phase difference between the THRU and the LINE should be greater than 20 degrees and less than 160 degrees. This means in practice that a single LINE standard is only usable over an 8:1 frequency range (Frequency Span / Start Frequency). Therefore, for broad frequency coverage, multiple lines are required.
- At low frequencies, the LINE standard can become too long for practical use. The optimal length of the LINE standard is 1/4 wavelength at the geometric mean of the frequency span (square root of  $f_1 \times f_2$ ).

## MATCH

If the LINE standard of appropriate length or loss cannot be fabricated, a MATCH standard may be used instead of the LINE.

- The MATCH standard is a low-reflection termination connected to both Port 1 and Port 2.
- The MATCH standard may be defined as an infinite length transmission line OR as a 1-port low reflect termination, such as a load.
- When defined as an infinite length transmission line, both test ports must be terminated by a MATCH standard at the same time. When defined as a 1-port load standard, the loads are measured separately. The loads are assumed to have the same characteristics.
- The impedance of the MATCH standard becomes the reference impedance for the measurement. For best results, use the same load on both ports. The load may be defined using the data-based definition, the arbitrary impedance definition, or the fixed load definition.

## See Also

- See [Modify Calibration Kits](#) for detailed information about creating and modifying Calibration kit definitions.
- For more information, read [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(Application Note 1287-11\)](#)

## TRL on a 4-port PNA and with an External Test Set

Beginning with the PNA code revision 5.25, TRL CAN be performed on a 4-port PNA and with an [External Test Set](#) enabled. Previously, a TRL calibration required a PNA with a [reference receiver for each test port](#). With the new

TRL method, a [Delta Match Calibration](#) is first performed and applied.

The accuracy of this TRL cal greatly depends on the accuracy of the Delta Match Calibration. With an accurate Delta Match Calibration, the difference in accuracy between a traditional TRL cal and this TRL cal is negligible.

### **How to Perform a TRL Cal on a multiport PNA**

1. Click **Calibration, Cal Wizard**.
2. Select a TRL cal kit for the ports to be calibrated.
3. During the calibration, the Cal Wizard prompts you for a [valid Delta Match Cal](#).

---

Last modified:

14-Apr-2008    Added App Note link

9/12/06    with Ext Test Set

## Calibration Preferences Wizard

---

Two Cal Preferences are set from this Wizard:

1. [Whether or not to show the first 'Method' Page of the Cal Wizard](#)
2. [Select and order the Cal Types that are available during a SmartCal with Mechanical Standards](#)

To change either of these choices, you must select **Yes, Enable the calibration preferences** at the first Wizard page.

### How to change Cal Preferences

Programming commands are NOT available for the preference settings discussed in this topic, although there are other [Cal Preferences](#) that can be set remotely.

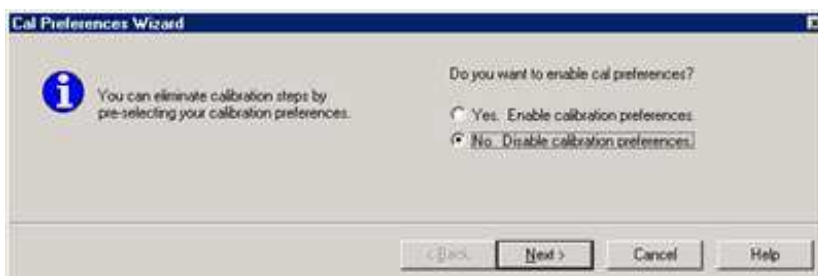
#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Start Cal]**
3. then **[Preferences]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Start Cal**
4. then **Preferences**

#### Programming Commands



### Cal Preferences Wizard dialog box help

Use this dialog to change either of the following preferences:

- Show or Hide the first page of the Cal Wizard
- Select order of calibrations that are offered.

To change either of these choices, you must select **Yes, Enable the calibration preferences**.



### Cal Preferences of... dialog box help

Use this dialog to change which Cal method to perform.

After making this selection, the first page of the Cal Wizard will not be shown on subsequent calibrations.

To change ONLY the order of Cal Types that are offered, and none of these preferences, you must do the following:

1. Select one of these choices and click **Next**.
2. Select and order the Cal Types, then click **Next**
3. Click **Finish**
4. Click **Cal**, then **Cal Wizard**.
5. On the first Cal Wizard page that shows, click **Back**, then clear the **Preferences** checkbox.



## Cal Type Preferences dialog box help

This dialog is used to set which Cal Types are available, and the order in which they are selected as the default choice, during a SmartCal with Mechanical Standards. This setting is also used to set the default Cal Type for Guided calibrations using SCPI or COM.

The specified Cal Type order should allow you to make fewer changes to the Cal Type during a SmartCal with Mechanical Standards.

For example, in the above image, the first Cal Type on the list is TRL. When doing a SmartCal with Mechanical Standards:

- If a TRL Cal Kit is available for the specified DUT connectors, then TRL becomes the default Cal Type.
- If a TRL Cal Kit is NOT available, then the second Cal Type on the list (SOLT) is evaluated for compatibility with the available Cal Kits, and so forth with the Cal Types that remain on the list.
- If TRL is removed from the list, that Cal Type is NOT available for selection during a SmartCal with Mechanical Standards.

[Learn more about Cal Types.](#)

[See where you choose Cal Type during a SmartCal](#)

**Prioritized list of choices for default Cal Type** Shows the current list of Cal Types and the order in which they will be selected for Mechanical calibrations.

**Change** Click to invoke the [Modify list of default Cal Types](#) dialog.

**Restore factory defaults** Returns the list to the original selections and order. The factory defaults are in order of accuracy from highest (TRL) to lowest (QSOLT).

**Cancel** Closes the dialog without making changes.

### Notes:

- Your Cal Type settings are saved only until the PNA application is closed. When re-opened, the factory default settings are restored.

[Learn more about QSOLT Calibration](#)



#### Modify list of default Cal Types dialog box help

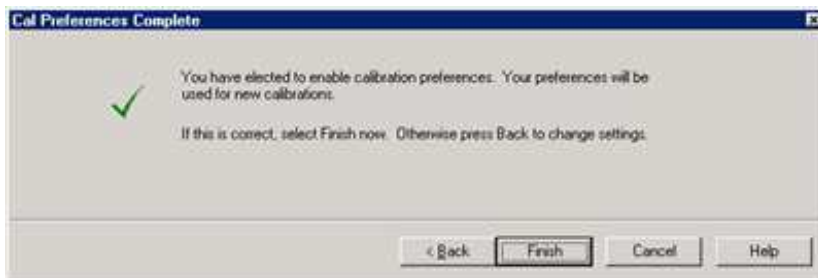
Use this dialog to Add, Remove, and re-order the available Cal Types. There must be at least ONE selected Cal Type to perform a SmartCal with Mechanical Standards.

**Unselected Cal Types** Cal Types in this list will not be presented as a choice during a Calibration.

**Selected Cal Types** Cal Types in this list will be presented, in order, as the default choice during a Calibration. Click a Cal Type to select it, then click the following buttons to perform that operation.

**Add / Remove buttons** Click to Add and Remove the selected Cal Types from the Selected Cal Types list.

**Move Up / Down** Click to re-order the Selected Cal Types list.



#### Cal Preferences Complete dialog box help

Either Enable or Disable Cal Preferences.

See how to [select ONLY Cal Type Preferences](#) and continue to show the first Cal Wizard page.

Last Modified:

3-Sep-2008	Removed legacy content
19-Mar-2008	Small cosmetic edits
1-Jan-2007	MX added UI



## Measurement Errors

---

You can improve accuracy by knowing how errors occur and how to correct for them. This topic discusses the sources of measurement error and how to monitor error terms.

- [Drift Errors](#)
- [Random Errors](#)
- [Systematic Errors](#)
  - [3-Port Error Terms](#)
  - [4-Port Error Terms](#)
- [Monitoring Error Terms](#)

[See other Calibration Topics](#)

### Drift Errors

Drift errors are due to the instrument or test-system performance changing after a calibration has been done.

Drift errors are primarily caused by thermal expansion characteristics of interconnecting cables within the test set and conversion stability of the microwave frequency converter and can be removed by re-calibrating.

The time frame over which a calibration remains accurate is dependent on the rate of drift that the test system undergoes in your test environment.

Providing a stable ambient temperature usually minimizes drift. For more information, see [Measurement Stability](#).

### Random Errors

Random errors are not predictable and cannot be removed through error correction. However, there are things that can be done to minimize their impact on measurement accuracy. The following explains the three main sources of random errors.

#### Instrument Noise Errors

Noise is unwanted electrical disturbances generated in the components of the analyzer. These disturbances include:

- Low level noise due to the broadband noise floor of the receiver.
- High level noise or jitter of the trace data due to the noise floor and the phase noise of the LO source inside the test set.

You can reduce noise errors by doing one or more of the following:

- Increase the [source power](#) to the device being measured - ONLY reduces low-level noise.

- [Narrow the IF bandwidth.](#)
- Apply several measurement [sweep averages.](#)

### Switch Repeatability Errors

Mechanical RF switches are used in the analyzer to switch the source attenuator settings.

Sometimes when mechanical RF switches are activated, the contacts close differently from when they were previously activated. When this occurs, it can adversely affect the accuracy of a measurement.

You can reduce the effects of switch repeatability errors by avoiding switching attenuator settings during a critical measurement.

### Connector Repeatability Errors

Connector wear causes changes in electrical performance. You can reduce connector repeatability errors by practicing good connector care methods. See [Connector Care.](#)

### Systematic Errors

Systematic errors are caused by imperfections in the analyzer and test setup.

- They are repeatable (and therefore predictable), and are assumed to be time invariant.
- They can be characterized during the calibration process and mathematically reduced during measurements.
- They are never completely removed. There are always some residual errors due to limitations in the calibration process. The residual (after measurement calibration) systematic errors result from:
  - imperfections in the calibration standards
  - connector interface
  - interconnecting cables
  - instrumentation

**Reflection** measurements generate the following three systematic errors:

- [Directivity](#)
- [Source Match](#)
- [Frequency Response Reflection Tracking](#)

**Transmission** measurements generate the following three systematic errors:

- [Isolation](#)
- [Load Match](#)
- [Frequency Response Transmission Tracking](#)

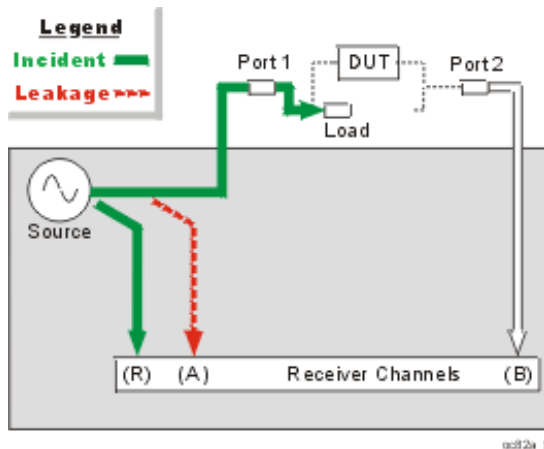
## Notes about the following Systematic Error descriptions:

- The figures for the following six systematic errors show the relevant hardware configured for a forward measurement. For reverse measurements, internal switching in the analyzer makes Port 2 the source and Port 1 the receiver. 'A' becomes the transmitted receiver, 'B' becomes the reflected receiver, and 'R2' becomes the reference receiver. These six systematic errors, times two directions, results in 12 systematic errors for a two port device.
- For simplicity, it may be stated that ONE standard is used to determine each systematic error. In reality, ALL standards are used to determine ALL of the systematic errors.
- The following describes an SOLT calibration. This does not apply to TRL, or other types of calibration.

### Directivity Error

All network analyzers make reflection measurements using directional couplers or bridges.

With an ideal coupler, only the reflected signal from the DUT appears at the 'A' receiver. In reality, a small amount of incident signal leaks through the forward path of the coupler and into the 'A' receiver. This leakage path, and any other path that allows energy to arrive at the 'A' receiver without reflecting off the DUT, contributes to directivity error.



### How the Analyzer Measures and Reduces Directivity Error.

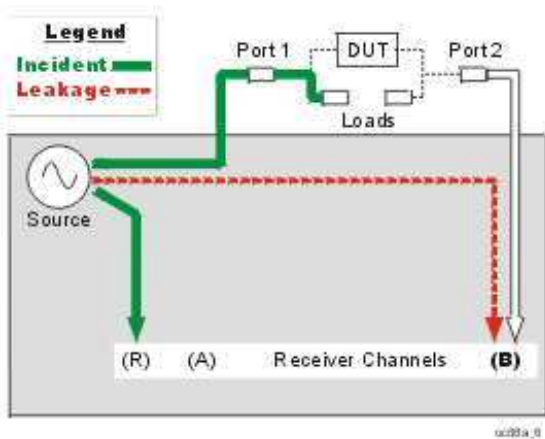
1. During calibration, a load standard is connected to Port 1. We assume no reflections from the load.
2. The signal measured at the 'A' receiver results from the incident signal leakage through the coupler and other paths.
3. Directivity error is mathematically removed from subsequent reflection measurements.

### Isolation Error

Ideally, only signal transmitted through the DUT is measured at the 'B' receiver.

In reality, a small amount of signal leaks into the 'B' receiver through various paths in the analyzer.

The signal leakage, also known as crosstalk, is isolation error which can be characterized and reduced by the analyzer.



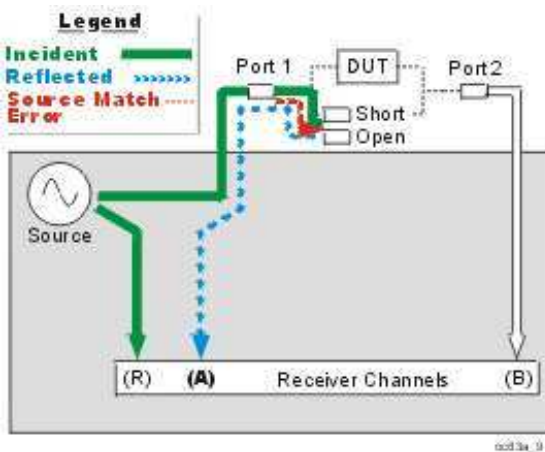
### How the Analyzer Measures and Reduces Isolation Error

1. During calibration, load standards are connected to both Port 1 and Port 2.
2. The signal measured at the 'B' receiver is leakage through various paths in the analyzer.
3. This isolation error is mathematically removed from subsequent transmission measurements.

### Source Match Error

Ideally in reflection measurements, all of the signal that is reflected off of the DUT is measured at the 'A' receiver. In reality, some of the signal reflects off the DUT, and multiple internal reflections occur between the analyzer and the DUT. These reflections combine with the incident signal and are measured at the 'A' receiver, but not at the 'R' receiver.

This measurement error is called source match error which can be characterized and reduced by the analyzer.



### How the Analyzer Measures and Reduces Source Match Error

1. During calibration, all reflection standards are connected to Port 1. Known reflections from the standards are measured at the 'A' receiver.
2. Complex math is used to calculate source match error.

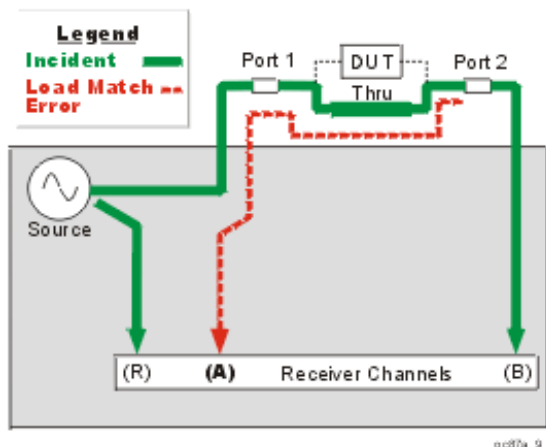
3. Source match error is mathematically removed from subsequent reflection and transmission measurements.

### Load Match Error

Ideally in transmission measurements, an incident signal is transmitted through the DUT and is measured at the 'B' receiver.

In reality, some of the signal is reflected off of Port 2 and other components and is not measured at the 'B' receiver.

This measurement error is called load match error which can be characterized and reduced by the analyzer.



### How the Analyzer Measures and Reduces Load Match Error

1. The Port 1 and Port 2 test connectors are mated together for a perfect zero-length thru connection. If this is not possible, a [characterized thru adapter](#) is inserted. This allows a known amount of incident signal at Port 2.
2. The signal measured at the 'A' receiver is reflection signal off of Port 2
3. The resulting load match error is mathematically removed from subsequent transmission and reflection measurements.

### Frequency Response Reflection Tracking Error

Reflection measurements are made by comparing signal at the 'A' receiver to signal at the 'R1' receiver. This is called a ratio measurement or "A over R1" (A/R1).

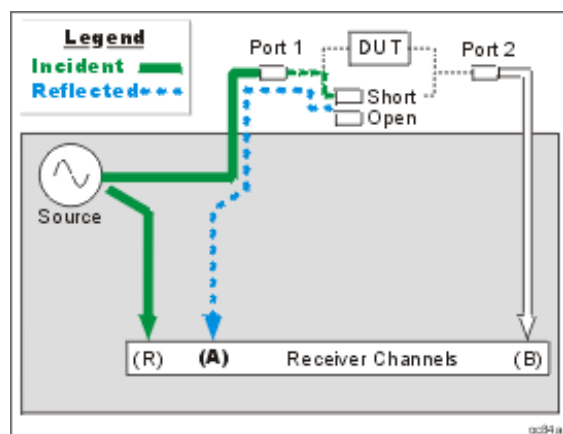
For ideal reflection measurements, the frequency response of the 'A' and 'R1' receivers would be identical.

In reality, they are not, causing a frequency response reflection tracking error. This is the vector sum of all test variations in which magnitude and phase change as a function of frequency. This includes variations contributed by:

- signal-separation devices
- test cables
- adapters

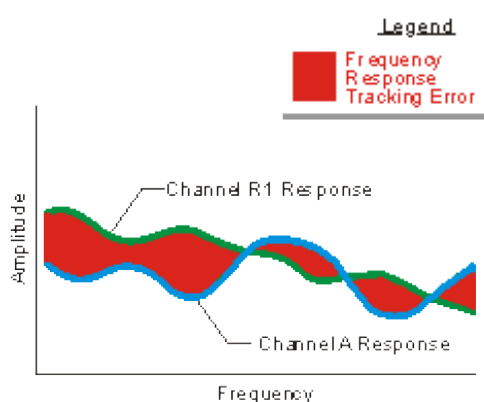
- variations between the reference and test signal paths

Frequency response reflection tracking error can be characterized and reduced by the analyzer.



### How the Analyzer Measures and Reduces Frequency Response Reflection Tracking Error.

1. During calibration, all reflection standards are used to determine reflection tracking.
2. The average 'A' receiver response is compared with the 'R1' receiver response.
3. Complex math is used to calculate Frequency Response Reflection Tracking Error (see the following diagram). This frequency response reflection tracking error is mathematically removed from subsequent DUT measurements.



**Note:** In reflection response calibrations, only a single calibration standard is measured (open or short) and thus only its contribution to the error correction is used.

### Frequency Response Transmission Tracking Error

Transmission measurements are made by comparing signal at the 'B' receiver to signal at the 'R1' receiver. This is called a ratio measurement or "B over R1" (B/R1).

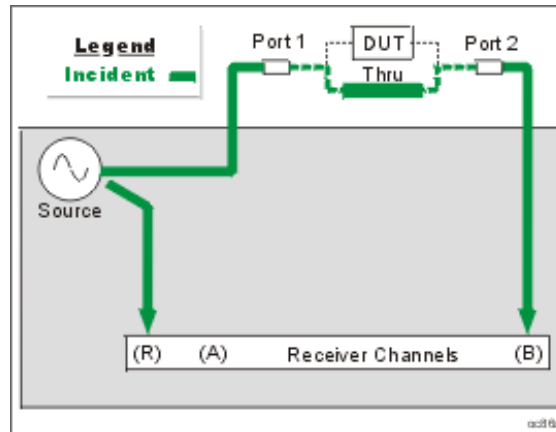
For ideal transmission measurements, the frequency response of the 'B' and 'R1' receivers would be identical.

In reality, they are not, causing a frequency response transmission tracking error. This is the vector sum of all test variations in which magnitude and phase change as a function of frequency. This includes variations contributed

by:

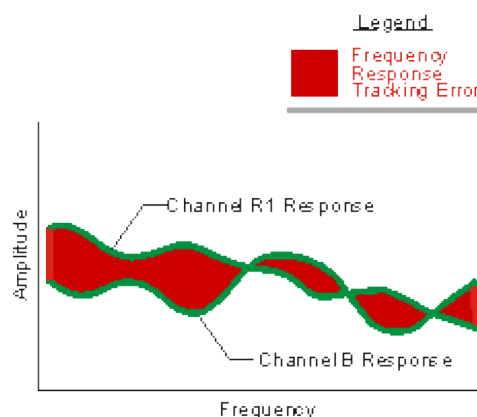
- signal-separation devices
- test cables
- adapters
- variations between the reference and test signal paths

Frequency response transmission tracking error can be characterized and reduced by the analyzer.



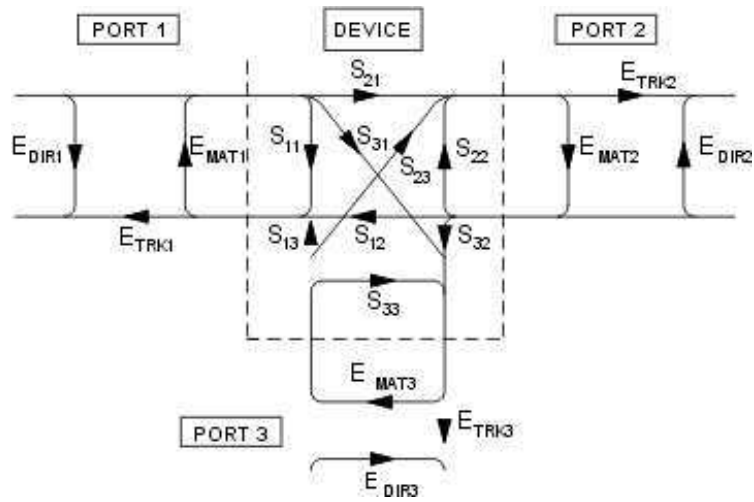
#### How the Analyzer Measures and Reduces Frequency Response Transmission Tracking Error.

1. During calibration, the Port 1 and Port 2 test connectors are mated together for a perfect zero-length thru connection. If this is not possible, a [characterized thru adapter](#) is inserted. This allows a known amount of incident signal to reach Port 2.
2. Measurements are made at the 'B' and 'R1' receivers.
3. Complex math is used to calculate Frequency Response Transmission Tracking Error (see the following diagram). This frequency response transmission tracking error is mathematically removed from subsequent **DUT** measurements.



3-Port Error Terms

The following flow diagram displays the 3-port error term model:



where:

- E = error term
- DIR = Directivity
- MAT = Forward Source Match and Reverse Load Match
- TRK = Forward Reflection Tracking and Reverse Transmission Tracking

4-Port error terms

A full 4-port calibration requires the following terms:

[Learn about the port numbering convention](#) for error terms.

		Source Port			
		1	2	3	4
R e c p	1	DIR 1,1 RTRK 1,1 SRM 1,1	LDM 1,2 TTRK 1,2 XTLK 1,2	LDM 1,3 TTRK 1,3 XTLK 1,3	LDM 1,4 TTRK 1,4 XTLK 1,4
	2	LDM 2,1 TTRK 2,1 XTLK 2,1	DIR 2,2 RTRK 2,2 SRM 2,2	LDM 2,3 TTRK 2,3 XTLK 2,3	LDM 2,4 TTRK 2,4 XTLK 2,4
		LDM 3,1	LDM 3,2	DIR 3,3	LDM 3,4



o r t	3	TTRK 3,1 XTLK 3,1	TTRK 3,2 XTLK 3,2	RTRK 3,3 SRM 3,3	TTRK 3,4 XTLK 3,4
	4	LDM 4,1 TTRK 4,1 XTLK 4,1	LDM 4,2 TTRK 4,2 XTLK 4,2	LDM 4,3 TTRK 4,3 XTLK 4,3	DIR 4,4 RTRK 4,4 SRM 4,4

#### Reflection terms

- DIR: Directivity
- RTRK: Reflection Tracking
- SRM: Source Match

#### Transmission terms

- LDM: Load Match
- TTRK: Transmission Tracking
- XTLK: Cross Talk

#### How can we measure only 3 THRU connections?

On a 4-port PNA, a full 4-port cal can be performed while measuring only 3 THRU connections. Measuring more than 3 THRU connections will give higher accuracy.

By measuring all of the reflection terms, and 3 transmission THRU connections, there is adequate information available to calculate the remaining transmission terms. The following is a high level explanation of the concept. The actual calculations are much more complex.

To simplify, let's substitute letters (A,B,C,D) for port numbers from the diagram above so that they can be combined without confusion. Also for simplicity, let's assume that the source match and directivity errors are zero.

	A	B	C	D
A	AA	AB	AC	AD
B	BA	BB	BC	BD
C	CA	CB	CC	CD
D	DA	DB	DC	DD

- The reflection errors are all measured (AA, BB, CC, DD).
- Lets assume we measure a THRU between ports AB, AC, AD. The reverse direction for these THRUs are

also measured at the same time (BA, CA, DA).

- The terms left to calculate are BC, CB, BD, DB, CD, DC.

The following shows how the BC term is calculated from BA and AC:

$$\frac{BA * AC}{AA} = \frac{B * \cancel{AA} * C}{\cancel{AA}} = BC$$

Similarly:

- CB is calculated from CA and AB
- BD is calculated from BA and AD
- DB is calculated from AB and DA
- CD is calculated from CA and AD
- DC is calculated from DA and AC

## Monitoring Error Terms using Cal Set Viewer

You can use **Cal Set Viewer** to monitor the measured data and the calculated error term. This will help to determine the health of your PNA and the accuracy of your measurements.

By printing or saving the error terms, you can periodically compare current error terms with previously recorded error terms that have been generated by the same PNA, measurement setup, and calibration kit. If previously generated values are not available, refer to Typical Error Term Data in Appendix A, "Error Terms", of the Service Guide.

**Note:** The service guide for your PNA is available at <http://www.agilent.com/find/pna>  
It is also on the CDROM that was shipped with your PNA.

- A stable system should generate repeatable error terms over about six months.
- A sudden shift in error terms over the same frequency range, power, and receiver settings, may indicate the need for troubleshooting system components. For information on troubleshooting error terms, see Appendix A, "Error Terms", of the Service Guide.
- A subtle, long-term shift in error terms often reflects drift or connector and cable wear. The cure is often as simple as cleaning and gauging connectors or inspecting cables.

## Viewing Cal Set Data

- Existing measurement traces are unaffected by the Cal Set Viewer.
- The Cal Set data trace is presented in the highest unused channel number (usually 32) in the active window.
- The Cal Set data trace is labeled as S11 in the status bar regardless of the type of error term or standard.
- Only one Cal Set error term or standard data can be viewed at a time. However, a data trace can be stored into memory and then compared to other data traces.

## How to access Cal Set Viewer

### Using front-panel HARDKEY [softkey] buttons

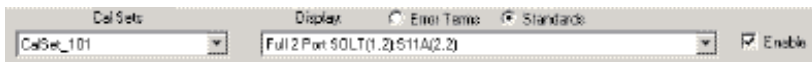
1. Press **CAL**
2. then **[Manage Cals]**
3. then **[Cal Set Viewer]**

### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Manage Cals**
4. then **Cal Set Viewer**

## Programming Commands

## How to use Cal Set Viewer



1. Use the down arrow to select a Cal Set. Then click either:
  - **Error Terms** - calculated data.
  - **Standards** - the raw measurement data of the Standard. **ONLY** available with Unguided Cal (not ECal or Guided Cal).
2. Use the down arrow to select an error term or standard to view.
3. Select the **Enable** check box to view the data on the PNA screen.

**Port numbering convention** for error terms is the same as for S-Parameters:

**E Term (Receiver, Source)** with the following exceptions:

- Load Match (2,1) - The match of port 2 which is measured by making an S11 measurement.
- Load Match (1,2) - The match of port 1 which is measured by making an S22 measurement.
- Transmission Tracking (2,1) - The port 2 receiver relative to the port 1 reference. (source=port 1).
- Transmission Tracking (1,2) - The port 1 receiver relative to the port 2 reference. (source=port 2).
- And so forth for multiport calibrations.

---

Last modified:

3-Sep-2008    Removed legacy content

## Modify Calibration Kits

---

You can create or modify calibration kit files using Advanced Modify Cal Kits.

- [About Modifying Calibration Kits](#)
- [Creating a New Cal Kit from an Existing Cal Kit](#)
- [Creating Custom Calibration Kits using a New Connector Family](#)
- [How to Modify Cal Kits](#)
- [Calibration Class Assignments](#)
- [Waveguide Cal Kits](#)

**Note:** For a detailed discussion of Cal Kits and standards, read [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(Application Note 1287-11\)](#)

[See other Calibration Topics](#)

### About Modifying Calibration Kits

You can modify calibration kit files or create a custom one.

**Note:** You CAN modify Data-based Cal Kits. [Learn how.](#)

For most applications, the default calibration kit models provide sufficient accuracy for your calibration. However, several situations exist that may require you to create a custom calibration kit:

- Using a connector interface different from those used in the predefined calibration kit models.
- Using standards (or combinations of standards) that are different from the predefined calibration kits. For example, using three offset SHORTs instead of an OPEN, SHORT, and LOAD to perform a 1-port calibration.
- Improving the accuracy of the models for predefined kits. When the model describes the actual performance of the standard, the calibration is more accurate. (Example: A 7 mm LOAD is determined to be 50.4 instead of 50.0 .)
- Modifying the THRU definition when performing a calibration for a non-insertable device.
- Performing a [TRL calibration](#).

### Creating a New Cal Kit from an Existing Cal Kit

You can create a new custom Cal Kit using a copy of an existing Cal Kit as a starting point. Here is how:

1. From the [Edit PNA Cal Kits](#) dialog, click **Import Kit** to load the Cal Kit you want to use as a starting point. A "Duplicate Name..." message appears. Click **OK** to load a duplicate copy of the Cal kit into the last position of

the Edit PNA Cal Kits dialog.

2. Select the imported kit.
3. Click **Edit Kit**, then change the [Cal Kit Name and Description](#).
4. Click **Installed Kits - Save As** to save the new Cal Kit to a .ckt file.
5. Recommended: Also click **Edit PNA Cal Kits - Save As** to save the entire collection of Cal Kits to a .wks file.
6. If using a new or modified connector, click [Change Family](#) to change the connector family.
7. Click **Add or Edit** to change connector descriptions and parameters.
8. Make modifications to your new custom Cal Kit as required. Save your work by clicking **Installed Kits - Save As**

**Note:** Custom Cal Kits must be imported after a firmware upgrade. [Learn more](#)

### Creating Custom Calibration Kits using a New Connector Family

To create a custom calibration kit that uses a new connector type, you must first define the connector family. The connector family is the name of the connector-type of the calibration kit, such as:

- APC7
- 2.4 mm
- Type-N (50 )

Although more than one connector family is allowed, it is best to limit each calibration kit to only one connector family.

If you are using a connector family that has male and female connectors, include definitions of both genders. If you are using a family with no gender, such as APC7, only one connector definition is required.

Use the following steps to create a custom calibration kit:

1. In the [Edit PNA Cal Kits dialog box](#), click **Insert New** to add the new connector family.
2. In the [Edit Kit dialog box](#):
  - Type the Kit Description for the custom cal kit.
  - Click **Add** in the Connectors section of the dialog box.
3. In the [Add Connector dialog box](#):
  - Type a Connector Family name.
  - Type a Description of the connector.
  - Select the Gender of one of the connectors.

- Type the minimum and maximum Frequency Range.
  - Type the Impedance.
  - Click the down-arrow to select the Media.
  - Type the cut-off frequency.
  - Click **Apply**.
  - Click **OK**.
4. If you need to add another connector gender, in the [Edit Kit dialog box](#) :
    - Click **Add** in the Connectors section again for the next connector gender.
  5. If you are adding another connector gender, repeat step 3.

**Note:** If you have male and female versions of the connector family, you probably do NOT also have a NO GENDER version.

6. Now that the connector family is added to the custom cal kit, you are ready to add new calibration standards. In the [Edit Kit dialog box](#):
  - Under the list of standards, click **Add**.
7. In the [Add Standard dialog box](#):
  - Select the type of standard (OPEN, SHORT, LOAD, or THRU).
  - Click **OK**.
8. In the [Edit/Add Standards dialog box](#):
  - Complete the information in the dialog box for the standard you selected. Note that for banded standards, the start and stop frequency may be different than the frequency range of the specified connector. Edit the start and stop frequencies as needed.
  - Click **OK** when all the settings are correct.
9. Repeat steps 6 - 8, as necessary, to add all standards and definitions to the new custom cal kit.
10. Assign each of the standards to a calibration class. This is done through the [Modify Calibration Class Assignment](#) dialog box.
11. Click **File, PrintToFile**. PrintToFile will generate a .prn file (ascii file with comma delimiters) that can be imported into a spreadsheet.
12. Import the .prn file into an application such as Microsoft Excel, and print the results.

13. Use the spreadsheet to verify that each standard in the kit belongs to the same connector family and the gender of each standard is properly specified. It is important that the connectors and genders for your standards are correctly defined and verified in order for your SmartCal (guided calibrations) to work properly.

### How to Modify Cal Kits

The series of dialog boxes that follow allow you to modify the standard definitions or class assignments of calibration kit files.

#### Using front-panel HARDKEY [softkey] buttons

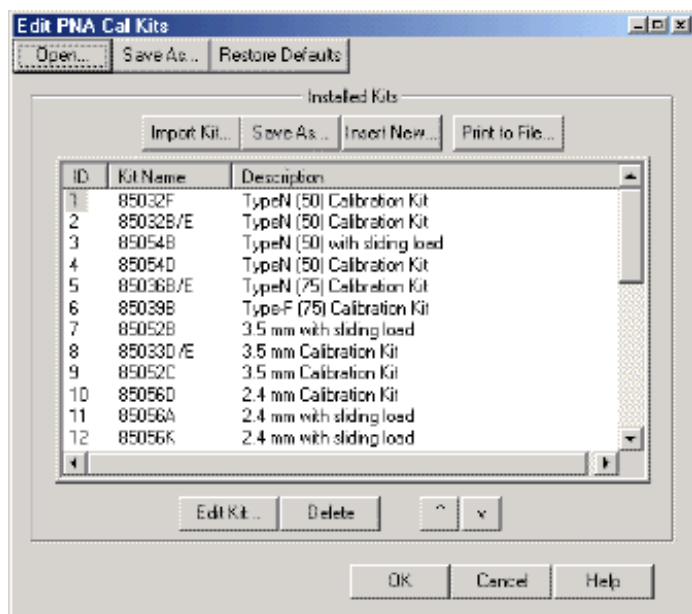
1. Press **CAL**
2. then **[More]**
3. then **[Cal Kit]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **More**
4. then **Cal Kit**

◀ Programming Commands ▶

### Edit PNA Cal Kits





## Edit PNA Cal Kits dialog box help

Provides access to all Agilent cal kits and allows modification of their standard definitions.

### PNA Cal Kits and Firmware Upgrades

- The default "factory" cal kits are overwritten when new firmware is installed. Your custom cal kits (files with custom filenames) are NOT overwritten. However, the custom cal kits must be imported (click **Import Kit**) into the new firmware.
- All PNA cal kits can only be imported by the current firmware revision and later. They can NOT be imported by PAST firmware revisions. Once a Cal Kit has been imported by a later firmware revision, it cannot be imported by the previous version of firmware from which it originated.
- When a firmware upgrade takes place, ALL cal kits, both factory and custom, that are present on the PNA are saved to a single \*.wks file using a unique filename. These files are NOT Excel spreadsheet files. They are opened using the **Open** button (see below). They can be used as archives of cal kits from previous firmware versions.

**Open** Opens an archive of cal kits from past firmware upgrades and 'Save As' operations.

**Save As** Saves ALL cal kits in the PNA to a \*.wks file.

**Restore Defaults** Re-installs the default factory contents of all Agilent cal kits from the PNA hard drive. The factory Agilent cal kits are stored on the PNA hard drive at C:/Program Files/Agilent/Network Analyzer/PnaCalKits/factory.

### Installed Kits

**Import Kit** Invokes the [Import Kit](#) dialog box.

**Save As** Saves the selected calibration kit and definitions (using .ckt file type).

**Insert New** Invokes a blank [Edit Kit dialog box](#) to create new calibration kit definitions.

**Print to File** Prints the contents of the selected cal kit to a .prn file.

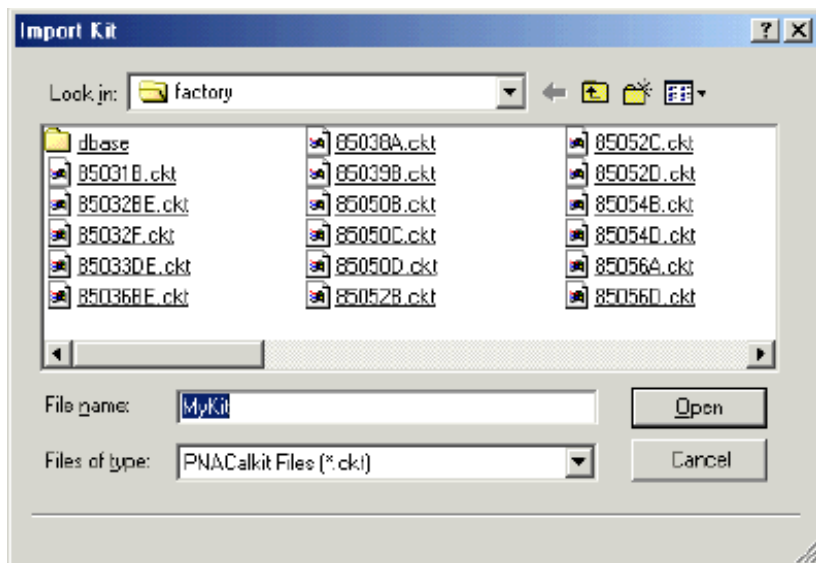
**Edit Kit...** Invokes the [Edit Kit dialog box](#) to modify selected calibration kit definitions.

**Note:** You CAN NOW modify Data-based Cal Kits. [Learn more.](#)

**Delete** Deletes selected calibration kit file.

^ Selects previous / next calibration kit in list.

For more information see [Creating Custom Calibration Kits using a New Connector Family](#).



### Import Kit dialog box help

**Note:** No more than 50 cal kits can be imported.

Imports calibration kit definitions from hard disk or other drive that are saved in the various formats. With PNA version 4.0 or later, four kit types can be imported.

**Note:** See [PNA Cal Kits and Firmware Upgrades](#)

**Files of type** Select the file type of your Cal Kit

Cal Kit Format	File Type
Current PNA Series Cal Kit	*.ckt
Old PNA Series Cal Kit (Version 1)	*.ck1
8510 Cal Kit	CK_*
8753, 8752, 8719, 8720, or 8722 Cal Kit	*.ck

**File name** Navigate and select your cal kit file.

**Open** Imports the selected file. The kit is added at the end of the list of cal kits.

### Importing Kits other than current PNA Series Kits

Cal kit files from Agilent "legacy" network analyzers (listed above) may not contain information that the PNA requires. Therefore, the PNA may modify the cal kit name and description, the cal standards, and the cal class assignments in a best effort manner. You may need to correct these modifications after importing your legacy cal kit to meet your specific requirements.

- "Legacy" cal kit files are based on the analyzer test port sex; PNA cal kits are based on the Device Under Test (DUT) connector sex. Therefore, when the kit is imported the standard's label and description are reversed and are noted as F- (female) and M- (male) .

- When a Coaxial standard is detected in the kit file, a pair of male/female connectors is typically created.
- Waveguide standards that are created as connector have no gender.

**Edit Kit**

**Identification**

Kit Number: 1      Kit Name: 850528

Kit Description: 3.5 mm with sliding load

**Connectors**

Description: APC 3.5 male      Add or Edit...

Family: APC 3.5      Change Family...

**Class Assignments**

SOLT      Edit...

ID	Standard	Description
2	OPEN -M-	3.5 mm male open
15	OPEN -F-	3.5 mm female open
1	SHORT -M-	3.5 mm male short
7	3.5 mm female s...	3.5 mm female short
6	LOWBAND LOA...	3.5 mm male lowband load
5	SLIDING LOAD ...	3.5 mm male sliding load
3	BROADBAND L...	3.5 mm male broadband load

Add...      Edit...      Delete      Delete All

OK      Cancel      Help

## Edit Kit dialog box help

### Identification

**Kit Number** Number of the selected calibration kit.

**Kit Name** Allows you to change the Name of the selected calibration kit.

**Kit Description** Allows you to change the description of the selected calibration kit.

### Connectors

**Note:** You can NOT use a connector with a new or modified name to perform an [ECal User Characterization](#).

The following is the list of Factory-defined connector type strings:

APC 3.5 female	Type N (50) female	7-16 female	X-band waveguide
APC 3.5 male	Type N (50) male	7-16 male	P-band waveguide
APC 2.4 female	Type N (75) female	2.92 mm female	K-band waveguide
APC 2.4 male	Type N (75) male	2.92 mm male	Q-band waveguide
APC 7	Type F (75) female	1.85 mm female	R-band waveguide
	Type F (75) male	1.85 mm male	U-band waveguide
	Type A (50) female	1.0 mm female	V-band waveguide
	Type A (50) male	1.0 mm male	W-band waveguide
	Type B		

Click the down arrow to change the connector type.

**Add or Edit** Invokes the [Add or Edit Connector](#) dialog box which allows you to add new connector type to the calibration kit or edit the connector properties.

**Change Family** Invokes the [Change Connector Family](#) dialog box which allows you to rename the entire connector family name.

### Class Assignments

Click the down arrow to change the Class Assignment.

**Edit** Invokes the [Modify Calibration Class Assignments](#) dialog box.

### Standards in Kit

Lists the current standards and descriptions in the cal kit.

**Add...** Invokes the [Add Standard](#) dialog box that allows you to add definitions for a standard.

**Edit...** Invokes the [Edit dialog box](#) that allows you to modify standard definitions for the selected standard: either Open, Short, Load, or Thru.

**Delete** Deletes selected standard from calibration kit.

## Add or Edit Connector dialog box help

### Identification

**Note:** You can NOT use a connector with a new or modified name to perform an [ECal User Characterization](#).

**Connector Family** Allows you to Add or Edit a specific connector name. If you change Connector Family to a unique name, the name and selected Gender is ADDED to the list of connectors in that kit.

**Note:** To change the Connector Family Name of all connectors in the Kit, click [Change Family](#) on the previous dialog box.

**Description** Displays connector type and gender.

### Frequency Range

**Min** Allows you to define the lowest frequency at which the standard is used for calibration.

**Max** Allows you to define the highest frequency at which the standard is used for calibration.

### Gender

Allows you to define the connector gender.

### Impedance

Allows you to define the impedance of the standard.

### Media

Allows you to define the medium (or 'geometry') of the connector: COAX or WAVEGUIDE.

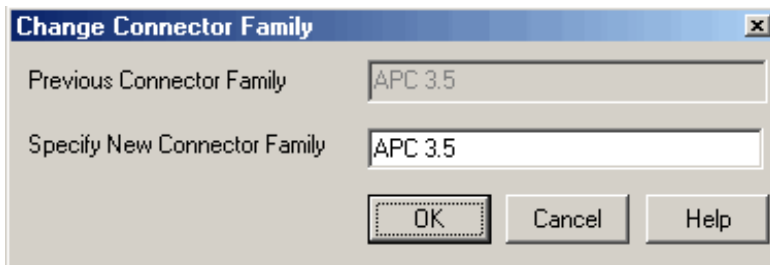
### Waveguide Cal Kits

If modifying or creating a waveguide cal kit, be sure to make the following settings. You can [create a custom waveguide cal kit](#) using an existing factory waveguide Cal kit as a starting point. The factory cal kits already have these settings.

- Frequency Range: **Min. frequency = Cutoff frequency.**
- Gender: **No Gender**
- Impedance Z0: **1 ohm**
- Media: **Waveguide**
- **Cutoff Frequency** enter the low-end cutoff frequency.
- **Height/Width Ratio** Used to calculate waveguide loss. This value is usually on the data sheet for waveguide devices. For more information see [App Note 1287-11](#).

Other waveguide settings

- If performing an [Unguided Cal](#), change [System Impedance](#) to 1 ohm.
- For waveguide, choose [TRL \(Thru-Reflect-Line\)](#) calibration type . These calibration types are more accurate and take fewer steps than SOLT.



#### Change Connector Family dialog box help

**Note:** You can NOT use a connector with a new or modified name to perform an [ECal User Characterization](#).

Performs a text "Search and Replace" function. Within the description field of each of the standards of the current Cal Kit, it searches for the Previous Connector Name and replaces it with the New Connector Name.

**Specify New Connector Name** Allows you to replace the primary connector-family name from the selected kit with the new connector-family name. The PNA allows multiple connector-families per kit.

**Previous Connector Name** Displays the primary connector-family name. All occurrences of the previous connector name will be replaced throughout calibration dialog boxes. This includes calibration kit labels and description fields.

#### Notes:

- String replacement requires an exact match and is case sensitive. For example, "Type N" does not match "type N", and "apc 7" does not match "APC 7".
- Some calibration kits may include connector names that do not match strings within labels or description fields. You may reuse the Change Connector Name dialog to standardize the name within the kit, and then to replace the standard name with the new name.

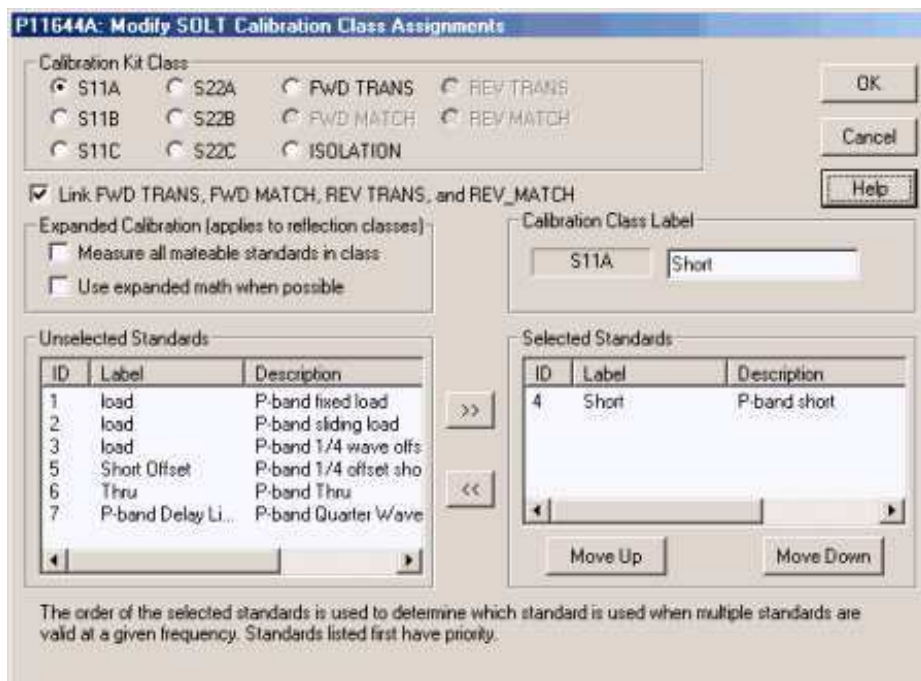
#### Example:

Select the 85056A calibration kit. The default connector-family name is "APC 2.4". However, many

standard description files are labeled "2.4 mm". You may want to replace the connector family name with a new name and update the standard descriptions to match the new name. For this kit, use a two step procedure.

1. Use the Change Connector Name dialog to replace "APC 2.4" with "2.4 mm".
2. Use the Change Connector Name dialog to replace "2.4 mm" with the new name, "PSC 2.4 mm".

See Also [Creating a New Cal Kit from an existing Cal Kit](#)



### Modify Calibration Class Assignments dialog box help

Allows you to assign single or multiple standards to Calibration Classes.

There are two ways to get here:

1. Click **Calibration**
2. Click **Advanced Modify Cal Kit..**
3. Select the Cal Kit, then click **Edit Kit**
4. Under Class Assignments, select the Cal Method (SOLT, TRL), then click **Edit**

You can also get here during a [SmartCal Calibration](#).

1. From the [Select DUT Connectors and Cal Kits dialog](#), check **Modify Cal**, then click **Next**.

2. At the Modify Cal dialog, click a **Mod Stds** button.
3. At the View/Modify Properties Dialog, select the Cal Method (SOLT, TRL), then click **View/Modify**

To assign a standard to a calibration class:

1. Select the **Calibration Kit Class**
2. Select the standard from the **Unselected Standards** field
3. Click the right arrow to move the standard to the **Selected Standards** field.

#### Notes:

- During an Unguided Cal all of the **Selected Standards** are presented. You then choose which of these standards to measure.
- The MATCH standards must be assigned to the FWD MATCH, REV MATCH, and LINE classes. See [TRL calibrations](#) to learn more about TRL standards.
- Use MOVE UP and MOVE DOWN to change the **ORDER** of the standard. The order is used during a [SmartCal](#) to determine overlap priorities when:
  - **Multiple standards are valid for a frequency** - standards are presented in the order in which they appear.
  - **Using two sets of standards** - modify the order in which standards appear to reflect the configuration of your DUT. For example, for a DUT with a male connector on port 1 and a female connector on port 2, order the devices within the S11 classes (A, B, and C) such that the MALE standards are first in the list. Then order the S22 classes specifying the FEMALE standards as the first in the list.

To Add or Edit standards, click Calibration then, click [Advanced Modify Cal Kit](#).

- See [TRL Class Assignments](#)
- [Learn more about Calibration Classes.](#)

#### Calibration Class Label

The label that appears on the Unguided Cal - Measure Mechanical Standards dialog box. For example, the Calibration Class Label "**Modified OPEN**" would yield the following prompt:





The following selections in this dialog box depend on your Class Assignment selection (**SOLT** or **TRL**) in the [Edit Kit](#) dialog box.

### SOLT ONLY

**Link FWD TRANS, FWD MATCH, REV TRANS, and REV MATCH** Check to automatically assign the standard definition for FWD TRANS to FWD MATCH, REV MATCH, and REV TRANS. Clear to separately assign FWD MATCH, REV MATCH and REV TRANS classes (SOLT calibrations only).

### Expanded Calibration

The following two check boxes **apply ONLY during Guided Calibrations**. For Unguided Calibration, these check boxes are ignored, including the case where the multiple standards dialog box is presented.

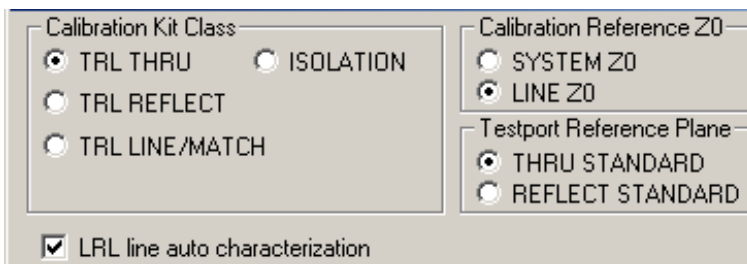
**Measure all mateable standards in class** Check this box to attain the very highest accuracy possible. For example, if a cal kit contains several load standards, during the calibration process you will be prompted to measure each of the standards. This could require a significant amount of calibration time. When checked, the "Use expanded math when possible" box is also checked automatically.

**Use expanded math when possible** Some kits contain multiple calibration standards of the same type that together cover a very wide frequency range. (For example: multiple shorts, or a lowband load and a sliding load.) If a calibration requires more than one standard to cover the calibration frequency range, there can be regions of overlapping measurements. When this checkbox is selected, the PNA automatically computes the most accurate measurement in the overlap regions using a "weighted least squares fit" algorithm. This function improves accuracy without slowing the calibration speed.

- Manually select this checkbox only when using a cal kit that contains multiple standards of the same type. (For example: multiple shorts, or a lowband load and a sliding load.)
- The checkbox is cleared by default when a [polynomial model](#) is selected from the cal kit menu.
- The checkbox is selected by default when the 85058B or 85058E [data-based model](#) is selected from the cal kit menu.

### TRL ONLY

If TRL is selected as Class Assignment in the [Edit Kit](#) dialog box, the following changes appear in this dialog:



The screenshot shows the 'Edit Kit' dialog box with the 'Calibration Kit Class' set to 'TRL THRU'. The 'Calibration Reference Z0' is set to 'LINE Z0'. The 'Testport Reference Plane' is set to 'THRU STANDARD'. The 'LRL line auto characterization' checkbox is checked.

Calibration Kit Class	Calibration Reference Z0	Testport Reference Plane
<input checked="" type="radio"/> TRL THRU	<input type="radio"/> SYSTEM Z0	<input checked="" type="radio"/> THRU STANDARD
<input type="radio"/> TRL REFLECT	<input checked="" type="radio"/> LINE Z0	<input type="radio"/> REFLECT STANDARD
<input type="radio"/> TRL LINE/MATCH		

☒ LRL line auto characterization

### Calibration Kit Class

- Learn more about [TRL standards](#).
- [Isolation calibration](#) is not usually necessary in the PNA.

### LRL line auto characterization

**Note:** This setting ONLY applies if an LRL Cal Kit is being modified **AND** Testport Reference Plane is set to Thru Standard **AND** the TRL Thru class standard and the TRL Line/Match class standard both have the same values for Offset Z0 and Loss. Otherwise, this setting is ignored.

- Check the box to allow the PNA to automatically correct for line loss and dispersion characteristics.
- Clear the box if anomalies appear during a calibrated measurement which may indicate different loss and impedance values for the Line standards.

#### Calibration Reference Z0 (TRL only)

**System Z0** The system impedance is used as the reference impedance. Choose when the desired test port impedance differs from the impedance of the LINE standard. Also, choose when skin effect impedance correction is desired for coax lines.

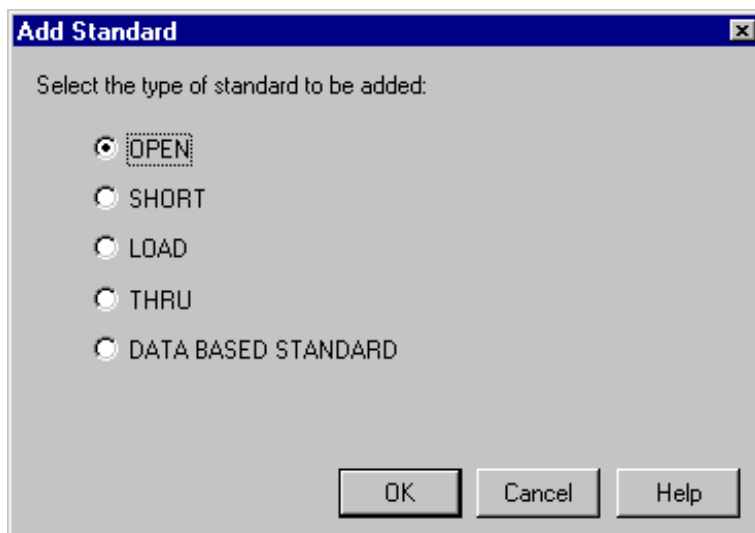
**Line Z0** The impedance of the line standard is used as the reference impedance, or center of the Smith Chart. Any reflection from the line standard is assumed to be part of the directivity error.

#### Testport Reference Plane (TRL only)

**Thru Standard** The THRU standard definition is used to establish the measurement reference plane. Select if the THRU standard is zero-length or very short.

**Reflect Standard** The REFLECT standard definition is used to establish the position of the measurement reference plane. Select if the THRU standard is not appropriate **AND** the delay of the REFLECT standard is well defined.

Also, select If a flush short is used for the REFLECT standard because a flush short provides a more accurate phase reference than a Thru standard.



### Add Standard dialog box help

Allows you to add standards to the calibration kit file.

**OPEN** Adds an open to the calibration kit file.

**SHORT** Adds a short to the calibration kit file.

**LOAD** Adds a load to the calibration kit file.

**THRU** Adds a thru to the calibration kit file.

**DATA BASED STANDARD** Adds a data-based standard to the calibration kit file.

**OK** Invokes a blank Edit Standards: [Open, Short, Load, Thru, or Data-Based](#) dialog box.

For more information see [Creating Custom Calibration Kits using a New Connector Family](#).

### Edit / Add Standards (Open, Short, Load, Thru, or Data-based)

The screenshot shows the 'Opens' dialog box with the following fields and values:

- Identification:**
  - Standard ID: 5
  - Label: OPEN\_M
- Open Description:** Type N (50) male open
- Frequency Range:**
  - Min: 0 MHz
  - Max: 999000 MHz
- Connector:** Type N (50) male
- Open Characteristics:**
  - C0: 0, Formula:  $F(e-15)$
  - C1: 0, Formula:  $F(e-27)\sqrt{\text{Hz}}$
  - C2: 0, Formula:  $F(e-36)\sqrt{\text{Hz}}^2$
  - C3: 0, Formula:  $F(e-45)\sqrt{\text{Hz}}^3$
- Delay Characteristics:**
  - Delay: 0 pSec
  - Loss: 0 dBrms/ft

### Edit / Add Standards dialog box help

The boxed areas of the previous graphic applies to all standard types.

The other areas change depending on the type of standard selected.

#### Identification

**Standard ID** Number in list of standards

**Label** Type of standard.

**Description** Description of standard.

#### Frequency Range

**Min** Defines the lowest frequency at which the standard is used for calibration.

**Max** Defines the highest frequency at which the standard is used for calibration.

### Connector

Indicates the type and gender (Male, Female, None) of the standard.

### Delay Characteristics

**Delay** Defines the one-way travel time from the calibration plane to the standard in seconds.

**Z0** Defines the impedance of the standard.

**Loss** Defines energy loss in Gohms, due to skin effect, along a one-way length of coaxial cable.

The following applies to standard types [Open](#), [Short](#), [Load](#), [Thru](#), and [Data-based](#)

### Open Standard

Open Characteristics					
C0	89.939	F(e-12)	C2	-264.9901	F(e-36)/Hz^2
C1	2536.7999	F(e-27)/Hz	C3	13.4	F(e-45)/Hz^3

**C0, C1, C2, C3** Specifies the fringing capacitance.

### Short Standard

Short Characteristics					
L0	0.7563	H(e-12)	L2	-52.429	H(e-33)/Hz^2
L1	459.8799	H(e-24)/Hz	L3	1.5846	H(e-42)/Hz^3

**L0, L1, L2, L3** Specifies the residual inductance.

### Load Standard

Load Type		Complex Impedance			
<input type="radio"/> Fixed Load	<input type="radio"/> Arbitrary Impedance	Real	50		
<input type="radio"/> Sliding Load	<input checked="" type="radio"/> Offset Load	Imag	0		
Delay Characteristics					
Delay	0	pSec	Loss	0	Gohms/s
Z0	50	ohms			
Offset Load Definition					
First Offset Standard	THRU				
Second Offset Standard	7-32 Line				
Load Standard	ADPTR/LOAD -M-				

Allows you to select the type of load.

## Load Type

**Fixed Load** Specifies the load type as Fixed. The fixed load is assumed to be a perfect termination without reflection.

**Sliding Load** A sliding load is defined by making multiple measurements of the device with the sliding load element positioned at various marked positions of a long transmission line. The transmission line is assumed to have zero reflections and the load element has a finite reflection that can be mathematically removed using a least squares circle fitting method.

A sliding load cal can be very accurate when performed perfectly. It can also be very inaccurate when not using proper technique. **For accurate results, closely follow the users manual instructions for the sliding load.**

**Arbitrary Impedance** Specifies the load type to be have an impedance value different from system Z0. An arbitrary impedance device is similar to a fixed load except that the load impedance is NOT perfect. Early firmware releases of the PNA series used a fixed resistance value. A complex terminating impedance has been added to allow for more accurate modeling of circuit board or on-wafer devices.

The following Complex Impedance settings are available ONLY when Arbitrary Impedance is selected.

- **Real** The real portion of the impedance value.
- **Imaginary** The imaginary portion of the impedance value.

**Offset Load** In Jan 2006, Offset Load definitions were added to TRL and Waveguide Cal Kit files. Using an Offset Load standard results in a more accurate calibration than with a Broadband Load. Therefore, when performing a calibration using one of the modified Cal Kit definitions, you may be prompted to connect more standards than before this change.

**Note:** The Offset Load standard can be used ONLY during a [SMART \(Guided\) calibration](#).

To revert to using the Broadband Load Standard without offset, do the following:

1. Click **Calibration**, then **Advanced Modify Cal Kit**
2. Select the kit, then click **Edit Kit**
3. Under Class Assignments, click **Edit**
4. Select Calibration Kit Class **S11C** (Loads)
5. Under Selected Standards, select **Broadband Load**, then click **Move Up** until the standard is at the top of the list. This will ensure that the Broadband Load is used first.

### About Offset Load

An offset load is a compound standard consisting of a load element and two known offset elements (transmission lines) of different length. The shorter offset element can be a zero-length (Flush-thru) offset. The load element is defined as a 1-port reflection standard. An offset load standard is used when the response of the offset elements are more precisely known than the response of the load element. This is the case with waveguide. Measurement of an offset load standard consists of two measurements, one with each of the two offset elements terminated by the load element. The frequency range of the offset load standard should be set so that there will be at least a 20 degree separation between the expected response of each measurement.

To specify more than two offset elements, define multiple offset load standards. In cases where more than two offsets are used, the frequency range may be extended as the internal algorithm at each frequency will search through all of the possible combinations of offsets to find the pair with the widest expected separation to use in determining the actual response of the load element.

The following Offset Load settings are available ONLY when Offset Load is selected.

- First Offset Standard
- Second Offset Standard
- Load Standard

### Thru Standard



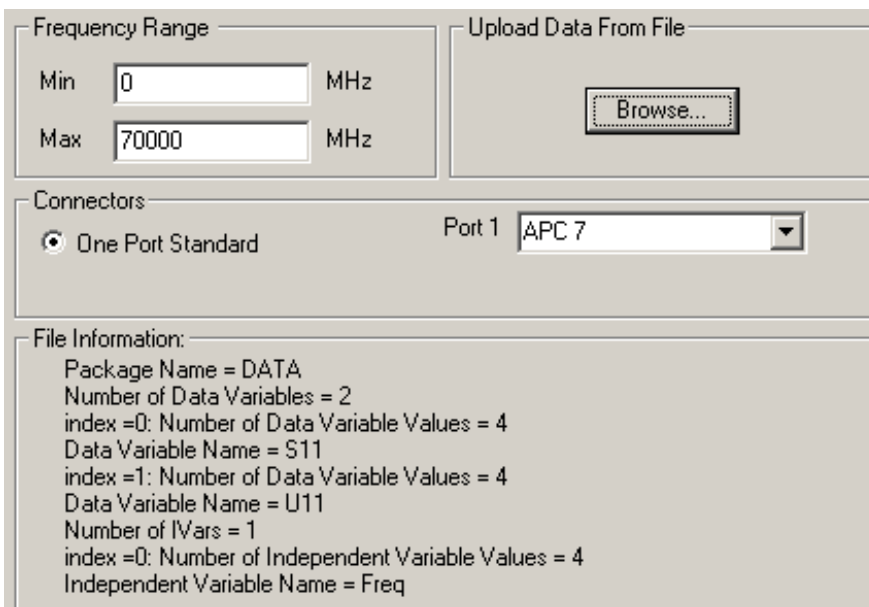
Connectors

Port 1 Type A (50) female Port 2 Type A (50) female

### Connectors

Defines connector type and gender at both ports.

### Data-Based Standard



Frequency Range

Min 0 MHz

Max 70000 MHz

Upload Data From File

Browse...

Connectors

☒ One Port Standard Port 1 APC 7

File Information:

Package Name = DATA

Number of Data Variables = 2

index =0: Number of Data Variable Values = 4

Data Variable Name = S11

index =1: Number of Data Variable Values = 4

Data Variable Name = U11

Number of IVars = 1

index =0: Number of Independent Variable Values = 4

Independent Variable Name = Freq

**Note:** To learn how to modify data-based standard files, visit <http://na.tm.agilent.com/pna/dbcal.html>

The modified file can then be uploaded into the PNA.

### Upload Data From File

Click Browse to load data from a file.

### Connectors

**One Port Standard** Currently only 1-port standards can be modified.

**Port 1** Select the type of connector.

**File Information** Information about the standard that is read from the uploaded file.

Last modified:

3-Sep-2008	Removed legacy content
14-Apr-2008	Add offset load note
4-Jan-2008	Added limit for imported kits
26-Oct-2007	Added Height/Width for Add connector. Moved waveguide settings.
2-Feb-2007	MX Added UI

## Power Calibration

---

Source and Receiver Power Calibrations work together to provide very accurate power levels from the source, and very accurate power measurements from the PNA receiver.

- [Source Power Calibration Overview](#)
- [Supported Power Meters and Sensors](#)
- [How to perform Source Power Calibration](#)
- [Setup](#)
- [Source Power Cal dialog](#)
  - [Source Power Calibration Option dialog](#)
  - [Power Meter Settings dialog](#)
  - [Power Loss Compensation dialog](#)
  - [Power Sensor Settings dialog](#)
- [Copy a Source Power Calibration to other Channels](#)
- [Saving a Source Power Calibration](#)
- [Reducing Time to Complete a Source Power Calibration](#)
- [Receiver Power Calibration](#)
- [Saving Receiver Cals](#)

[See other Calibration Topics](#)

### Source Power Calibration Overview

Perform Source Power Calibration when you need accurate power levels at some point in the measurement path between the PNA test ports. For example, you need to characterize the gain of an amplifier across a frequency range at a specified input power. You would perform a source power cal at the input of the amplifier to ensure the **exact** power level into the amplifier across the frequency range.

Using a Source Power Cal, you can expect the power at the point of calibration to be within the range of the uncertainty of the power meter and sensor that is used.

**Note:** You may not be allowed to perform a Source Power Cal unless you are logged on to the PNA with an [Administrator user account](#).

#### Source Power Calibration:

- Is independent of measurement type. It corrects the PNA source regardless of which receivers are being



used in a measurement. Therefore, it can be used with both [ratio or non-ratio measurements](#).

- Applies ONLY to those measurements on the selected channel that use the test port that was [specified as the Source](#) for the calibration. For example, if you specify Channel 1 and Port 1 as the source to be calibrated, only those measurements on channel 1 that use port 1 as the source will be corrected.
- Can be used in conjunction with other measurement calibrations, such as a full 2-port calibration. For highest accuracy, perform the measurement calibration AFTER the source calibration.
- Can be used with [Power Sweep](#) type. Source Power Cal will correct the power at all power levels across the power sweep.
- Can be used with [Port Power Uncoupled](#).
- Forces [sweep mode to Stepped](#) on measurements with source power correction turned ON.
- Beginning with PNA Rev. 7.50, an external source can be calibrated using Source Power Cal.

### Overview of How it works:

[Click to see the detailed procedure](#)

1. Specify the measurement settings (frequency range, IFBW and so forth).
2. Start Source Power Calibration.

**Note:** When using an Agilent 848X power sensor (sensors that do NOT have built-in calibration factors), enter the Cal Factors using the [Power Sensor Settings](#) dialog, because the PNA instructs the power meter to NOT use the Cal Factor tables internal to the power meter.

3. Connect a power meter sensor to the point at which you want a known power level. This may be at the input or output of your device, or some other point between the test ports.
4. The PNA source is stepped through the specified frequency range, and power is measured with the power meter. At each data point, the source power is adjusted until the measured power is within your specified accuracy level.
5. When complete, the power meter is preset. The source power calibration can be [saved as part of the instrument state](#).
6. The power meter is removed and the measurement path reconnected.
7. The calibration is automatically applied to the channel. All measurements on that channel using that source port benefit from the source power cal.

**Verify** the source power calibration using the following procedure.

1. Connect the power meter as it was during the source power calibration.
2. Set the PNA to [Point Trigger](#) mode.
3. Trigger the PNA across the trace. Read about the behavior of the [sweep indicator](#).

4. At each data point, the power meter should read the corrected power level within the specified tolerance.

## Supported Power Meters and Sensors

### Power Meters

- HP 437B / 438A
- E-Series power meter (E4418/9) and all supported sensors.
- P-Series power meters (N1911/2) and all supported sensors.
- EPM Series power meters (E4416/7) and all supported sensors.
- U2000 Series USB Power Sensors.

The [82357A USB/GPIB Interface](#) can be used to control the power meter.

[LAN connectivity](#) can ONLY be used with the Agilent P-series power meters.

In addition, you can [Create a Custom Power Meter Driver](#) for use with other power meters.

### Power Sensors

You can perform a Source Power Calibration with ALL power sensors that are supported by the above power meters. However, Source Power Calibration operates slowly with the Agilent E930x and E932x power sensors.

Up to two sensors can be used to cover the frequency span of the measurement.

**USB power sensors** are supported beginning with PNA Rev 7.50.

- Only one USB power sensor can be used to cover the entire frequency span.
- To select a USB power sensor:
  1. Connect the sensor directly to one of the PNA USB ports.
  2. From the [Source Power Cal](#) dialog, click **Power Meter Config**.
  3. On the [Power Meter Settings](#) dialog, select **USB**.

See note about [Zeroing USB Power Sensors](#).

See also: [Power Meters as Receivers](#) (PMAR)

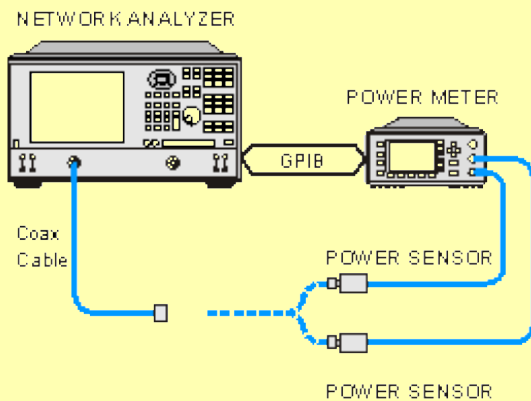
### How to perform Source Power Calibration

1. Setup your measurement (sweep type, frequency range, IFBW, and so forth). By default, a Source

Power Cal is performed on the source port of the active measurement.

2. Connect coax cable, GPIB cable, and power sensors to the PNA as shown in graphic below.

This image does NOT apply to USB power sensors, which are connected directly to a PNA USB port.



3. Apply power to the power meter and allow 30 minutes warm-up time before beginning calibration.
4. Select **Source Power Cal** as follows:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Power Cal]**
3. then **[Source Cal]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Power Cal**
4. then **Source Cal**

#### Programming Commands

5. Complete the Source Power Cal dialog box (below), including [Loss Compensation](#) and [Power Sensor Settings](#), as needed.

**Note:** When using an Agilent 848X power sensor (sensors that do NOT have built-in calibration factors), enter the Cal Factors using the [Power Sensor Settings](#) dialog, because the PNA instructs the power meter to NOT use the Cal Factor tables internal to the power meter.

6. When complete, click **Take a Cal Sweep** in the Source Power Cal dialog box.
7. Follow the prompts to connect the sensors as required.
8. At this time you can change the Source Port setting and perform a Source Power Cal on a different port.
9. When calibration is finished, click **OK**. Correction is then applied and turned ON for the calibrated ports

on the active channel.

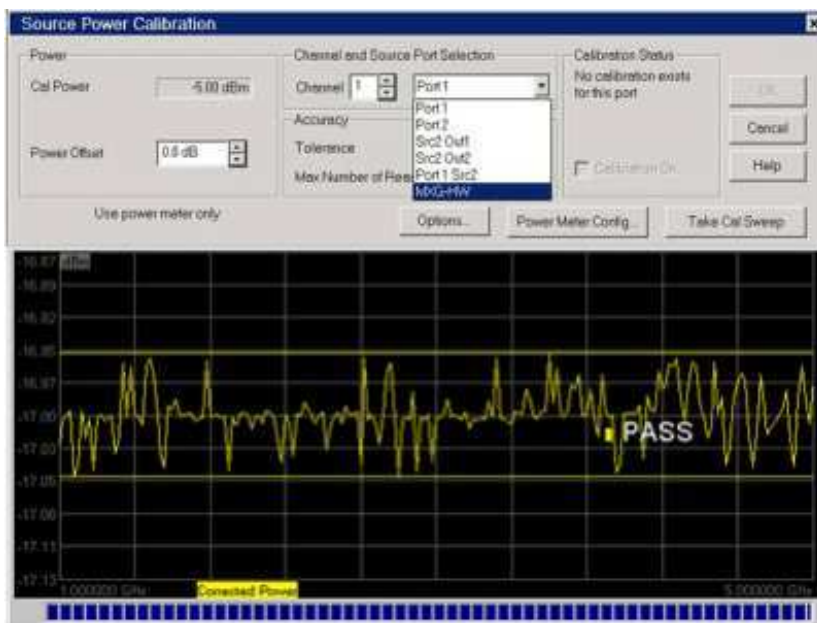
10. Remove sensor.
11. **SrcPwrCal** is displayed in the status bar when Source Power Correction is applied to the Active Measurement.

#### To turn Source Power Correction OFF:

- On the **Calibration** menu, point to **Power Calibration**, then click **Source Power Correction on/OFF**.
- ONLY correction for the source port of the ACTIVE MEASUREMENT is turned OFF (regardless of [port power coupling setting](#).)

#### Interpolation or Extrapolation

If the original stimulus settings are changed, Interpolation or EXTRAPOLATION is applied and **SrcPwrCal\*** is displayed in the status bar. This is different from [measurement calibration interpolation](#). For example, if the frequency span is increased, the PNA will extrapolate new correction values rather than turn correction off. This is to protect your test device from being overpowered by the source. If the original settings are restored, then source power calibration returns to full correction.



#### Source Power Cal dialog box help

**Note:** Be sure that the frequency range of your power sensor covers the frequency range of your measurement. This does NOT occur automatically.

#### Power

**Cal Power** The calculated power (in dBm) at the calibration point. This value is the specified PNA source

power plus the Power Offset value.

**Power Offset** Allows you to specify a gain or loss (in dB) to account for components you connect between the source and the reference plane of your measurement. These components will remain during a measurement. For example, specify 10 dB to account for a 10 dB amplifier in the path to your DUT. Following the calibration, the PNA power readouts are adjusted to this value.

To account for components that will be removed when the calibration is complete, use the [Loss Compensation table](#).

## Channel and Port Selection

**Channel** Specifies the channel on which to perform the calibration. This setting defaults to the active channel.

**Source Port** Specifies the source port to be corrected. This setting defaults to the source port for the active measurement.

Beginning with PNA Rev. 7.22, external sources can be calibrated using this dialog. [Learn more](#).

## Accuracy

At each data point, power is measured using the [specified Power Meter Settling Tolerance](#) and adjusted, until the reading is within this Accuracy **Tolerance** or the **Max Number of Readings** has been met. The **last** power reading is plotted on the screen against the Tolerance limit lines.

**Tolerance** Sets the maximum desired deviation from the specified **Cal Power** level in 0.005 dB increments from 0 to 5 dB.

**Max Number of Readings** Sets the maximum number of readings to take at each data point for iterating the source power.

## Calibration Status

Allows you to turn Source Power Cal ON | OFF and view Cal data for each port, regardless of the active measurement. This feature allows the [Internal Second Source](#) to be calibrated and turned ON | OFF, even when being used as an incidental source in a measurement, such as an LO.

**Calibration ON** Check to turn Source Power Calibration ON for the specified source port.

The displayed text indicates when [interpolation](#) is applied for the calibration.

## Buttons

**Options** Invokes the [Source Power Cal Options](#) dialog. Label to the left of the button displays the current 'Options' setting.

**Power Meter Config** Invokes the [Power Meter Settings](#) dialog box

**Take Cal Sweep** Begins source power calibration measurement.

**OK** Applies calibration. This button is disabled until the Take Cal Sweep has been pressed.

**Cancel** If a sweep is in progress, cancels the sweep. Press again to close the dialog.

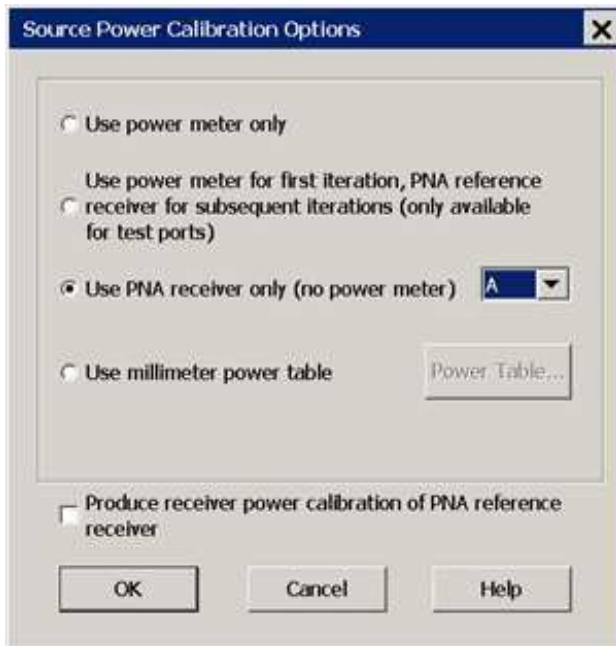
## Pass / Fail Limits

Limit lines are drawn on the Source Power Cal measurement graticule area. These lines are at the Cal Power +/- the current setting of Accuracy Tolerance. A FAIL during the Source Power Cal sweep means that the PNA was unable to measure power to within the Accuracy Tolerance. Tight tolerances are more difficult to achieve at lower Cal Power levels. When a FAIL indication appears, increase the Max Number of Readings. If this does

not cause a PASS condition, then decrease the Accuracy Tolerance value.

### See Also

- Learn more about [Source Power Cal](#)
- Learn about [External Testsets](#) and Source Power Cal.



### Source Power Calibration Options dialog box help

Provides options for measurement of the source power.

- **Use power meter only** Traditional source power calibration using only a power meter to measure the source power at each data point. Most accurate and slowest method.

**Note:** Because the following two settings use PNA receivers to make power measurements, they do NOT work correctly when a [Frequency Offset](#) value is being used.

- **Use power meter for first iteration...** When checked, the first reading at each data point uses a power meter to calibrate the reference receiver. Subsequent readings, if necessary to meet your accuracy requirement, are measured using the reference receiver. This technique is much faster than using the power meter with almost no degradation in accuracy.

**Note:** Do NOT use the "first iteration" feature if there is a component before the power sensor that exhibits non-linear behavior, such as a power amplifier in compression.

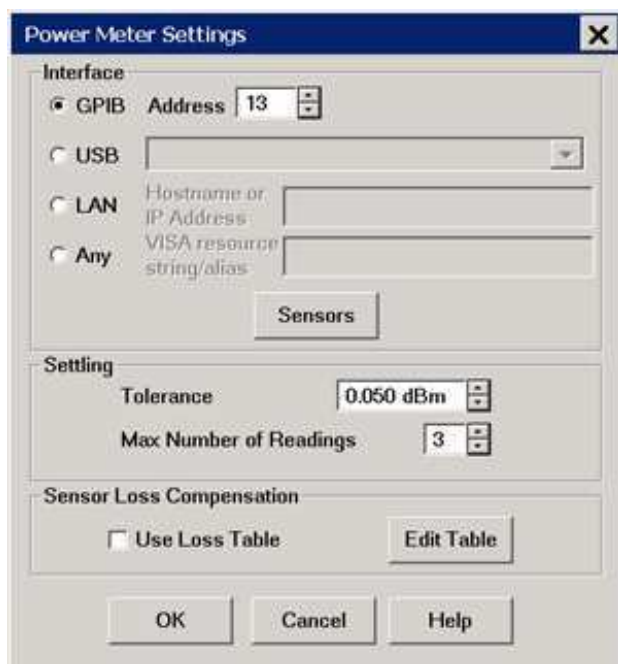
- **Use PNA receiver only** This feature assumes that the receiver to be used has already been calibrated by a source power cal using a power meter, then a [receiver cal](#). That receiver can then be used to quickly calibrate other PNA source ports, or used on another channel with different stimulus settings.

This would be useful, for example, if the power level of the measurement was below the sensitivity of the power sensor. Calibrate the PNA receiver using a source power cal that is within the sensitivity of the sensor. Then, use the calibrated receiver to perform a second source power cal at the reduced power level.

The receiver is specified using either standard receiver notation or [logical receiver notation](#).

- It is best to use the reference receiver for the source port to be calibrated. For example, if calibrating source port 2, specify the "B" or "a2" receiver.
- To ensure an accurate source power cal, the frequency range over which the receiver was calibrated must be the same or larger than the "receiver only" source power calibration.
- All accuracy and settling tolerance and number of reading settings apply just as they do with a power meter reading.
- **Use millimeter power table** Used to provide power leveling with mmWave test set and modules. [Learn more.](#)

**Produce receiver power calibration of PNA reference receiver** Check to calibrate the appropriate reference receiver to the power level that is measured at the calibration plane. Do this to make very accurate measurements using the calibrated reference receiver. This cal is done in addition to the standard source power cal using the any of the methods listed above. At the end of the source power cal measurement sweep, you can optionally save the reference receiver cal to a Cal Set to be recalled at a later time. The Cal is saved when the **OK** button is clicked to close the Source Power Cal dialog.



The image shows a software dialog box titled "Power Meter Settings". It has a standard Windows-style title bar with a close button (X). The dialog is divided into several sections:

- Interface:** Contains radio buttons for "GPIB", "USB", "LAN", and "Any". The "GPIB" option is selected. Next to "GPIB" is a text field labeled "Address" with the value "13". The "USB" option has a dropdown menu. The "LAN" option has a text field labeled "Hostname or IP Address". The "Any" option has a text field labeled "VISA resource string/alias". Below these fields is a button labeled "Sensors".
- Settling:** Contains two settings: "Tolerance" with a value of "0.050 dBm" and "Max Number of Readings" with a value of "3". Both values are in text boxes with up/down arrows.
- Sensor Loss Compensation:** Contains a checkbox labeled "Use Loss Table" which is currently unchecked. To the right of the checkbox is a button labeled "Edit Table".

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

## Power Meter Settings dialog box help

This dialog appears when you click the **Power Meter Config** button on many dialog boxes.

### Communication

- **GPIOB / Address** Select GPIOB power meter. Then select the address for the power meter. Default is 13. The PNA will search VISA interfaces that are configured in the Agilent IO Libraries on the PNA.
- **USB** PNA scans the USB for connected power sensors. Select a power sensor from the list. Only ONE USB power sensor can be configured to cover the entire frequency range of the calibration. [See USB sensor accuracy note.](#)
- **LAN** Specify the Hostname or IP address of the Power Meter. This setting can ONLY be used with the Agilent P-series power meters.
- **Any** For future use.

**Sensors** Invokes the [power sensor settings](#) dialog box.

### Settling

These Settling settings do not apply when a PNA receiver is the power measurement device. Each power meter reading is "settled" when either:

- two consecutive meter readings are within this **Tolerance** value or
- when the **Max Number of Readings** has been met.

The readings that were taken are averaged together to become the "settled" reading. The settled reading is then compared to the [Accuracy Tolerance requirements](#) (tolerance and max readings) specified on the Source Power Cal dialog box.

**Tolerance** When consecutive power meter readings are within this value of each other, then the reading is considered settled.

**Max Number of Readings** Sets the maximum number of readings the power meter will take to achieve settling.

### Sensor Loss Compensation

**Use Loss Table** Select this checkbox to apply loss data to Source Power calibration correction (such as for an adapter on the power sensor).

**Edit Table** Invokes the [Power Loss Compensation](#) dialog box.



The dialog box titled "Power Loss Compensation" contains a table with two columns: "FREQUENCY" and "LOSS". The table has two rows of data. Below the table, there is a "Delete Table Segment" button followed by a numeric input field set to "1" and a "Delete All" button. At the bottom are "OK", "Cancel", and "Help" buttons.

	FREQUENCY	LOSS
1	1.000000 GHz	0.50 dB
2	2.000000 GHz	1.00 dB

### Power Loss Compensation dialog box help

**To Add a Row** to the table, click on a row in the table and press the down arrow on either the PNA front panel or keyboard.

**To Edit a value**, double-click in the cell to be edited.

Beginning with A.08.33, these values can be loaded from an S2P file using the [Characterize Adaptor Macro](#).

Compensates for losses that occur when using an adapter or coupler to connect the power sensor to the measurement port. These components will be removed when the calibration is complete. To account for components that will remain during the measurement, use the [Power Offset setting](#).

The Frequency / Loss pairs define the amount of loss for the entire frequency range. For example, using the entries in the above dialog image:

- 0.5 dB is used to compensate power sensor measurements up to 1 GHz.
- Each data point between 1 GHz to 2 GHz is linearly interpolated between 0.5 dB and 1 dB.
- 1 dB is used above 2 GHz.
- A single frequency/loss segment is applied to the entire frequency range.

Enter up to 100 segments to achieve greater accuracy.

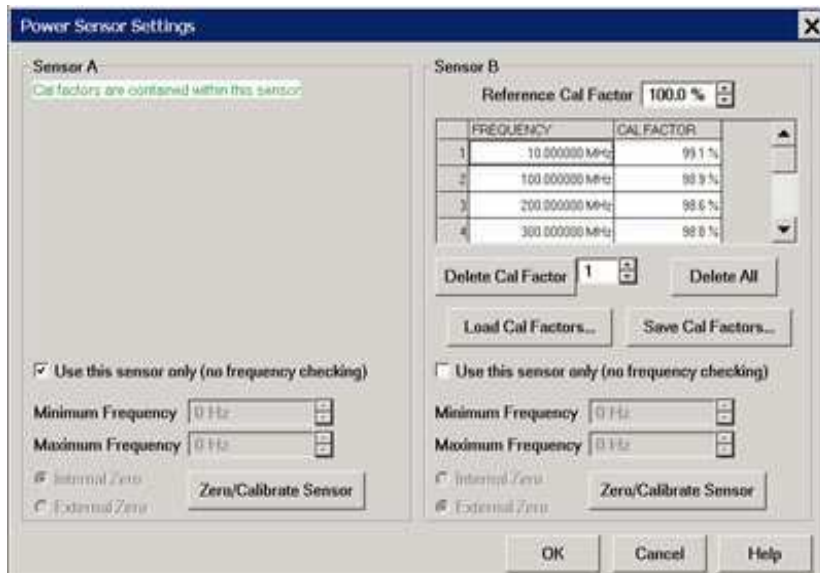
**Frequency** Enter a frequency in Hz.

**Loss** Enter a loss as a POSITIVE value in dB. To compensate for gain, use NEGATIVE values.

**Delete Table Segment** Deletes row indicated in the field.

**Delete All** Deletes all data in the table.

The Power Loss Compensation table survives PNA Preset and Power OFF. To NOT use Loss compensation, clear the Use Loss table checkbox on the [Power Meter Settings](#) dialog.



### Power Sensor Settings dialog box help

This dialog appears when you click the **Sensors** button on the [Power Meter Settings dialog](#).

**Note:** Be sure that the frequency range of your power sensor covers the frequency range of your measurement. This does NOT occur automatically.

**Sensor A (B)** Displays one of the following messages depending on type of sensor.

- **Not connected** The PNA is not detecting a power sensor.
- **Cal factors are contained within this sensor** Internal Reference Cal Factor and Cal Factor data are loaded automatically.
- **Sensor Data** Allows the following entries for power sensor data:
  - **Reference Cal Factor** Specifies the sensor's Reference Cal Factor.
  - **Cal Factor Table** Specifies the frequency and corresponding Cal Factor for the sensor.
  - **Delete Cal Factor** Deletes the indicated row in the table.
  - **Delete All** Deletes all data in the table.
  - **To Add a Row** to the table, click on a row in the table and press the down arrow on either the PNA front panel or keyboard. A row is added to the bottom of the table. The table is automatically sorted by frequency when OK is pressed.

**Use this sensor only** Check this box to use this sensor over the entire frequency span of the measurement, even if two sensors are connected to power meter.

Clear this box to allow entry of minimum and maximum frequencies for the sensor. Only ONE of the two sensors can have this box checked. You will be prompted to connect the appropriate sensor during the power calibration.

**Minimum Frequency** Specifies the minimum frequency range for the sensor when using dual sensors.

**Maximum Frequency** Specifies the maximum frequency range for the sensor when using dual sensors.

**Perform Sensor Zeroing and Calibration** Zero and/or calibrate the power sensor before measuring data. Follow prompts that may appear. Press **Zero/Calibrate Sensor** to perform required calibration steps.

If the following settings are 'greyed', Internal or External zeroing is selected automatically based on the power meter/sensor model.

- **Internal Zero** - A switch inside the power sensor removes the zero circuit from the incident power.
- **External Zero** - Requires that you physically remove the sensor from incident power.

#### Note for the U2000 Series USB power sensors

Select External Zero ONLY when the power to be measured is **below** the specified level. Otherwise, the U2000 series performs internal zeroing automatically when needed. See your power sensor documentation for more details.

- U200xA - below -30 dBm
- U200xH - below -20 dBm
- U200xB - below 0 dBm

If your U2000 power sensor 'hangs' when external zeroing, upgrade the power sensor firmware to Rev. A.01.02.00 or higher to fix this problem.

## Copy a Source Power Calibration to other Channels

A macro application is now available that copies a Source Power Calibration to other channels. Once downloaded and installed on a PNA, the [macro](#) is automatically configured up. To learn more, click **Help** on the application main dialog. Get the application from <http://na.tm.agilent.com/pna/apps/applications.htm>.

## Saving a Source Power Calibration

Because Source Power Cal calibrates source hardware, the calibration data is saved as part of the **Instrument State**, in either a .sta file or a .cst file. This correction is applied to all measurements on the channel that uses the calibrated source. See [Save Instrument State](#).

## Reducing Time to Complete a Source Power Calibration

The time required to perform a Source Power Calibration depends on source power, number of points, and number of readings taken. You can reduce this measurement time with the following methods:

- **Reduce number of points before calibration.** You can reduce the number of points before the measurement, then return the number of points to its original value after calibration is complete and correction is ON. The analyzer will perform a linear interpolation, although with some loss in accuracy.

- **Use an Agilent E-Series sensor.** You can obtain 40+ readings per second over GPIB with this type of sensor on the PNA.
- **Increase power to the sensor.** Lower power may have longer settling time with some sensors.
- Check [Use Reference Receiver for Iteration](#).

## Receiver Power Calibration

Receiver power calibration mathematically removes frequency response errors in the specified PNA receiver, and adjusts readings to the same, or a value offset from, the source power calibration level. It is the same as doing a **Response Cal** or **Data / Memory, (Normalization)** but with the data shifted to the [Cal Power](#) value.

Use Receiver Power Calibration to make very accurate absolute power (amplitude) measurements.

### Receiver Power Calibration:

- Is ONLY allowed when making absolute power ([unratioed](#)) measurements.
- Is most accurate when a source power calibration was performed first.
- Applies to all unratioed measurements in the active channel using that receiver.
- Can be saved in a Cal Set and later reapplied to a like measurement.

## Interpolation

Like other calibration types, if the original stimulus settings are narrowed, interpolation is applied and **C\* Rcvr Pwr** is displayed in the status bar. If the original stimulus settings are made wider, the PNA will turn Receiver Power Correction **OFF**.

If the original settings are restored, then receiver power calibration returns to full correction.

### How to perform a Receiver Power Calibration

1. Perform a [Source Power Calibration](#).
2. Set the active measurement to unratioed. [Learn How](#).
3. Connect a THRU line from the source port to the receiver port.
  - When performing a receiver power cal on a reference receiver, no connection is necessary as the receiver is internally connected to the source.
  - When the receiver port and the source port are the same (receiver A, source port 1), then connect an open or short to get maximum power to the receiver. This practice is not recommended. It is best to use different ports for the source and receiver.
4. Ensure correction for Source Power Calibration is ON as indicated by **Src Pwr Cal** or **Src Pwr Cal\*** in the status bar.

5. Start the [Calibration Wizard](#)

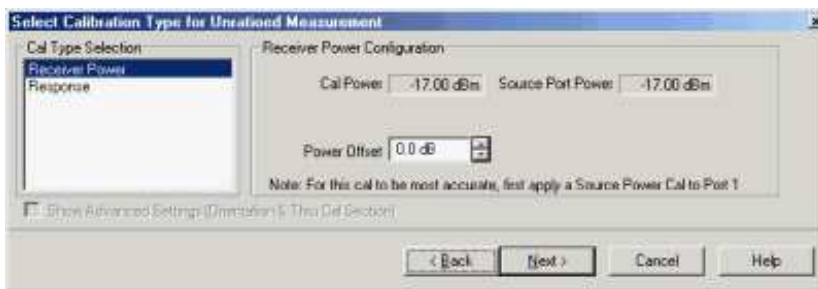
**Using front-panel  
HARDKEY [softkey] buttons**

1. Press **CAL**
2. then **[Power Cal]**
3. then **[Receiver Cal]**

**Using a mouse with PNA Menus**

1. Click **Response**
2. then **Cal**
3. then **Power Cal**
4. then **Receiver Cal**

**Programming Commands**



**Select Calibration Type for Unratioed Measurement dialog box help**

**Cal Type Selection** Select **Receiver Power**

**Receiver Power Configuration**

**Cal Power** Specifies the power level to be displayed on the measurement when complete. (Source Port Power + Power Offset).

**Source Port Power** Test port Power set for the measurement. [Learn how to change Test Port Power](#)

**Power Offset** Allows you to specify a gain or loss (in dB) to account for components you connect between the source and the reference plane of your measurement AFTER a source power cal has been performed. Following the calibration, the PNA power readouts are adjusted to the Cal Power value.

**Next** Click to continue the Calibration Wizard.

**Notes:**

- When Receiver Power Cal is finished, **C RcvrPwr** is displayed in the status bar and correction data is applied to subsequent sweeps.
- To turn correction **OFF**, click **Calibration**, point to **Power Calibration**, then set **Receiver Power Correction** to **OFF**.

[Learn more about Receiver Power Cal \(scroll up\).](#)

## Saving a Receiver Power Calibration

Beginning with PNA Revision 5.0, Receiver Power Cal is saved to a [Cal Register](#) and optionally to a [User Cal Set](#). It can be applied to measurements in the same way as other Cal Types. Previously, Receiver Power Cal data was saved as part of an Instrument State and was only applied to the measurement on which it was performed.

[Learn more about Saving PNA files types.](#)

---

Last modified:

30-Jan-2009	Added USB sensor note
11-Dec-2008	Added tolerance increment.
8-Dec-2008	Added persistence of power loss table Link to characterize Adaptor Macro A.08.33
14-Aug-2008	Clarified a few concepts from BH.
24-Jun-2008	Updated diag image
2-Jun-2008	Clarified Power Loss compensation.
21-Feb-2008	Added 848x note
4-Jan-2008	Added Cal note for USB sensors
30-Oct-2007	Added link to supported Power meters/ sensors
20-Jul-2007	Added USB / LAN support and Apply macro
21-Jan-2007	MX Added UI
14 Sept-2006	MQ Added Receiver-only SPC.

## Fixture Simulator

---

The following features allow you to mathematically add (embed) or remove (de-embed) circuits to, or from, your PNA measurements. The mathematical models are applied to specific ports for all measurements on the channel.

### Notes

- Beginning with PNA Rev. A.09.20, the following features are available in [GCA](#), [Swept IMD](#), [Swept IMDx](#), [Noise Figure](#), and [NFX](#) Apps:
  - [Port Extensions](#) (Not available in Swept IMD or IMDx)
  - [2 Port De-embedding](#)
  - [Port Matching](#)
  - [Port Z Conversion](#)
- All other Fixturing features are available ONLY in a [standard channel](#).
- When fixturing is enabled, all of the enabled fixturing features are applied when [snp files are saved](#).

### See Also

- ["De-embedding and Embedding S-Parameter Networks Using a Vector Network Analyzer" App note](#) for more conceptual information on Fixture Simulation.
- [Characterize Adaptor Macro](#) can be used to create S2P files from Cal Sets.
- [To Embed or De-embed? and the associated procedures](#)
- [See an example](#) of how these functions can be used to de-embed unwanted effects of a test fixture, and then mathematically embed the DUT in the circuit in which it is used.

### Order of Fixture Operations

- The fixturing operations are applied to the measurement results. The order of operations 1 through 4 can be changed using the SCPI command: [CALC:FSIM:SEND:OORD](#). Learn how to send this command from the [GPIB Command Processor Console](#).
- The order of the operations 5 through 8 can NOT be changed.
- In the [PNA data processing chain](#), the Fixture Simulator functions occur at the same time as the **Apply Error Terms** block.

First, the following **Single-ended** measurement functions are processed in this order:

1. [Port Extensions](#)

2. [2-Port De-embedding](#)
3. [Port Matching Circuit Embedding](#)
4. [Port Z \(Impedance\) Conversion](#)
5. [4-Port Network \(single-ended\) Embed/De-embed](#)

Then, **Balanced** measurement functions are processed in this order:

6. [Balanced Conversion](#)
7. [Differential](#) / [Common Mode](#) Port Z Conversion
8. [Differential Matching Circuit Embedding](#)

Then, [Source power compensation](#) is optionally applied to compensate for the aggregate loss through all enabled fixturing operations.

**Note: Port Z (Impedance) conversion uses values in the following prioritized order:**

1. Balanced ([Differential](#) or [Common Mode](#)) - if enabled, these values are always used.
2. [Single Port Impedance](#) - if enabled, this value is used if Balanced is not enabled.
3. [System Impedance](#) - if neither balanced or single port is enabled, this value is used.

## How to select Fixturing Simulator

### About Fixturing ON/off

**BOTH** of the following must occur to turn a fixturing selection **ON**.

**EITHER ONE** will turn a fixturing selection **OFF**.

1. Check **Fixturing ON/off**  
Port Extensions is NOT affected by Fixturing ON/off.
2. Check **Enable** on the individual fixturing selection dialog box.

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[More]**
3. then **[Fixtures]**

#### Using a mouse with PNA Menus

1. Click **Cal**
2. then **More**
3. then **Fixtures**



**Port Matching**

☐ Enable Port Matching

Choose Circuit Model for Matching

Port 1 None

User S2P File... <None Selected>

Circuit Values

C 1.000 uF

L 1.000 mH

R 50.000 ohms

G 20.000 mS

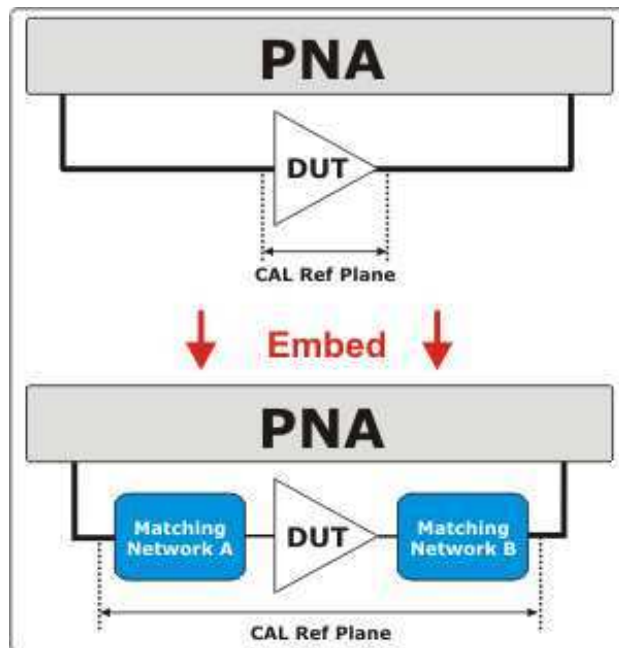
Reset

Close Help

#### Port Matching dialog box help

**Note:** Beginning with PNA Rev. A.09.20, this feature is available to the following [measurement classes](#): [GCA](#), [Swept IMD](#), [Swept IMDx](#), [Noise Figure](#), [NFX](#), and standard (S-Parameter) channels.

This function specifies a circuit to embed (add) to the measurement results. [See Order of Fixture Operations.](#)

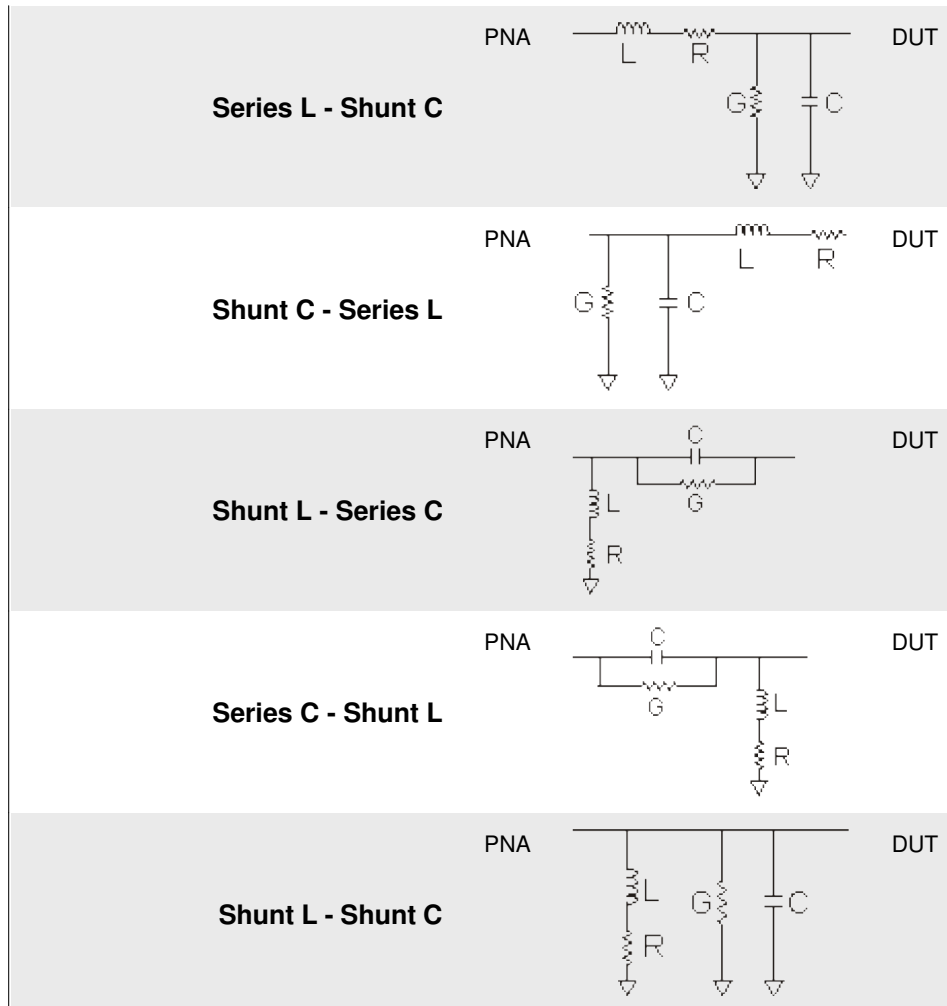


**Enable Port Matching** Check to apply the settings to the measurement results. Must also enable [Fixturing](#)

[ON/off.](#)

**Port** - Select Port in which to apply simulation.

**Circuit Model for Matching** - Choose one of the following that best emulates your fixture at the selected PNA port:



**User Defined (S2P File)** Load a file that is specified with **User S2P File** button.

**None** Use no circuit model.

**User S2P File** Click to specify an S2P file of the circuit model to embed at the selected port. If the normalized impedance value in a recalled User .S2P file is different from the port reference impedance setting of the PNA, the PNA setting is used. [Characterize Adaptor Macro](#) can be used to create S2P files from Cal Sets.

#### Circuit Values

**Capacitance (C), Inductance(L), Resistance(R), Conductance(G)** Values for the specific components of the circuit type that models your fixture.

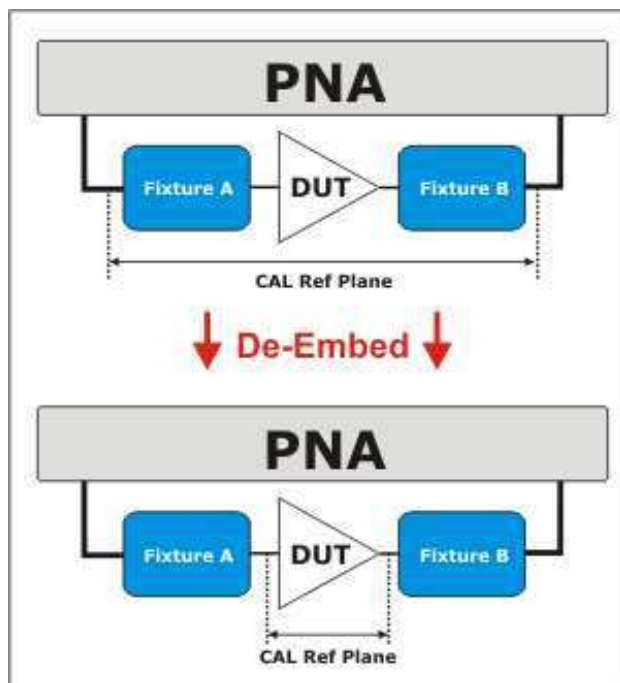
**Reset** Restores the default values.



## 2 Port De-embedding dialog box help

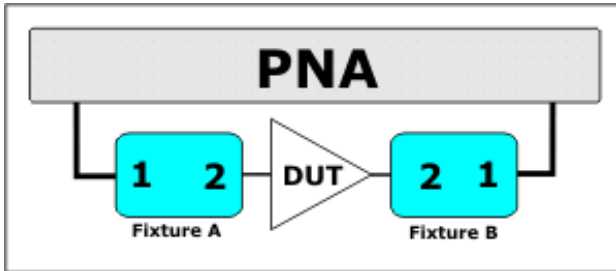
**Note:** Beginning with PNA Rev. A.09.20, this feature is available to the following [measurement classes](#): [GCA](#), [Swept IMD](#), [Swept IMDx](#), [Noise Figure](#), [NFX](#), and standard (S-Parameter) channels.

De-Embed when you have performed a calibration and then added a fixture (an adapter, an attenuator, a longer cable, etc.) that connects between the Cal reference plane and your DUT. This function **removes** the effects of a test fixture from the measurement results.



The de-embedding operation recalls an .s2p file (Touchstone format) which includes the electrical characteristics of a 2-port fixture or device. The file can be in any standard format (real-imaginary, magnitude-angle, dB-angle).

In the following image, the 2-port fixture would be either Fixture A **OR** Fixture B. To de-embed both, perform this operation for each fixture.



**Note:** In all cases:

- Port 1 of the fixture is assumed to be connected to the PNA.
- Port 2 of the fixture is assumed to be connected to the DUT.

**Enable De-embedding** Check to apply the settings to the measurement results. Must also enable [Fixturing ON/off](#).

**Port** - The **PNA** port to which the recalled de-embedding file is applied.

From the drop-down menu, select **User S2P**.

**User S2P File** Click to specify an existing .S2P file. If the normalized impedance value in a recalled User .S2P file is different from the port reference impedance setting of the PNA, the PNA setting is used. [Characterize Adaptor Macro](#) can be used to create S2P files from Cal Sets.



### Port Z (Impedance) Conversion dialog box help

**Note:** Beginning with PNA Rev. A.09.20, this feature is available to the following [measurement classes](#): [GCA](#), [Swept IMD](#), [Swept IMDx](#), [Noise Figure](#), [NFX](#), and standard (S-Parameter) channels.

This function corrects the measurement and displays the results as if the measurement had been made into the specified impedance value. However, the physical port termination is still approximately 50 ohms.

The specified impedance value is applied to all of the measurements on **ONLY** the active channel.

[See Order of Fixture Operations.](#)

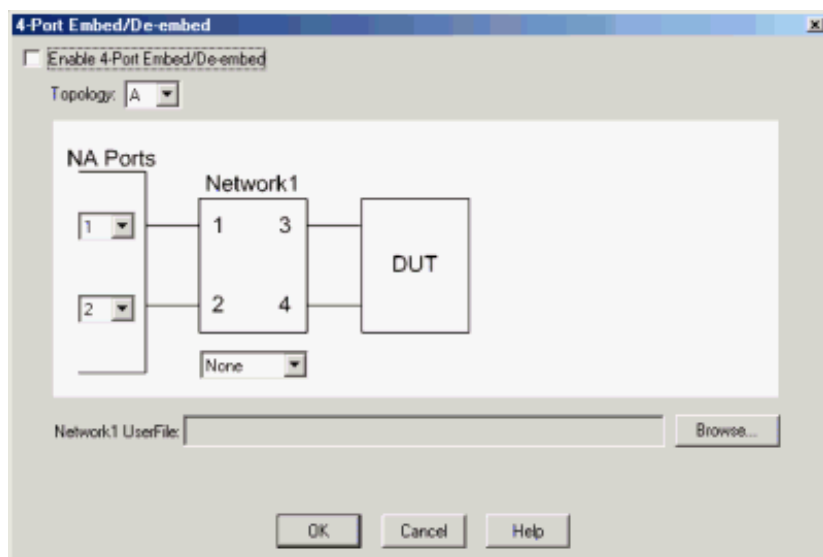
**Enable Port Z Conversion** Check to apply the settings to the measurement results. Must also enable [Fixturing ON/off](#).

**R** Real part of the impedance value.

**jX** Imaginary part of the impedance value.

**Close** Applies the entries and closes the dialog box

[See note about Port Impedance priority.](#)



#### 4-Port Embed/De-embed dialog box help

This function specifies a single-ended 4-port circuit (\*.S4P file) to embed (add) or de-embed (remove) from the measurement results. Computation takes place BEFORE Balanced conversion. [See Order of Fixture Operations.](#)

There is a single normalized impedance value for each port in the \*.S4P file. This impedance value must match the impedance of the previous Port Z setting, or the PNA port impedance.

The PNA will interpolate if the number of data points that are read is different from the current PNA setting.

**Enable 4-Port Embed/De-embed** Check to apply the settings to the measurement results. Must also enable [Fixturing ON/off.](#)

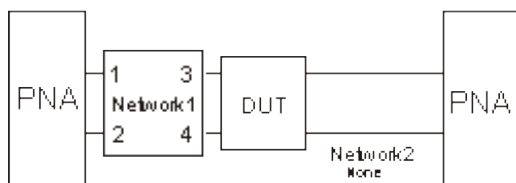
**Topology:** Select a DUT topology.

Refer to the images on 4-port embed/De-embed dialog box.

- **A** - 2 PNA/DUT Ports
- **B** - 3 PNA/DUT Ports
- **C** - 4 PNA/DUT Ports

#### Topology configurations that are not addressed with standard images in dialog box:

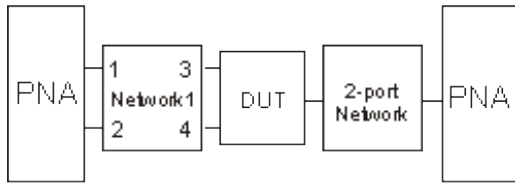
1. If you have a 4-port DUT; 4-port network on one side; None on the other side.



- Specify **Topology C.**
- Use 4-port Network on one side.

- Use 4-port Network on the other side; set to **None**.

2. If you have a 3-port DUT and networks as follows:



- Specify **Topology B**.
- Use 4-port Network1 on one side.
- Use [2-port network](#) on the other side.

**NA Ports** - Select the PNA Port that is connected to each circuit port.

**Note:** The \*S4P file always assumes that:

Network ports 1 and 2 are connected to the PNA

Network ports 3 and 4 are connected to the DUT

**None, Embed, De-embed** For Network1 and Network2, select:

- **None** - The same as disabling.
- **Embed** - Add the specified network circuit to the measurement results. [See 2-port Embed image](#).
- **De-embed** - Remove the specified network circuit from the measurement results. [See 2-port De-embed image](#).

**Browse** For both Network1 and Network2, navigate to find the \*.S4P file to embed or de-embed.

**OK** Applies the changes and closes the dialog box.

**Cancel** Does NOT apply the changes and closes the dialog box.



### Differential Impedance Conversion dialog box help

This function sets the Differential impedance value for each balanced port.

The default value for **R**: is the SUM of the impedance values for both ports that make the logical port. If [Port Z Conversion](#) is not enabled, then [System Z0](#) values for both ports are summed.

[See Order of Fixture Operations.](#)

**Enable Differential Z Conversion** Check to apply the settings to the measurement results. Must also enable [Fixturing ON/off](#).

**Logical Port** Select the logical (balanced) port to receive impedance value. To see logical port numbers, see the [measurement topology](#).

**R** Real part of the impedance value.

**jX** Imaginary part of the impedance value.

**Close** Closes the dialog box.

[See note about Port Impedance priority.](#)



### Common Mode Impedance Conversion dialog box help

This function sets Common Mode Impedance value for each balanced port.

The default value for **R**: is calculated as follows.

$$(Z1 * Z2) / (Z1 + Z2)$$

Where ports 1 and 2 comprise the logical port:

Z1 = the Port Impedance values for port 1

Z2 = the Port Impedance values for port 2

If [Port Z Conversion](#) is not enabled, then [System Z0](#) values for port 1 and 2 are used in the calculation.

[See Order of Fixture Operations.](#)

**Enable Common Mode Z Conversion** Check to apply the settings to the measurement results. Must also enable [Fixturing ON/off](#).

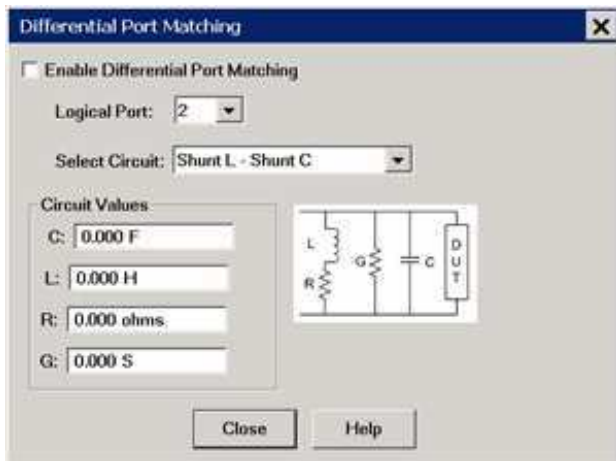
**Logical Port** Select the logical (balanced) port to receive impedance value. To see logical port numbers, see the [measurement topology](#).

**R** Real part of the impedance value.

**jX** Imaginary part of the impedance value.

**Close** Closes the dialog box.

[See note about Port Impedance priority.](#)



### Differential Port Matching dialog box help

This function allows the embedding of a differential matching circuit at a balanced port.

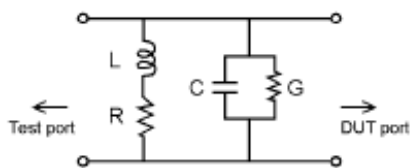
[See Order of Fixture Operations.](#)

**Enable Differential Port Matching** Check to embed the selected matching circuit to the measurement results. Must also enable [Fixturing ON/off](#).

**Logical Port** Choose [Logical DUT port](#) to receive the selected matching circuit. To see logical port numbers, see the [measurement topology](#).

**Select Circuit** Select a matching circuit. Choose from:

- **Shunt L - Shunt C** Predefined circuit.



**Circuit Values** Choose from:

- **C** Capacitance value
- **G** Conductance value
- **L** Inductance value
- **R** Resistance value
- **User defined** Select an \*.S2P file that represents the matching circuit. Then click **Browse** to navigate to the \*.S2P file.

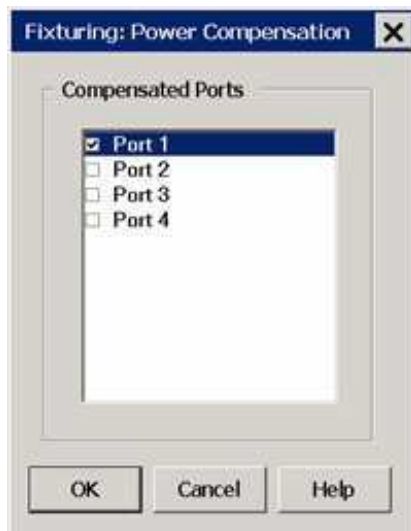
**Note:** For the \*.S2P file:  
Port 1 of the circuit is assumed to be connected to the PNA



Port 2 of the circuit is assumed to be connected to the DUT.

- **None** No embedded circuit on selected port.

**Close** Closes the dialog box.



#### Power Compensation dialog help

This function adjusts the source power at the specified port to compensate for the combined amount of gain or loss through ALL enabled fixturing operations. Use this function to set the power level at the DUT input.

For example:

- Your DUT requires a fixture on the input port which is connected to PNA port 1.
- The fixture description (such as an S2P file at the [2-port De-embed function](#)) indicates the fixture has approximately 2 dB of loss across the frequency span.
- You set source power to 0 dBm. But you want 0 dBm at the DUT input (the fixture output).
- Check Power Compensation on Port 1 and enable [Fixturing](#).
- Power Compensation causes the source power to be increased by approximately 2 dB so that the power at the fixture output plane will remain at 0 dBm.

Power Compensation affects all measurements in the channel.

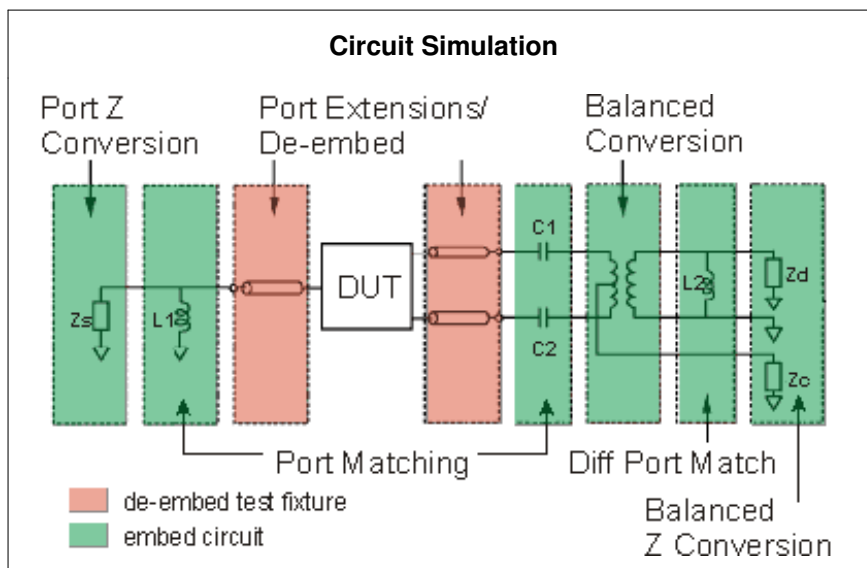
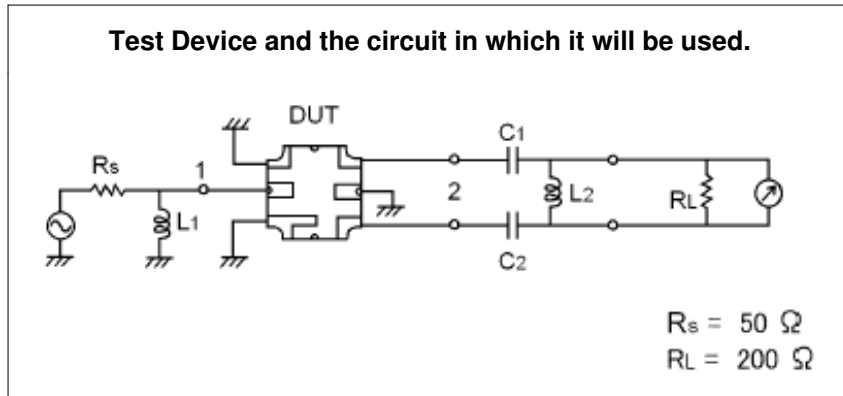
Enable [Fixturing](#) to use Power Compensation.

**Note:** Use caution when applying power compensation. Always test your setup without a DUT in a place. If you are using S2P files, [Recall](#) your S2P file into the PNA so you can verify that the device your S2P file describes is what you intended it to be. It is too easy to misalign data in S2P files if they are constructed by hand.

#### Fixture Simulator Example

The following example shows a DUT and the matching circuit with which the DUT will be used in its intended application. When the DUT is tested in a high-volume manufacturing environment, multiple test fixtures are often required. The most accurate way to test the DUT and ensure measurement consistency between the different test fixtures is to use a simple, repeatable, test fixture without the actual matching elements.

To get the desired performance data, the parasitic effects of the fixture must first be removed (de-embedded) from the measured data. Then a perfect "virtual" matching circuit must be simulated and added mathematically (embedded) to the corrected, measured data. The result is an accurate display of the DUT as though it was actually tested with a physical matching circuit, but without the uncertainties of using real components.



This diagram does NOT refer to the order in which operations are performed.

[See Order of Fixture Operations.](#)

1. [Create a balanced measurement](#) using single-ended to balanced (SE-Bal) [topology](#). Include all relevant measurement settings (IFBW, number of points, and so forth). Once the measurement is created and calibrated, the measurement parameter can be easily changed. For example, Sdd22 to Sds21.
2. Calibrate the measurement at the point where the simple test fixture is connected to the PNA. Use accurate calibration standards and definitions.
3. Remove the effects of the three uncalibrated transmission lines of the simple test fixture. This can be done in

several different methods. The easiest is to use manual or automatic [Port Extensions](#) to move the calibration reference plane to the DUT. This removes the electrical length and loss of the fixture's transmission lines, but does not account for fixture mismatch. Another method is to de-embed previously-created \*.S2p files of the 3 transmission lines. The files can be created using external ADS modeling software. Another alternative is to create the \*.S2P files by independently measuring all 3 ports of the test fixture and [saving the results of each to an S2P file](#).

4. With the test fixture connected to the PNA and a DUT inserted, the measurement results now appear as though calibration was performed at the connections to the DUT, and the device was measured in a 50-ohm single-ended test environment. The following steps will cause the results to reflect the performance of the device as though the device is embedded in the circuit in which it will be used.
5. Port 1 of the device is a single-ended port and sees a source impedance the same as the PNA system impedance, so no change is required. However, if Rs were a value other than 50 ohms, [Port 1 Impedance Conversion](#) would be used to simulate the different impedance.
6. [Port Matching](#) is used to simulate L1 inductance. Select any of the Shunt L circuits to embed (add) to the measurement results. Enter the value of L and R. The C and G values can be entered as 0 (zero).
7. [Port Matching](#) is used to simulate C1 and C2 capacitance. For both port 2 and port 3, select any of the **Series C** circuits to embed (add) to the measurement results. Enter the value of C and G. The L and R values can be entered as 0 (zero).
8. [Balanced Conversion](#) mathematically simulates the measurement in balanced mode.
9. [Differential Port Matching](#) is used to simulate L2 inductance. Select Shunt L- Shunt C and enter the inductance / resistance value. The C and G values can be entered as 0 (zero).
10. Finally, [Differential Z Conversion](#) is used to simulate a circuit termination of 200 ohms. If you are making Common Mode measurements, specify [Common Mode Z Conversion](#).

---

Last modified:

12-Apr-2010	Updated for A.09.20
11-Nov-2009	Add SNP note
2-Feb-2009	Reorder and add CAN change first four
29-Sep-2008	Added note about standard channel
3-Sep-2008	Removed legacy content
Sept 12, 2006	Added link to programming commands

## Port Extensions

---

Port extensions allow you to electrically move the measurement reference plane after you have performed a calibration.

**Note:** Beginning with PNA Rev. A.09.20, this feature is available to the following [measurement classes](#): [GCA](#), [Noise Fixture](#), [NFX](#), and standard (S-Parameter) channels.

- [Why and How to use Port Extensions](#)
- [Manual Port Extensions Procedure](#)
- [Port Extensions dialog and Toolbar](#)
- [Step Size dialog](#)
- [Automatic Port Extension dialog](#)

### See Also

[Fixture Compensation features](#)

[Phase Accuracy](#)

[Comparing the PNA Delay Functions](#)

### Other Calibration Topics

### Why to use Port Extensions

1. You are unable to perform a calibration directly at your device because it is in a test fixture. Perform a calibration at a convenient place, then use port extensions to compensate for the time delay (phase shift), and optionally the loss, caused by the fixture.
2. You have already performed a calibration, and then decide that you need to add a length of transmission line in the measurement configuration. Use port extensions to "tell" the analyzer you have added the length to a specific port.

### How to use Port Extensions

- If you know the **electrical length** of the fixture or additional transmission line, enter the value directly to the **Time** setting.
- If you know the **physical length** of the fixture or additional transmission line, enter the value directly to the **Distance** setting.
- If you do **NOT** know either the electrical or physical length of the fixture or additional transmission line, you must be able to connect an OPEN or SHORT to the new reference plane - in place of the DUT. In most cases, removing the DUT will leave a suitable OPEN at the new reference plane.

- Port Extensions can then be added manually (as follows), or by using [Automatic Port Extensions](#).

## Manual Port Extensions Procedure

1. Select a calibrated S11 measurement.
2. Select Phase format.
3. With an OPEN or SHORT at the calibration reference plane, verify that the phase across the frequency span is at or near zero.
4. Connect the fixture or added transmission line and attach an OPEN or SHORT in place of the DUT. In most cases, removing the DUT will leave a suitable OPEN at the new reference plane. On the Port Extension toolbar or dialog, increase either **Time** or **Distance** until the phase response is flat across the frequency span of interest.
5. If you know the loss of the additional transmission line, enter the [Loss Compensation](#) values using either one or two data points.

**Note:** Most OPEN and SHORT standards have delay. Therefore, adjusting delay with this method results in a delay equal to two times the delay of the OPEN or SHORT.

### How to access Port Extensions settings

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Port Extensions]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Port Extension**

◀ **Programming Commands** ▶

**Port Extensions**

Port 1 | Port Extensions ON | Show Toolbar

**Delay**

Dist: 0  $\mu$ m | Time: 0 ps | Distance Units: Meters

**Velocity**

Velocity Factor: 1.000000000 | ☒ Couple to system Velocity Factor

**Loss**

Loss at DC: 0.000000 dB

☐ Loss1: 0  $\mu$ dB @ Freq1: 1.000 GHz

☐ Loss2: 0  $\mu$ dB @ Freq2: 1.000 GHz

**Media**

☒ Coax | ☐ Waveguide | Port Cutoff Freq: 45.000 MHz

☒ Couple to system Media definition

Reset | Auto... | OK | Cancel | Help



## Port Extensions dialog and Toolbar help

**Note:** Beginning with PNA Rev. A.09.20, this feature is available to the following [measurement classes](#): [GCA](#), [Noise Fixture](#), [NFX](#), and standard (S-Parameter) channels.

Port extensions settings affect all measurements on the active channel that are associated with a particular port.


[Learn Why and How to use Port Extensions](#) (scroll up).

**Port Extension** Turns ON and OFF port extensions on all ports.

**Port** Select a PNA port for delay and loss values. Port Extensions settings affect ALL measurements on the active channel that are associated with a particular port.

**Show Toolbar** Check to show the Port Extensions toolbar. The toolbar allows you to make adjustments to the port extensions while showing more of the PNA screen. Beginning with PNA rev A.08.50, this is the only way to show or hide the toolbar.

### Delay settings

Enter delay in either Distance or Time by entering a value or clicking the up/down arrows. Click  to start the [Step Size](#) dialog.

**Time** The amount of port extension delay in time. Enter a positive value.

**Distance** The amount of port extension delay in physical length. Enter a positive value.

**Distance Units** (Dialog ONLY) Select from Meters, Inches, or Feet. The Step Size setting will not change automatically. [Learn more.](#)

### Loss Compensation

The following settings allow the entire frequency span to be corrected for loss.

**Loss at DC** Offsets the entire frequency span by this value. Loss1 or Use1 must also be checked. To compensate for loss at DC, enter a positive value which causes the trace to shift in the positive (up) direction.

**Loss @Frequency** Check the box, and enter values for Loss and Frequency

When **Loss1** or **Loss1/Loss2** are used, a curved-fit algorithm is used as follows:

**Loss1 ONLY:**

$$\text{Loss}(f) = \text{Loss1} * (f/\text{Freq1}) ^ 0.5$$

**Loss1 and Loss2:**

Set the lower frequency to Loss1, and the higher frequency to Loss2.

$$\text{Loss}(f) = \text{Loss1} * (f/\text{Freq1}) ^ n$$

Where:

$$n = \log_{10} [\text{abs}(\text{Loss1}/\text{Loss2})] / \log_{10} (\text{Freq1}/\text{Freq2})$$

**Note:** abs = absolute value

### Velocity

**Velocity Factor** For each port, sets the velocity factor that applies to the medium of the device that was inserted after the measurement calibration. The value for a polyethylene dielectric cable is 0.66 and 0.7 for Teflon dielectric. 1.0 corresponds to the speed of light in a vacuum.

**Couple to system Velocity Factor** When unchecked, the Velocity Factor is set for only the specified port and only for Port Extensions. When checked, sets the Velocity Factor for all ports. In addition, changing this value also changes this setting for the [Electrical Delay](#) and [Time Domain Distance Marker](#) features.

## Media

For each port, select the media of the added transmission line or fixturing.

**Coax** Select when the fixture or added transmission line is coax. Also specify the velocity factor of the coax.

**Waveguide / Cutoff Frequency** Select when the fixture or added transmission line is waveguide. Also enter cutoff (minimum) frequency of the waveguide.

**Note:** when using a Waveguide cal Kit, set [System Z0](#) to 1 ohm before calibrating.

**Couple to system Media Definition.** When unchecked, the Waveguide Cutoff Frequency is set for only the specified port and only for Port Extensions. When checked, sets the Waveguide Cutoff Frequency for all ports. In addition, changing this value also changes this setting for the [Electrical Delay](#) feature.

**Reset** All port extensions settings are changed to preset values. The Port Extension ON / OFF state is NOT affected.

**Auto Ext.** Starts the [Automatic Port Extensions](#) dialog box

**Note:** Individual receiver port extensions (A,B, and so forth) can no longer be set. (Sept. 2004)



## Step Size dialog box help

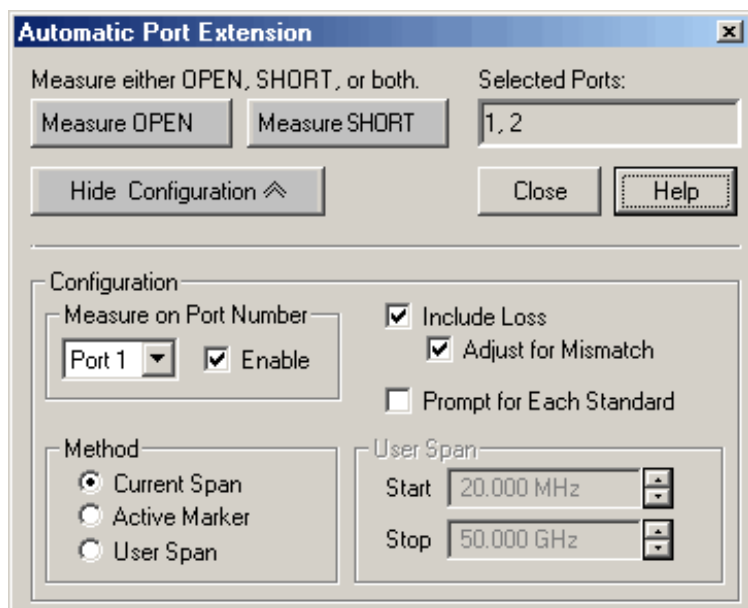
Changes the step size that occurs when the Time or Dist up/down arrows are pressed on the Port Extension toolbar. The Units for step size are changed on the Port Extension dialog.

**Auto** Step Size is set to the default value.

**User Defined** Enter a step size value, then click OK.

This value remains the same when the units are changed. For example if a step size of 12 is entered on this dialog, then you change the units from Inches to Feet, the step size of 12 inches becomes 12 feet, not 1 foot. Therefore, change the units first, then set the step size.

Learn about [Port Extensions](#) (scroll up)



### Automatic Port Extension dialog box help

Automatic Port Extension AUTOMATICALLY performs the same operation as [Manual Port Extension](#). By connecting a SHORT or OPEN, the reference plane is automatically moved to the point at which the standard is connected. In addition, Automatic Port Extension will optionally measure and compensate for the loss of the additional transmission line.

Auto Port Extension is NOT available when:

- Sweep type is set to power sweep
- Frequency Offset is ON
- Media is set to Waveguide

**Note:** Turn OFF [Equations](#) that may exist on the active trace when using Automatic Port Extensions.

### Auto Port Extensions Procedure

1. Connect the added transmission line or fixture. Attach an OPEN or SHORT to all affected ports at the new reference plane. In most cases, removing the DUT will leave a suitable OPEN at the new reference plane.
2. On the Port Extension toolbar, click **Auto Port Ext.** Click **Show Configuration** to make additional settings.
3. Click **Measure** to perform the port extension calculations. The resulting delay and loss settings are entered into the port extension toolbar. These settings are saved with Instrument Save or you can manually record the values and enter them again when required.

### Settings

**Measure either OPEN, SHORT, or both** Press a button to make the measurement of the reflection standard.



Measure either OPEN or SHORT depending on which is most convenient. An ideal OPEN and SHORT, with zero loss and delay, is assumed. Therefore, accuracy is most affected by the quality of the standard. In most cases, removing the DUT will leave a suitable OPEN at the new reference plane. When measuring both OPEN and SHORT standards, the average of the two is used and will slightly improve accuracy.

**Selected Ports** Indicates the ports that currently have automatic port extension enabled. By default, ALL PNA ports are enabled. To disable a port, see **Measure on Port Number** below.

**Note:** Port Extensions settings affect ALL measurements on the active channel that are associated with a particular port.

**Show/Hide Configuration** Press to either show or hide the following configuration settings in the dialog box.

### Measure on Port Number

Select port number to enable or disable automatic port extension.

**Enable** Check to enable the specified port. All enabled ports will have their reference plane automatically adjusted after performing Automatic Port Extension.

**Include Loss** Check to automatically measure the loss in the additional transmission line and apply compensation. To calculate loss compensation, frequencies at 1/4 and 3/4 through the frequency range are usually used as Freq1 and Freq2 values. [Learn more about Loss Compensation.](#)

**Adjust for Mismatch** Only available when **Include Loss** is checked. During the measurement of the OPEN or SHORT standard, mismatch could cause ripple in the magnitude (loss) response. The [Loss compensation curved-fit algorithm](#) allows half of the ripple to be positive and half negative. When measuring low-loss devices, it is possible that some magnitude responses could become slightly positive, indicating gain rather than loss.

Check - Offsets the trace to cause all of the data points to be at or below zero.

Clear - Most accurate application of the curve-fit calculation, but allows positive responses.

**Prompt for Each Standard** Check to invoke a prompt when the Measure OPEN or SHORT button is pressed. The prompt will indicate which standard to connect to which port.

### Method

Select the span of data points which will be used to determine correction values for phase and loss (optional). If a portion of the current frequency span does not have flat or linear response, you can eliminate this portion from the calculations by using a reduced User Span.

To calculate loss compensation, Current Span and User Span methods usually use frequencies at 1/4 and 3/4 through the frequency range as Freq1 and Freq2 values. See [Loss Compensation](#) to learn more about how loss is calculated.

**Current Span** Use the entire frequency span to determine phase and loss values.

**Active Marker** Use only the frequency at the active marker, and one data point higher in frequency, to calculate phase and loss values. If a marker is not present, one will be created in the center of the frequency span.

**User Span** Use the following User Span settings to determine phase and loss values.

### User Span

**Start** Enter start frequency of the user span.

**Stop** Enter stop frequency of the user span.

Learn about [Port Extensions](#) (scroll up).

See also [Comparing the PNA Delay Functions](#)

---

Last modified:

18-May-2009	Added note about equations
5-Feb-2009	Added many features (A.08.50)
3-Sep-2008	Removed legacy content
9/12/06	Added link to programming commands

## Characterize Adaptor Macro

---

This external [Macro](#) application creates an S2P file that models a device such as an adaptor, the input OR output side of a test fixture, or an on-wafer probe head. This is done by calculating the four S-parameters of the device from two 1-port calibrations. Such S2P files can be used for embedding or de-embedding the device from S-parameter measurements and FCA calibrations.

This application, along with the FCA [Embed/De-embed feature](#), can be especially useful when performing FCA calibrations.

- An SMC calibration requires a [power meter measurement](#) at the port 1 reference plane. This could be very difficult in on-wafer applications where the measurement reference plane is at the tip of a probe. This macro, in conjunction with the FCA Embed/De-embed feature, enables you to model the probe and connect the power sensor at the coax connector where the probe connects.
- Likewise, a VMC calibration requires that a [calibration mixer](#) be used for the Thru standard. Again, this can be very difficult in on-wafer applications where the measurement reference plane is at the tip of a probe. This macro, in conjunction with the FCA Embed/De-embed feature, enables you to model the probe and connect the calibration mixer at the coax connectors where the probe connects.

### New: Characterize Adaptor Macro - Version A.02.10

- [Reverse S2P files](#)
- [Loads the PNA Power Loss Table from an existing S2P file.](#)
- This macro update requires PNA firmware version A.08.20.04 or higher

### For S-Parameter measurements:

- [To Embed or De-embed](#)
- [Procedures](#)

### How to start the Characterize Adaptor Macro

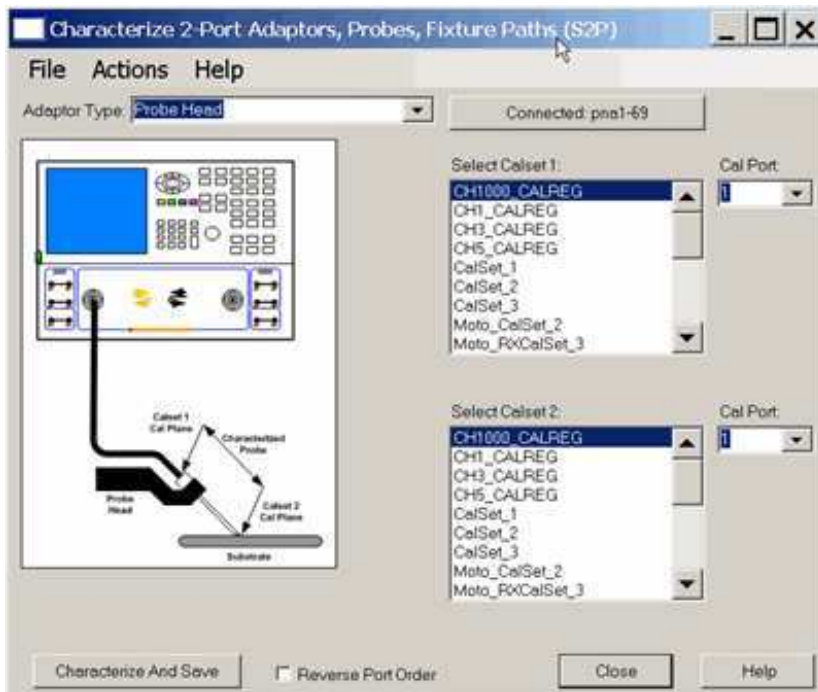
#### Using front-panel HARDKEY [softkey] buttons

1. Press **MACRO**
2. then **[AdaptorChar]**

#### Using a mouse with PNA Menus

1. Click **Utility**
2. then **Macro**
3. then **AdaptorChar**

◀ **Programming Commands** ▶



## Characterize 2-Port Adaptors, Probes, Fixture Paths (S2P) dialog box help

### Important Notes

- The device to be characterized (probe, adapter...) MUST be reciprocal ( $S_{21} = S_{12}$ ).
- Two 1-port cals must be performed and saved to Cal Sets BEFORE using the Characterize Adaptor application.
- The frequencies and number of points of the two Cal Sets MUST be identical.
- **CRITICAL:** The calculations that are performed to create the S2P file require that **Calset 1** ALWAYS be from the side closest to the PNA and **Calset 2** ALWAYS be from the other side of the device. If your application that uses the resulting S2P file requires that the ports be reversed, check **Reverse Port Order**. Or learn how to [reverse the port order on an existing S2P file](#).
- The majority of this topic describes the characterization of a 2-port device from two 1-port cal sets. However, you can also use the cal sets from two 2-port calibrations, or one 2-port and two 1-port calibrations. See procedures for both.

[Learn more about the Characterize Adaptor Macro.\(scroll up\)](#)

**Connected** <PNA host name> The two 1-port Cal Sets can reside in another PNA. Click to connect to another PNA that is DCOM configured. [Learn how to configure DCOM](#).

**Adaptor Type** Select the type of device to be characterized.

**Note:** The image that appears in the macro does not influence the calculations. It only appears to help you visualize the measurement reference plane of the Cal Sets.

**Select Calset 1 and Calset-2** Select a 1-port Cal Set from each list. Although all Cal sets are listed, only the Cal Sets that have error terms to satisfy a 1-port calibration may be used.

**Cal Port** Select the port within the selected Cal Set which represents the modeled device. The Cal Ports must be the same for both selected Cal Sets.

**Characterize and Save** Calculates four S-parameters, then invokes the [Save As dialog](#) with [S2P file type](#). This button is not available until valid Cal Sets and Cal Ports are selected.

**Reverse Port Order** By default, the calculations that are performed to create the S2P file require that **Calset 1** ALWAYS be from the side closest to the PNA and **Calset 2** ALWAYS be from the other side of the device. Check **Reverse Port Order** to cause the ports to be reversed in the resulting S2P file.

## Reverse S2P

Click **Actions**, then **Reverse S2P File**

This action causes ports to be reversed on an existing S2P file. For example:

- The data for S11 becomes the data for S22 and vice versa.
- The data for S21 becomes the data for S12 and vice versa.

The resulting file is written in the standard [PNA S2P file format](#).

1. Navigate to the S2P file to be reversed.
2. Navigate to where the new reversed S2P file will be saved. By default, the file is saved to the same folder as: [old filename]\_Reversed.s2p.
3. Click **OK**.

## Write to Power Loss Table

Click **Actions**, then **Write to PNA Pwr Loss Table**.



### Power Loss Table Setup dialog box help

Loads the S2P Frequency / Loss pairs into the PNA [Power Loss Compensation table](#) to compensate for losses that occur when using the device to connect a power sensor to the measurement port during a Source Power Cal.

**Note:** PNA firmware revisions prior to A.08.33.07 ONLY support up to 100 segments in the power loss table. Therefore, to save data to the power loss table in one of the earlier firmware revs, then the Cal Sets that created the S2P file must contain 100 data points or less. Revision A.08.33.07 and above allows any number of segments.

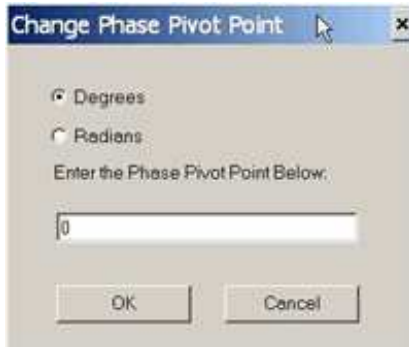
**Use Selected Calsets** Computes the transmission loss of the fixture based on the selected Cal Sets. This choice is NOT available until two valid Cal Sets are selected.

**Use S2P File** Uses the S21 data in an existing S2P file to build the PNA's power loss table. Select, then click ... then navigate to the S2P file, then click **OK**.

**Note:** In the PNA Power Loss Compensation table, loss is expressed as a positive number. The macro assumes that any negative S21 value in the S2P file is a loss and therefore multiplies the S21 values in the file by -1 to express that value as a positive number. This ensures proper handling of the offset during a source power cal.

### Change Phase Pivot Point

Click **File**, then **Set Phase Pivot Point**.



### Change Phase Pivot Point dialog box help

Before the A.01.50 revision (see Help, About) the Characterize Adaptor macro projected the phase of S21 at DC to cross the X-axis between 0° and -180°, based on the delay of the adapter. For most adapters the value is 0°.

However, when characterizing electrically long cables, cables with significant mismatch, or high noise in the measurements, it is possible that the projection of phase goes above 0°. This results in a 180° phase difference between the results computed by the macro versus the results you might get by measuring the same adapter with a 2-port calibration.

This revision allows you to adjust the center of the 180° phase pivot window. The default value 0° yields a window between +90° to -90° which should be adequate for the majority of adapters.

Enter the value in degrees or radians. This value remains while the application is running.

Restarting the application will result in the Phase Pivot Point to revert back to 0°, the default value.

## To Embed or De-embed S-Parameter Measurements

- For FCA measurements, see [FCA Embed or De-embed](#).
- In this section, the term “adapter ” can mean any type of 2-port device and its associated S2P file.

To make an accurate measurement, the setup configuration during the DUT measurement MUST exactly match the setup configuration during Calibration. In other words, if you calibrate with an adapter, you must also measure the DUT with the adapter.

However, the PNA provides some flexibility by allowing you to ‘**Virtually**’ add (embed) or remove (de-embed) an adapter from the measurement. Knowing how to do this can be confusing.

- To perform a calibration **WITHOUT** the adapter, but make DUT measurements **WITH** the adapter, then **De-embed** (remove) the adapter from the DUT measurement.
- To perform a calibration **WITH** the adapter, but make DUT measurements **WITHOUT** the adapter then **Embed** (add) the adapter during the DUT measurement.

## Procedures

- [Create an S2P file using Characterize Adaptor Macro](#)
- [De-embed the S2P file from DUT measurement](#)
- [Embed the S2P file in DUT measurement](#)
- [De-Embedding a Fixture that has a THRU Standard](#)
- [De-Embedding a Fixture with No THRU Standard](#)

## Create an S2P file using the Characterize Adaptor Macro

1. Configure your PNA measurement (frequency span, power level, IF bandwidth, and number of points).
2. Perform a 1-port SmartCal at the reference plane. Save the cal to a User Cal Set using a descriptive name (for example, **Ref Plane**).
3. Connect the adapter to be characterized at the reference plane.
4. Perform another 1-port SmartCal at the end of the adapter. Save it to a User Cal Set using a different descriptive name (for example, **Adapt End**).
5. [Start the Characterize Adaptor Macro](#).
6. In the **Select Calset1** field of the dialog box, select the Cal Set for the reference plane (from step 2 above).
7. In the **Select Calset2** field of the dialog box, select the Cal Set for the end of the adapter (from step 4 above).
8. Click **Characterize and Save**. In the resulting dialog box, enter the .S2P file name and location.
9. Click **Close**.

## De-embed the adapter (S2P file) from subsequent S-parameter measurements

See [Fixture De-embedding](#)

**Note:** Subsequent measurements must have the same or smaller frequency range (within the Start / Stop frequencies) as that of the S2P file.

1. Perform a 2 port SOLT calibration **without** the adapter/fixture.
2. Select 2-port De-embedding: click **Response**, then **Cal**, then **More**, point to **Fixtures**, then click **2 port De-embedding**.
3. Select the Port to add the adapter to, then select **User Defined (S2P file)**.
4. Click **Use S2P file** and select the S2P file created using the Characterize Adaptor macro.
5. Check **Enable De-embedding**, then click **Close**.
6. Enable Fixturing: click **Response**, then **Cal**, then **More**, point to **Fixtures**, then click **Fixturing on/OFF**.
7. **Sim** appears in the Status Bar to indicate that Fixture Simulation is ON.

## Embed the adapter (S2P file) into subsequent S-parameter measurements

The adapter (S2P file) can also be a matching network. [Learn more](#).

1. Perform a 2 port SOLT calibration **including** the adapter. Note the port number on which the adapter is calibrated.
2. Select Port Matching: click **Response**, then **Cal**, then **More**, point to **Fixtures**, then click **Port Matching**



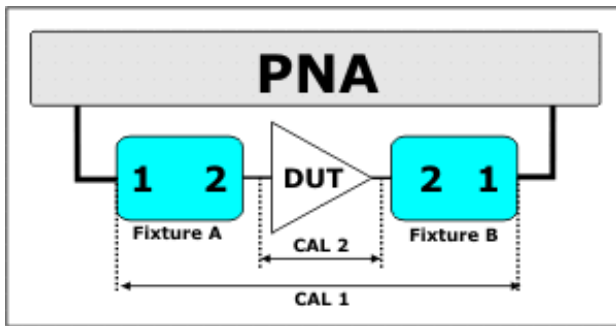
3. Under Choose Circuit Model for Matching, select the Port that the adapter was on during calibration, then select **User Defined (S2P file)**.
4. Press **Use S2P File** and navigate to the S2P file created using the Characterize Adaptor macro.
5. Check **Enable Port Matching**, then click **Close**.
6. Enable Fixturing: click **Response**, then **Cal**, then **More**, point to **Fixtures**, then **Fixturing on/OFF**.
7. **Sim** appears in the Status Bar to indicate that Fixture Simulation is ON.

### De-Embedding a Fixture that has a THRU Standard

A test fixture is generally regarded as a single 'bed' in which a DUT is placed. However, for modeling purposes such as this, it is separated into two circuits: Fixture A on the input of the DUT, and Fixture B on the output.

Use this procedure to perform calibrations **WITHOUT** the test fixture while making measurements **WITH** the test fixture. A calibration is performed once **WITH** the test fixture, and then again as it wears with use and electrical performance changes. The fixture is de-embedded from subsequent measurements to match the regular calibrations that are performed without the fixture.

If you have a THRU standard for your test fixture, you can perform a full 2-port calibration in the fixture, and from that create the required S2P files for de-embedding.



1. Perform a full 2-port **CAL 1** at the connections of the PNA to the fixture as shown above. Save to **MyCalSet1**.
2. Perform a full 2-port **CAL 2** where the DUT is inserted (reference plane). Save to **MyCalSet2**.

Follow the [Create an S2P file](#) procedure, beginning with step 6, using the following selections:

1. Create #1 S2P file:
  1. For CalSet1, choose **MyCalSet1** and select **CalPort=1**
  2. For CalSet2, choose **MyCalSet2** and select **CalPort=1**
  3. Save to **FixtureA.s2p**
2. Create #2 S2P file:
  1. For CalSet1, choose **MyCalSet1** and select **CalPort=2**

2. For CalSet2, choose **MyCalSet2** and select **CalPort=2**

3. Save to **FixtureB.s2p**

Follow steps in [To De-embed the adapter...](#)

Perform these steps TWICE; once for each of the following S2P files:

1. For PNA Port 1, select **FixtureA.s2p**

2. For PNA Port 2, select **FixtureB.s2p**

### De-Embedding a Fixture with No THRU Standard

This procedure is a slight modification of the above. Cal 2 is performed from two 1-port cals when a THRU standard for the fixture is not readily available.

1. Perform a full 2-port **CAL 1** at the connections of the PNA to the fixture as shown above. Save to **MyCalSet1**.

2. **CAL 2** is performed using two 1 port cals

- **Cal2A** at the Fixture A / DUT plane. Save to **MyCalSet2A**

- **Cal2B** at the Fixture B / DUT plane. Save to **MyCalSet2B**

Follow steps in [Create an S2P file...Step 6](#) above, except:

1. Create #1 S2P file:

1. For CalSet1, choose **MyCalSet1** and select **CalPort=1**

2. For CalSet2, choose **MyCalSet2A** and select **CalPort=1**

3. Save to **FixtureA.s2p**

2. Create #2 S2P file:

1. For CalSet1, choose **MyCalSet1** and select **CalPort=2**

2. For CalSet2, choose **MyCalSet2B** and select **CalPort=2**

3. Save to **FixtureB.s2p**

Follow steps in [To De-embed the adapter](#) above, except:

1. For PNA Port 1, select **FixtureA.s2p**

2. For PNA Port 2, select **FixtureB.s2p**

---

Last modified:

25-Mar-2010	Added support for version 2.1
23-Mar-2009	Fixed error in "De-embedding a fixture..." procedures
8-Dec-2008	Added support for version 2.0
3-Sep-2008	Removed legacy content
10-Mar-2008	Added Change Pivot dialog
13-Mar-2008	Moved <a href="#">FCA embed or de-embed</a>
12-Feb-2008	New procedures
24-Jan-2008	Fixed error in procedures and added section
30-Nov-2007	Clarified and highlight order of calsets.
26-Feb-2007	Fixed " Note: Subsequent measurements...".
12-Sept-2006	Added link to programming commands

## Delta Match Calibration

---

A [TRL Cal](#), [QSOLT](#), or [Unknown Thru](#) Cal requires a [reference receiver for each test port](#). The 4-port PNA-L model does NOT have a reference receiver for each test port.

A Delta Match Calibration can be thought of as a software method which provides a reference receiver for each test port when not otherwise available in the hardware. The Delta Match Calibration measures the [source match](#) and [load match](#) of the PNA test ports, and then calculates the differences, or "delta", of the two match terms. The results are then used to correct subsequent TRL, QSOLT, or Unknown Thru calibrations.

There are several ways to acquire the Delta Match Calibration:

1. **From an existing User Cal Set** that meets the following Delta Match criteria: (Not allowed for use with [external test sets](#).)
  - Must have been performed using ECal or as a guided mechanical Cal (not Unguided).
  - Must have the same start frequency, stop frequency, and number of points as the channel being calibrated.
  - Must calibrate the ports that require the delta match terms.
2. **From a [Global Delta Match Calibration](#).**
3. **From a 'Self Delta Match'** when other portions of the calibration fully characterize all ports using SOLT with [Defined Thru](#) or [Flush Thru](#). For example, when calibrating all four ports of a PNA-L, perform a SOLT between ports 1 and 2, and also between ports 3 and 4, then Unknown Thru could be used between any combination of the remaining ports. This is allowed with an [external test set](#).

**Which to use?** A Self Delta Match Cal will always be used when possible. Otherwise, the Cal Wizard will use a Global Delta Match Cal when available unless you select [Choose Delta Match](#).

### Global Delta Match Cal

A Global Delta Match Cal is an "all-inclusive" calibration that can be applied whenever the delta match terms are required.

A Global Delta Match Cal differs from a standard SOLT Cal in the following ways:

- It is always performed using a Flush Thru, a Known Thru, or an insertable ECal module. You can NOT use an Unknown Thru in the calibration process.
- Only two Thru connections are required to characterize the delta match terms on a 4-port PNA. This is less than the minimum number of Thrus of a standard 4-port Cal.
- Upon completion, the Global Delta Match Cal is stored as a special type of Cal Set and should be used ONLY as a Delta Match Cal. It provides Delta Match error terms, but does NOT provide all of the standard error correction terms.
- To attain the highest accuracy, the following settings are automatically used to perform a Global Delta Match Cal. When applied, it will likely be [interpolated](#).

- Performed over the entire frequency range of the PNA.
- Uses very dense data points, particularly at low frequencies.
- Uses 100 Hz IF Bandwidth.

**Note:** For highest accuracy, perform Global Delta Match Cal using an insertable ECal module and select [Flush-thru](#) as the Calibration Thru method.

### How to perform a Global Delta Match Cal

These selections will only be available if the PNA hardware requires a Delta Match Calibration.

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Start Cal]**
3. then **[Global Delta Match]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**
3. then **Start Cal**
4. then **Global Delta Match**

### Programming Commands



#### Delta Match Calibration. Select DUT Connectors and Cal Kit dialog box help

- Only one Cal Kit is specified and necessary to perform a Delta Match Cal. However, ALL of the PNA test ports are calibrated in a Delta Match Cal.
- You must configure ALL test ports to terminate in the specified connector / gender using the necessary adapters. The errors from adapters are removed during calibration, but the Thru connections must be made as specified.
- **If you select an ECal module that does NOT cover the entire frequency range of the PNA**, your selection will change to a different Cal Kit. The Global Delta Match Cal covers the entire frequency range of the PNA. Your selected Cal Kit or ECal module must also cover the frequency range of the PNA.



#### Guided Calibration Steps dialog box help

Click **Measure** for each standard.

When all standards have been measured, click **Done** to complete the measurement steps.



#### Delta Match Calibration Complete dialog box help

Click **Finish** to store the Global Delta Match Calibration as a special type of Cal Set.

By default, it will be used when a Delta Match Calibration is required.

It should ONLY be used as a Delta Match Cal. It does NOT provide all of the standard error correction terms.

Last modified:

- 3-Sep-2008    Removed legacy content
- 9-Nov-2007    Edits for requirements
- 23-Feb-2007    Modified requirements for multiport
- 9/12/06    Added link to programming commands

## Markers

Markers provide a numerical readout of measured data, a search capability for specific values, and can change stimulus settings. There are 9 regular markers and one [Reference marker](#) (used with Delta markers) available per trace. This topic discusses all aspects of markers.

**Note:** Marker Readout can be turned ON / OFF and customized from the **View/Display** menu. See [Marker Readout](#)

- [Creating and Moving Markers](#)
- [Delta Markers](#)
- [Searching with Markers](#)
- [Marker Functions](#) (Change Instrument Settings)
- [Marker Table](#)

### Other Analyze Data topics

#### How to Create Markers

##### Using front-panel HARDKEY [softkey] buttons

1. Press **MARKER**
2. then **[Marker n]**

##### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Marker**
3. select a marker number

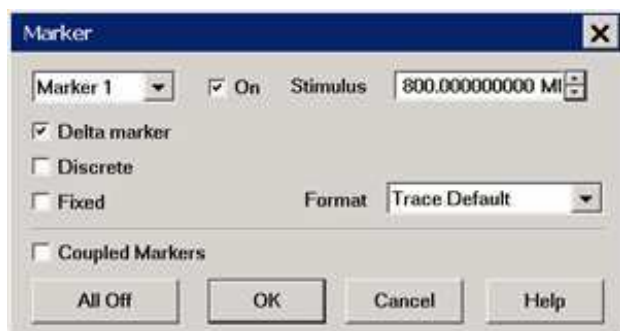
◀ [Programming Commands](#) ▶

#### Moving a Marker

To move a marker, make the marker active by selecting its number in any of the previous 3 methods. The **active marker** appears on the analyzer display as . All of the other markers are inactive and are represented on the analyzer display as . Then change the stimulus value using any of the following methods:

- Type a value.
- Scroll to a stimulus value using the up / down arrows. The resolution can not be changed.
- Click the stimulus box, then use the front-panel knob.
- Click and Drag Markers using a finger (touchscreen) or by left-clicking and holding a marker symbol. Then

drag the marker to any point on the trace. This feature is NOT allowed in Smith Chart or Polar [display formats](#) or with a [Fixed Marker type](#).



### Marker dialog box help

**Marker** Specifies the current (active) marker number that you are defining.

**On** Check to display the marker and corresponding data on the screen.

**Stimulus** Specifies the X-axis value of the active marker. To change stimulus value, type a value, use the up and down arrows, click in the text box and use the front-panel knob, or drag the marker on the screen.

**Delta Marker** Check to make the active marker display data that is relative to the reference (R) marker. There is only one reference marker per trace. All nine other markers can be regular markers or delta markers. When a delta marker is created, if not already displayed, the reference marker is displayed automatically. A delta marker can be activated from the [Marker dialog box](#) or the [Marker Toolbar](#).

**Discrete Marker** Check to display values at only the discrete points where data is measured. Clear to display values that are interpolated from the data points. The interpolated marker will report y-axis data from ANY frequency value between the start and stop frequency.

**Fixed** Check to cause the marker to have a fixed X-axis and **Y-axis** position based on its placement on the trace when it was set to fixed. It does NOT move with trace data amplitude. It can be scrolled left and right on the X-axis by changing the marker stimulus value. Use this marker type to quickly monitor "before and after" changes to your test device. For example, you could use fixed markers to record the difference of test results before and after tuning a filter.

Clear the box to create a **Normal** marker, which has a fixed stimulus position (X-axis) and responds to changes in data amplitude (Y-axis). It can be scrolled left and right on the X-axis by changing the marker stimulus value. Use this marker type with one of the marker search types to locate the desired data.

**Coupled Markers** Check to couple markers by marker number, 1 to 1, 2 to 2 and so forth. The markers will remain coupled until this box is unchecked. [Learn more about coupled markers.](#)

**Format** Displays the marker data in a format that you choose. The marker format could be different from the grid format. The Trace Default setting has the marker and grid formats the same.

**All Off** Switches OFF all markers on the active trace.

## Searching with Markers

You can use markers to search and return data for the following trace criteria:

- [Max and Min](#): find the highest or lowest points on the trace



- [Peak](#), then move to other peaks (left, right, next highest)
- [Target Value](#): find a specific Y-axis value
- [Bandwidth](#) (Filters)
- [Compression Point](#) (Amplifiers)
- [About PSAT and PNOP Markers](#)
  - [Power Saturation](#) (Amplifiers)
  - [Power Normal Operating Point](#) (Amplifiers)
- [What is a Peak?](#)
- [Search Domain](#)

If there is no valid data match for any of the search types, the marker will not move from its current position.

## How to Search with Markers

Using front-panel  
HARDKEY [softkey] buttons

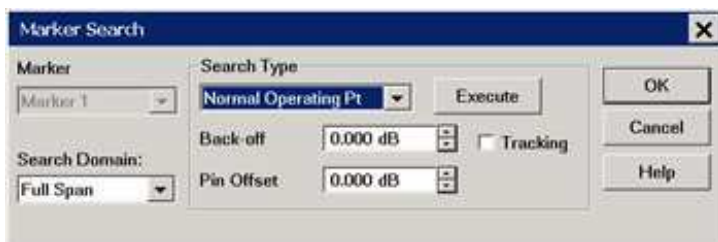
Using a mouse with PNA Menus

For [PNA-X](#) and ['C'](#) models

◀ **Programming Commands** ▶

1. Press **SEARCH**

1. Click **Marker/Analysis**  
2. then **Marker Search**



## Marker Search dialog box help

**Marker** Specifies the marker that you are defining. Not available for search types that deploy specific markers.

**Search Domain** Defines the area where the marker can move or search. For full span, the marker searches for specified values within the full measurement span. For user span, the marker searches for specified values within a measurement span that you define. [Learn more about Search Domain.](#)

## Search Type

**Note** You must either press **Execute** or check **Tracking** to initiate all search types.

- **Execute** Click to cause the marker to search for the specified criteria.

- **Tracking** Check to cause the marker to search for the specified criteria with each new sweep. The searches begin with the first sweep after Tracking has been checked, based on the current search type and domain information. Therefore, make sure that the search criteria are in the desired state before using the data. You cannot manually change the stimulus setting for a marker if Tracking is selected for that marker.

**Maximum** Marker locates the maximum (highest) data value.

**Minimum** Marker locates the minimum (lowest) data value.

**Next Peak** Marker locates the peak with the next lower amplitude value relative to its starting position.

**Peak Right** The marker locates the **next valid peak to the right** of its starting position on the X-axis.

**Peak Left** The marker locates the **next valid peak to the left** of its starting position on the X-axis.

- **Threshold** - Minimum amplitude (dB). To be considered valid, the peak must be **above** the threshold level. The valley on either side can be below the threshold level.
- **Excursion** The vertical distance (dB) between the peak and the valleys on both sides. To be considered a peak, data values must "fall off" from the peak on both sides by the excursion value.

For more information, see [What is a Peak?](#)

**Target** Enter the Target value. The marker moves to the first occurrence of the Target value to the right of its current position. Subsequent presses of the Execute button cause the marker to move to the next value to the right that meets the Target value. When the marker reaches the upper end of the stimulus range, it will "wrap around" and continue the search from the lower end of the stimulus range (left side of the window).

- If **Discrete Marker** is OFF, the marker locates the interpolated data point that equals the target value.
- If **Discrete Marker** is ON and there are two data points on either side of the target value, the marker locates the data point closest to the Target value

## Bandwidth Markers

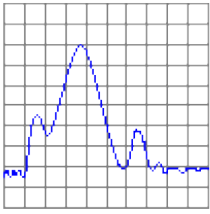
## Programming Commands

Four markers are automatically deployed to find the first negative or positive bandpass in the selected search domain.

To create Bandwidth markers:

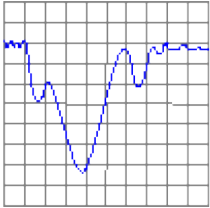
1. Press **SEARCH**, then **[Bandwidth]**
  2. Specify the level in dB from the peak or valley where bandwidth is measured.
- Bandwidth Search can be used ONLY with [Log Mag display format](#).
  - To use Bandwidth Search on a peak or valley other than the maximum or minimum values, change the Search Domain.

Enter a **Negative** number to search for a **Peak** bandpass, such as a filter S21 response:



- Marker 1: Maximum value within the [Search Domain](#).
- Marker 2: Specified level DOWN the left of the peak.
- Marker 3: Specified level DOWN the right of the peak.
- Marker 4: Center frequency between markers 2 and 3.

Enter a **Positive** number to search for a **Valley** bandpass, such as a filter S11 response:



- Marker 1: Minimum value within the [Search Domain](#).
- Marker 2: Specified level UP the left of the valley.
- Marker 3: Specified level UP the right of the valley.
- Marker 4: Center frequency between markers 2 and 3.

The following four values are displayed for Bandwidth Search:

- **BW:** (Marker 3 x-axis value) - (Marker 2 x-axis value) = width of the filter.
- **Center:** Mathematical midpoint between markers 2 and 3.
- **Q:** Ratio of Center Frequency to Bandwidth (Center Frequency / Bandwidth).
- **Loss:** Y-axis value of Marker 4. This is the loss of the filter at its center frequency. The ideal filter has no loss (0 dB) in the passband.

## Compression Markers

## Programming Commands

Uses the active marker to find the specified gain **Compression Level**. [Learn more about Gain Compression.](#)

**Note:** Valid ONLY for S21 (Gain) measurements with a [Power Sweep](#).

To create Compression markers:

1. Press **SEARCH**, then **[Compression]**
2. Specify the compression level in dB,
3. Optionally press **Tracking** to search for the specified compression level with each sweep.

Linear gain is defined as the Y-axis value (gain) of the first data point of the [Search Domain](#) (Full Span by default).



**Marker > N:** X-axis value and Y-axis value

**Comp Pin:** Input power (marker X-axis value)

**Comp Pout:** Output power (Pin + gain)

**Comp Level:** Compression level found.

- When [Discrete](#) is OFF (default setting) the marker finds the exact specified compression, interpolated between the two closest data points and calculates the Comp Pin and Comp Pout value for that point.
- When Discrete is ON (not interpolated), the marker resides on the closest data point to the requested compression level.

**Comp. Not Found:** Displayed when the requested compression level is not found.

## About PSAT and PNOP Markers

Compression measurements based on the Pout vs Pin curves are common in the satellite test industry. In the case of Travelling Wave Tube (TWT) amplifiers, PSAT markers identify the normal operating point near saturation, and the amplifiers are operated with the power slightly backed-off approximately 0.03 to 0.1 dB. For TWT amplifiers, the saturation curve always "folds over" and produces a maximum power out.

For Solid State Power Amplifiers (SSPA), the saturation is not as well defined. A common reference is the Normal Operating Point, which is a power backed-off by 8 to 10 dB from the maximum power. In this case, the normal operating point marker replaces the Psat with the PNOP values. Also, because the backoff is important, the backoff output and input powers are displayed (PBO Out), (PBO in) as well as gain at back off (PBO Gain).

### Power Saturation (PSAT) Markers

### Programming Commands

Uses Markers 1, 2, and 3 to quickly identify output power saturation parameters of an amplifier.

**Note:** Valid ONLY for Power IN vs Power OUT measurements. These markers can also be used on a CompOut trace with [Compression Analysis](#) mode in the Gain Compression Application.

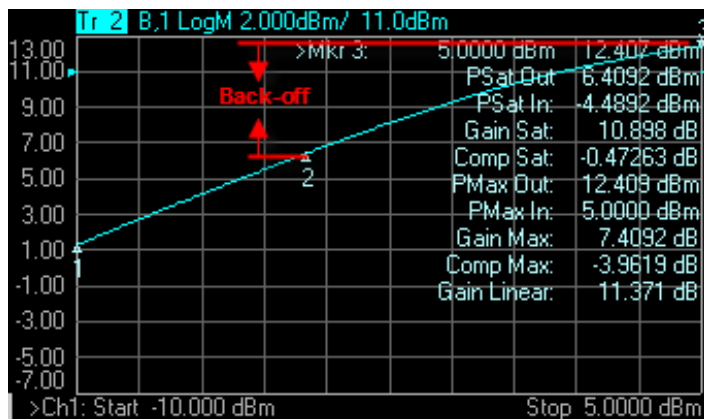
To make a Power IN (X-axis) vs Power OUT (Y-axis) measurement:

1. [Preset](#)
2. Set [Sweep Type](#): Power Sweep

3. Set [Trace Meas](#) to "B" Receiver
4. Connect DUT input to port 1
5. Connect DUT output to port 2

To create PSAT markers:

1. Press **SEARCH**, then **[Search...]**
2. From Search Type, select **Power Saturation**
3. For **PMax Back-Off**, enter the Y-axis (Power OUT) difference between the Max Power marker (3) and the Back-off marker (2).
4. Press **Execute** or check **Tracking**. [Learn more](#).



This setting uses **three** markers to calculate and display 10 values.

**The three markers:**

- Marker 1: Linear gain; the first data point in the sweep.
- Marker 2: Specified output power **Back-off** from max power.
- Marker 3: Max Power output; usually the last data point.

The 10 displayed values:

<b>&gt;Mkr 3:</b>	Marker 3 X-axis and Y-axis value
<b>PSat Out:</b>	Marker 2 Y-axis value
<b>Psat In:</b>	Marker 2 X-axis value
<b>Gain Sat:</b>	Psat Out - Psat In
<b>Comp Sat:</b>	Gain Sat - Gain Linear
<b>PMax Out:</b>	Marker 3 Y-axis value
<b>PMax In:</b>	Marker 3 X-axis value
<b>Gain Max:</b>	PMax Out - PMax In
<b>Comp Max:</b>	Gain Max - Gain Linear
<b>Gain Linear:</b>	Marker 1 - Y-axis value MINUS X-axis value

**Comp. Not Found:** Displayed when the requested Back-off point is not found.

When [Discrete](#) marker is NOT selected (the default setting), the three markers find an interpolated value between the two closest data points.

When [Discrete](#) marker is selected (NOT interpolated), the three markers reside on the closest data points.

### Power Normal Operating Point Marker

### Programming Commands

Uses Markers 1, 2, 3, and 4 to quickly identify Normal Operating Point parameters of an amplifier.

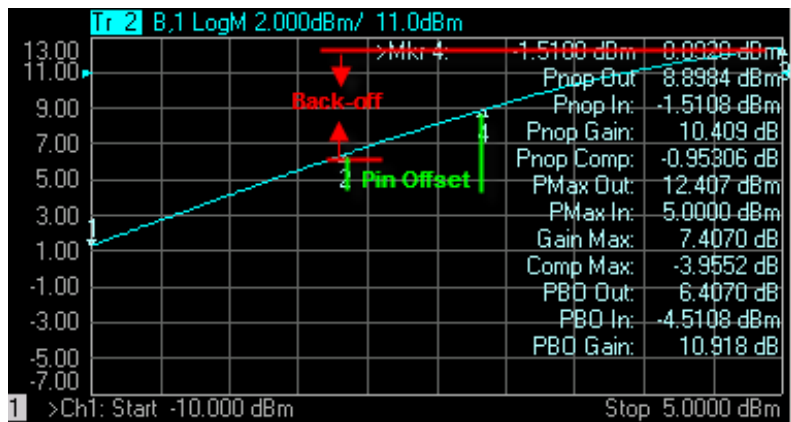
**Note:** Valid ONLY for Power IN vs Power OUT measurements.

These markers can also be used on a CompOut trace with [Compression Analysis](#) mode in the Gain Compression Application.

See [Power Saturation](#) to learn how to make a Power IN (X-axis) vs Power OUT (Y-axis) measurement.

To create PSAT markers:

1. Press **SEARCH**, then **[Search...]**
2. From Search Type, select **Normal Operating Pt**
3. For **Back-Off**, enter the Y-axis (Power OUT) difference between the Max Power marker (3) and the Back-off marker (2).
4. For **Pin Offset**, enter the X-axis (Power IN) difference between Back-off marker (2) and PNOP marker (4).
5. Press **Execute** or check **Tracking**. [Learn more](#).



This setting uses **four** markers to calculate and display 12 values.

#### The four markers:

- Marker 1: Linear gain; the first data point in the sweep.
- Marker 2: Max Output Power MINUS the specified Output (Y-axis ) **Back-off** value in dB.
- Marker 3: Max Output Power; usually the last data point in the sweep.
- Marker 4: X-axis value of Back-off (Marker 2) plus the **Pin Offset** (X-axis) value in dB.

The 12 displayed values:

<b>&gt;Mkr 4:</b>	Marker 4 - X-axis and Y-axis values
<b>Pnop Out:</b>	Marker 4 Y-axis value
<b>Pnop In:</b>	Marker 4 X-axis value
<b>Pnop Gain:</b>	Pnop Out - Pnop In
<b>Pnop Comp:</b>	Pnop Gain - Linear Gain*
<b>PMax Out:</b>	Marker 3 Y-axis value
<b>PMax In:</b>	Marker 3 X-axis value
<b>Gain Max:</b>	PMax Out - PMax In
<b>Comp Max:</b>	Gain Max - Linear Gain*
<b>PBO Out:</b>	Marker 2 Y-axis
<b>PBO In:</b>	Marker 2 X-axis
<b>PBO Gain:</b>	PBO Out - PBO In

\*Linear Gain (not shown): Marker 1 - Y-axis value MINUS X-axis value

**PNOP Not Found:** Displayed when the requested back-off level is not found.

When [Discrete](#) marker is NOT selected (the default setting), the four markers each find an interpolated value between the two closest data points.

When [Discrete](#) marker is selected (NOT interpolated), the four markers each reside on the closest data point.

## What Is a "Peak"?

You define what the analyzer considers a "peak" by selecting the following two peak criteria settings:

- **Threshold** - Minimum amplitude (dB). To be considered valid, the peak must be **above** the threshold level. The valley on either side can be below the threshold level.
- **Excursion** - The vertical distance (dB) between the peak and the valleys on both sides. To be considered a peak, data values must "fall off" from the peak on both sides by the excursion value.

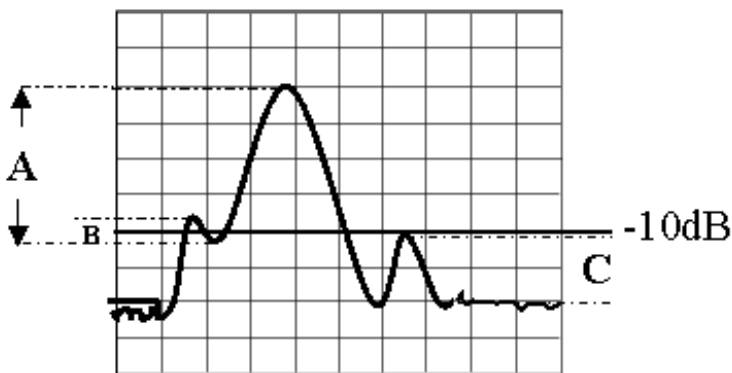
### Example:

Threshold Setting: -10dB

Excursion Setting: 1dB

Scale = 1 dB / Division

Mouse over the graphic to find a valid peak.



- **Peak A** = Valid Peak (Above Threshold and Excursion Settings)
- **Peak B** = Invalid Peak (Below Excursion Setting)
- **Peak C** = Invalid Peak (Below Threshold Setting)

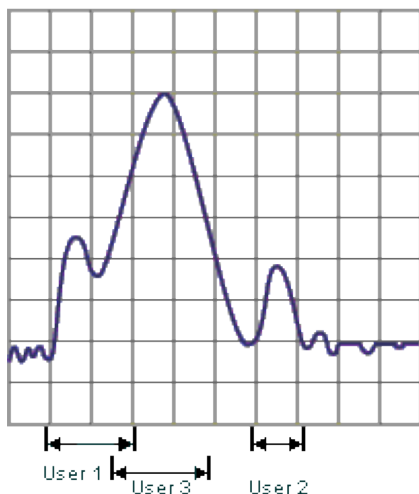
## Search Domain

Search domain settings restrict the stimulus values (X-axis for rectangular format) to a specified span. Set the Start and Stop stimulus settings of these **User** spans. If Start is greater than Stop, the marker will not move. [Learn how to set Search Domain.](#)

- The default domain of each new marker is "full span".
- There are 16 user-defined domains for every channel.
- The user-defined domains can overlap.
- More than one marker can use a defined domain.

The graphic below shows examples of search domains.





## Marker Functions - Change Instrument Settings

The following settings change the relevant PNA settings to the position of the active maker.

### How to change instrument setting using markers

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MARKER**
2. then **[Marker Function]**

#### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Marker Function**

### Programming Commands



## Marker Function dialog box help

**Note:** Marker Functions do not work with channels that are in [CW](#) or [Segment Sweep](#) mode.

**Marker =>Start** Sets the start sweep setting to the value of the active marker.

**Marker =>Stop** Sets the stop sweep setting to the value of the active marker.

**Marker =>Center** Sets the center of the sweep to the value of the active marker.

**Marker =>Ref Level** Sets the screen [reference level](#) to the value of the active marker.

**Marker =>Delay** The phase slope at the **active marker** stimulus position is used to adjust the line length to the receiver input. This effectively flattens the phase trace around the active marker. Additional Electrical Delay adjustments are required on devices without constant group delay over the measured frequency span. You can use this to measure the electrical length or deviation from linear phase.

This feature adds phase delay to a variation in phase versus frequency; therefore, it is only applicable for ratioed measurements. See [Measurement Parameters](#).

**Marker =>Span** Sets the sweep span to the span that is defined by the [delta marker](#) and the marker that it references. Unavailable if there is no delta marker.

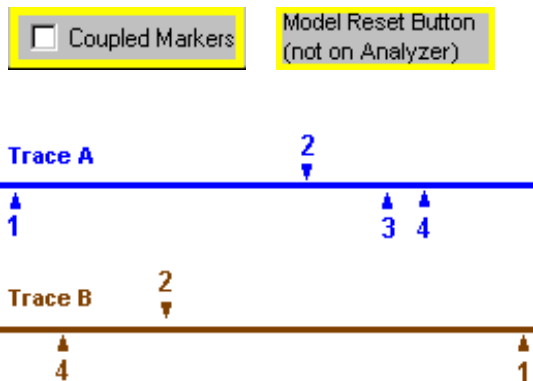
**Marker =>CW Freq** Sets the CW frequency to the frequency of the active marker. NOT available when the channel is in CW or Power Sweep. Use this function to first set the CW Frequency to a value that is known to be within the current calibrated range, THEN set [Sweep Type](#) to Power or CW.

## Coupled Markers

The coupled markers feature causes markers on different traces to line up with the markers on the selected trace. Markers are coupled by marker number, 1 to 1, 2 to 2, 3 to 3, and so forth. If the x-axis domain is the same (such as frequency or time), coupling occurs across all channels, windows, and traces. Trace markers in a different x-axis domain will not be coupled. If a trace marker has no marker to couple with on the selected trace, the marker remains independent.

### Coupled Markers Model

This model simulates the use of coupled markers in the PNA:



1. Click **Trace A** or **Trace B**
2. Click **Coupled Markers**

3. Notice the following:

- Markers on the unselected trace move to the x-axis position of the selected trace.
- If a marker number on the unselected trace has no corresponding marker on the selected trace, no movement occurs for that marker.

4. Click **Reset** to run the model again. There is no Reset for coupled markers on the PNA.

## Marker Table

You can display a table that provides a summary of marker data for the active trace. The marker data is displayed in the specified format for each marker.

### How to view the Marker Table

#### Using front-panel HARDKEY [softkey] buttons

1. Press **DISPLAY**
2. then **[More]**
3. then **[Tables]**
4. then **[Marker Table]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Display**
3. then **Tables**
4. then **Marker Table**

◀ **Programming Commands** ▶

Last Modified:

- |             |                                  |
|-------------|----------------------------------|
| 9-Feb-2010  | Added PSAT, PNOP, and new dialog |
| 6-Feb-2009  | Added compression marker         |
| 3-Oct-2008  | Added Marker->CW function        |
| 3-Sep-2008  | Removed legacy content           |
| 4-Jan-2008  | Added bookmark to move marker    |
| 17-Jul-2007 | Clarified bandwidth search       |
| 2-Feb-2007  | MX Added UI                      |

## Using Math / Memory Operations

---

You can perform four types of math on the active trace versus a memory trace. In addition three statistics (Mean, Standard Deviation and Peak to Peak) can be calculated and displayed for the active data trace.

- [Trace Math](#)
- [Trace Statistics](#)

**Note:** Trace Math (described here) allows you to quickly apply one of four math operations using memory traces. [Equation Editor](#) allows you to build custom equations using several types of traces from the same, or different channels.

### [Other Analyze Data topics](#)

## Trace Math

To perform any of the math operations, you must first store a trace to memory. You can display the memory trace using the [View](#) options.

Trace math is performed on the complex data before it is formatted for display. See the [PNA data processing map](#).

Markers can be used while viewing a memory trace.

### How to select Trace Math

#### Using front-panel HARDKEY [softkey] buttons

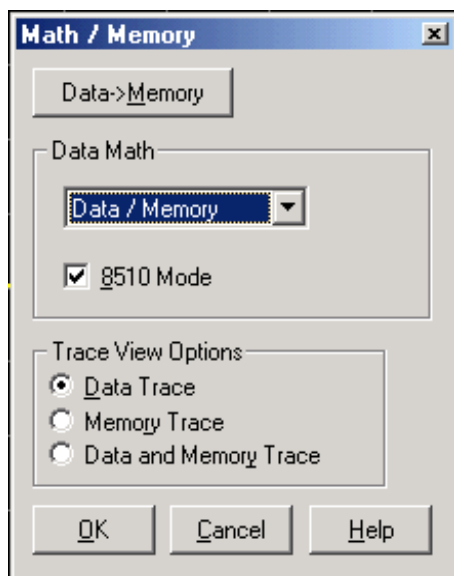
1. Press **MEMORY**

#### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Memory**
3. then **Memory**

**Normalize**, available only from the Memory menu, (not on the Math / Memory dialog), performs the same function as **Data=>Memory**, then **Data / Memory**.

◀ [Programming Commands](#) ▶



### Math / Memory dialog box help

**Normalize**, available only from the Memory menu, (not on the Math / Memory dialog), performs the same function as **Data=>Memory**, then **Data / Memory**.

**Data=>Memory** Puts the active data trace into memory. You can store one memory trace for every displayed trace.

**Note:** Many PNA features are NOT allowed on Memory traces. For example, Memory traces can NOT be stored to an SnP file. However, you can restore a memory trace to a data trace using the **Memory-to-Data** utility at the <http://na.tm.agilent.com/pna/apps/applications.htm> website.

### Data Math

All math operations are performed on linear (real and imaginary) data before being formatted. See the PNA Data flow (below).

**Data (or OFF)** Does no mathematical operation.

**Data / Memory** - Current measurement data is divided by the data in memory. Use for ratio comparison of two traces, such as measurements of gain or attenuation. [Learn more.](#)

**Data – Memory** - Data in memory is subtracted from the current measurement data. For example, you can use this feature for storing a measured vector error, then subtracting this error from the DUT measurement. [Learn more.](#)

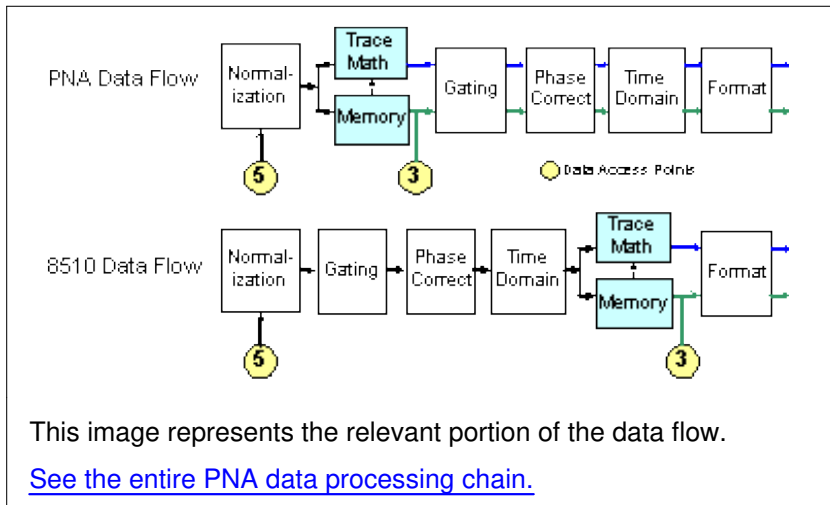
**Data + Memory** - Current measurement data is added to the data in memory. [Learn more.](#)

**Data \* Memory** - Current measurement data is multiplied by the data in memory. [Learn more.](#)

**8510 Mode** Check to simulate the Agilent 8510 data processing chain as it pertains to Trace Math and Memory. This setting applies to all channels. When the box is checked or cleared, the PNA performs an [Instrument Preset](#) and retains its setting through subsequent Instrument Presets.

This setting can be saved as part of an [instrument state](#). However, when recalled, this setting is assumed only temporarily. When a subsequent PNA Preset is performed, the PNA reverts to the setting that was in effect before the state was recalled.

You can [set a preference](#) to always use 8510 mode.



This image represents the relevant portion of the data flow.

[See the entire PNA data processing chain.](#)

A settings change in any of the operations that occur after the Memory operation on the above **PNA Data Flow** diagram changes both the Data trace and the Memory trace. For example, after storing a data trace to memory, when you change the format for the Data Trace, the format for the Memory Trace is also changed to the same setting.

### Trace View Options

**Data Trace** Displays ONLY the Data trace (**with selected math operation applied**).

**Memory Trace** Displays ONLY the trace that was put in memory.

**Data and Memory Trace** Displays BOTH the Data trace (with selected math operation applied). and the trace that was put in memory.

[Learn more about Trace Math](#) (scroll up)

### (Data / Memory) and (Data - Memory)

(Data / Memory) and (Data - Memory) math operations are performed on linear data before it is formatted. Because data is often viewed in log format, it is not always clear which of the two math operations should be used.

Remember: dividing linear data is the same as subtracting logarithmic data. The following illustrates, in general, when to use each operation.

Use **Data / Memory** for normalization purposes, such as when comparing S21 traces "before" and "after" a change is made or measurement of trace noise. In the following table, the Data/Mem values intuitively show the differences between traces. It is not obvious what Data-Mem is displaying.

S21 values to compare	Data/Mem	Data-Mem
0.5 dB and 0.6 dB	0.1 dB	-39 dB
0.5 dB and 0.7 dB	0.2 dB	-33 dB

Use **Data - Memory** to show the relative differences between two signals. Use for comparison of very small signals, such as the S11 match of two connectors.

In the following table, Data/Mem shows both pairs of connectors to have the same 2 dB difference. However, the second pair of connectors have much better S11 performance (-50 and -52) and the relative significance is shown in the Data-Mem values.

S11 values to compare	Data/Mem	Data-Mem
-10 dB and -12 dB	2 dB	-24 dB
-50 dB and -52 dB	2 dB	-64 dB

## Data \* Memory and Data + Memory

Use **Data \* Memory** and **Data + Memory** to perform math on an active data trace using data from your own formulas or algorithms rather than data from a measurement. For example, if you want to simulate the gain of a theoretical amplifier placed in series before the DUT, you could do the following:

1. Create an algorithm that would characterize the frequency response of the theoretical amplifier.
2. Enter complex data pairs that correspond to the number of data points for your data trace.
3. Load the data pairs into memory with SCPI or COM commands. The analyzer maps the complex pairs to correspond to the stimulus values at the actual measurement points.
4. Use the **data + memory** or **data \* memory** function to add or multiply the frequency response data to the measured data from the active data trace.

**Note:** The data trace must be configured before you attempt to load the memory.

## Trace Statistics

You can calculate and display statistics for the active data trace. These statistics are:

- Mean
- Standard deviation
- Peak-to-peak values

You can calculate statistics for the full stimulus span or for part of it with user ranges.

There are nine user ranges per channel. These user ranges are the same as the search domains specified for a marker search in that same channel; they use the same memory registers and thus share the same stimulus spans. If you specified search domains with marker search for a channel, you can recall these same spans by selecting the corresponding user ranges. The user ranges for a channel can overlap each other.

A convenient use for trace statistics is to find the peak-to-peak value of passband ripple without searching separately for the minimum and maximum values.

The trace statistics are calculated based on the format used to display the data.

- [Rectangular data formats](#) are calculated from the scalar data represented in the display

- [Polar](#) or [Smith Chart](#) formats are calculated from the data as it would be displayed in [Log Mag](#) format

[See how to make Trace Statistics display settings.](#)

### How to activate Trace Statistics

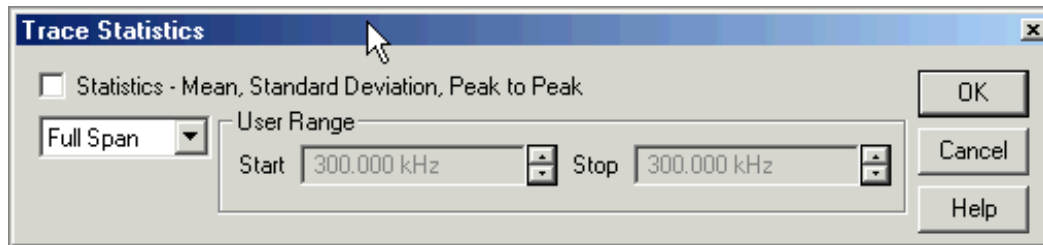
#### Using front-panel HARDKEY [softkey] buttons

1. Press **ANALYSIS**
2. then **[Statistics]**
3. then **[Trace Statistics]**

#### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Analysis**
3. then **Trace Statistics**

### Programming Commands



### Trace Statistics dialog box help

[See how to make Trace Statistics display settings.](#)

**Statistics** Check to display mean, standard deviation, and peak to peak values for the active trace.

**Span** Specifies the span of the active trace where data is collected for a math operation. You can define up to 9 user spans per channel with Start and Stop. You can also define the user spans from the [Marker Search dialog box](#).

**Start** Defines the start of a user span.

**Stop** Defines the stop of a user span.

[Learn more about Trace Statistics](#) (scroll up)

Last Modified:

- 3-Sep-2008 Removed legacy content
- 28-Aug-2008 Added Memory trace utility note
- 22-Apr-2008 Added 8510 preference link
- 27-Aug-2007 Edited trace display settings
- 2-Feb-2007 MX added UI





## Equation Editor

---

Equation Editor allows you to enter an algebraic equation that can mathematically manipulate measured data. The results are displayed as a data trace. Data that is used in the equation can be from the same or different channels.

**Note:** Equation Editor IS available with FCA measurements, Gain Compression, Noise Figure, or IMD.

- [Overview](#)
- [How to start Equation Editor](#)
- [Using Equation Editor](#)
- [Data that is used in Equation Editor](#)
- [Trace Settings, Error Correction, and an Example](#)
- [Functions and Constants](#)
- [Import Functions](#)
- [Operators](#)
- [Example Equations](#)
- [Saving Equation Editor Data](#)

### Other 'Analyze Data' topics

## Overview

Equation Editor allows you to enter an algebraic equation of standard mathematical operators and functions, referencing data that is available in the PNA. Once a valid equation is entered and enabled, the display of the active trace is replaced with the results of the equation, and updated in real-time as new data is acquired. For equations that can be expressed with Equation Editor's supported functions, operators, and data, there is no need for off-line processing in a separate program.

For example, enter the equation  $S_{21} / (1 - S_{11})$ . The resulting trace is computed as each  $S_{21}$  data point divided by one minus the corresponding  $S_{11}$  data point. For a 201 point sweep setup, the computation is repeated 201 times, once for each point.

As another example, suppose you want the PNA to make a directivity measurement of your 3-port DUT. This is not a native PNA measurement, but can be achieved using the Equation Editor. The desired result is the sum and difference of LogMag formatted traces, expressed as:  $S_{12} + S_{23} - S_{13}$ .

Because Equation Editor operates on **unformatted complex data**, the required equation is:

**DIR = S12 \* S23 / S13**

DIR becomes a display label to help you identify the computed data trace.

On the equation trace, set the format to LogMag.

## How to start Equation Editor

### Using front-panel HARDKEY [softkey] buttons

1. Press **ANALYSIS**
2. then **[Equation Editor]**

### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Analysis**
3. then **Equation Editor**

## Programming Commands



## Equation Editor dialog box help

### Notes

- **Double-click**, or type, the Functions, Operators, and Data to build an Equation.
- Equation Editor is NOT available with [FCA measurements](#).
- Scroll down to learn more about [Using Equation Editor](#)

**Equation:** The field in which equations are built. Click the down arrow to the right to use or modify equations that have been previously saved. This is where equations are saved when you press 'Store Equation'.

**Enabled** Check this box to enable the equation that is currently in the Equation field. If the Enabled box is not available, then the equation is not valid. If a data trace is used that is from a different channel than the Equation trace, the channels MUST have the same number of data points to be valid.

**<-Backspace** Moves the cursor to the left while erasing characters.

**<-** Moves the cursor to the left without erasing characters.

**->** Moves the cursor to the right without erasing characters.

**Store Equation** Press to save the current equation. To later recall the equation, click the down arrow to the right of the equation.

**Delete Equation** Removes the current equation from the drop-down list.

**Functions/Constants:** See [descriptions of Functions](#).

Select the "library" of functions to view. The "built-in" library appears by default which includes the standard functions of equation editor. Other functions that can appear here are functions that you have written and imported. [Learn more](#).

**Operators:** See [descriptions of Operators](#).

**Trace Data:** Select from ALL of the currently **displayed** traces on ALL channels.

**Ch Param Data:** Select from **undisplayed** data that is available ONLY from the active channel (same channel as the equation trace).

**Note:** With an external test set enabled, only parameters involving ports 1 through 4 are listed. However, all available parameters can be typed directly into the **Equation** field.

See [Data that is used in Equations](#).

**Show "Tr" annotation** Check to show the **TrX** annotation on PNA display and [Trace Status](#) buttons.

**Keypad:** Provided to allow navigation of the entire dialog with a mouse.

**Import Functions** Click to launch the [Import Functions](#) Dialog box.

## Using Equation Editor

### 1. Pick a trace in which to enter the equation

- Equation Editor works on the active trace.
- Either create a new trace, or click the [Trace Status](#) button on an existing trace to make the trace active.

### 2. Enter an equation

Start Equation Editor [See how](#).

**Note:** Equation Editor is NOT available for FCA measurements.

- The equation text can be in the form of an expression  $(S21)/(1-S11)$  or an equation  $DIR = S12 * S23 / S13$ . This topic refers to both types as equations.
- Either type, or double-click the Functions, Operators, and Data to build an equation.
- Functions and Constants ARE case-sensitive; Data names are NOT case sensitive.
- [Learn more about referring to data traces](#).

### 3. Check for a valid equation

When a valid equation is entered, the Enabled checkbox becomes available for checking. When the Enabled box is checked:

- The Equation Trace becomes computed data.
- The equation is visible on the [Trace Status](#) (up to about 10 characters).
- The equation is visible in the trace [Title](#) area (up to about 45 characters) when the Equation trace is active.
- The equation is visible in the [Status Bar](#) at the bottom of the display. This is updated only after the equation is entered and the [Trace Status](#) button is clicked.
- If an equation is NOT valid, and a trace from a different channel is used, make sure the number of data points is the same for both channels.

Learn more about the [Functions](#), [Operators](#), and [Data](#) that are used in Equation Editor.

## Data that is used in Equation Editor

### Definitions

- **Equation trace** A trace in which an equation resides.
- **Referred trace** A trace that is used as data in an equation.

**Example:**  $eq=Tr2+S11$  is entered into **Tr1**.

**Tr1** becomes an equation trace.

**Tr2** and **S11** are both referred traces because they are used in the equation trace.

### Notes

- Referred traces are processed one data point at a time. For example, the expression  $S11/S21$  means that for each data point in S11 and S21, divide point N of S11 by point N of S21.
- Once an equation is enabled, the trace is no longer identified by its original measurement parameter. It becomes an equation trace.
- An equation trace can NOT refer to itself. For example, an equation in Tr1 cannot refer to trace Tr1.
- Referred traces can be selected from S-Parameters, Receiver data, and [Memory traces](#).
- [See note regarding External Test Sets.](#)

### There are three ways to refer to traces:

The following distinction is important when discussing the three ways to refer to traces/data.

- **Trace** - a sequential collection of data points that are displayed on the PNA screen.
- **Data** - PNA measurements that are acquired but not displayed. When an equation trace refers to data that is not displayed, the PNA will automatically acquire the data.

### 1. Using **TrX** Trace notation (for example, Tr2).

When a trace is created, check ["Show Tr Annotation"](#) to see the **Tr** number of that trace.

- **Simple** - ALWAYS refers to displayed traces.
- Must be used for referring to traces in a different channel as the equation trace.
- All [trace settings](#) are preserved in the equation trace. If you do NOT want a trace setting to be used in the equation trace, you must disable it in the referred trace.
- If the referred trace is error corrected, then that data is corrected in the equation trace.
- Used to refer to a memory trace (it must already be stored in memory). Append .MEM to the **TrX** trace identifier. For example, **Tr2.mem** refers to the memory trace that is stored for Tr2.

### 2. Using **S-parameter** notation (for example, S11/S21)

- **Convenient** - ALWAYS refers to data that is NOT displayed.
- Refers to data that resides in the same channel as the equation.
- NOT the same as referring to a displayed S11 trace using **TrX** notation. [See Example.](#)
  - The referred data includes NO [trace settings](#).
  - If the channel has error correction available, then it can be applied by turning error correction ON for the Equation trace.

### 3. Using **Receiver** notation (for example AB\_2); NOT case sensitive.

At least one receiver is required, followed by an underscore and a number.

- The **letters** before the underscore refer to the receivers.
  - Letters alone refer to physical receivers.
  - Letters immediately followed by numbers refer to logical receivers. [Learn more.](#)
  - If two receivers are referenced, they are ratioed.
- The **number** after the underscore refers to the source port for the measurement.

Examples

- AR1\_2 = physical receiver A / physical receiver R1 with 2 as the source port.
- a3b4\_1 = reference receiver for port 3 / test port receiver for port 4 with 1 as the source port.

[Learn more about ratioed and unratioed receiver measurements.](#)

Receiver notation is like S-parameter notation in that:

- Refers to data that is NOT displayed and resides in the same channel as the equation.
- The referred data includes NO trace settings.
- If the channel has error correction available for that receiver, then it can be applied by turning error correction ON for the Equation trace.

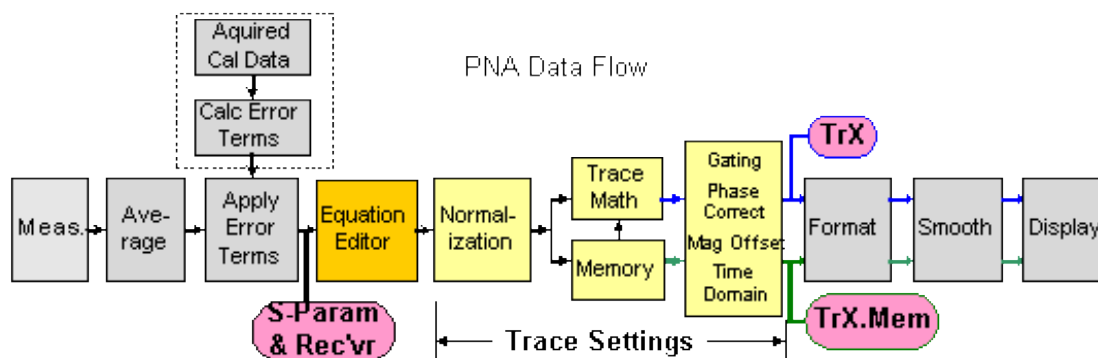
### Referring to Traces in a different channel

When the equation trace refers to a trace on a different channel:

- The trace must already be displayed.
- Must refer to the trace using **TrX** notation.
- The Equation trace and the referred trace **MUST** have the same number of data points or the Enable checkbox will not be available.
- The Equation trace is updated when the last referred data in the same channel is acquired. Therefore, to prevent 'stale' data from being used, the Equation trace must be on a higher numbered channel than the referred trace. This is because the PNA acquires data in ascending channel number order - first channel 1, then channel 2, and so forth. If the Equation trace is on channel 1, and it refers to a trace on channel 2, the Equation trace will update after channel 1 is finished sweeping, using 'old' data for the channel 2 trace.

### Trace Settings, Error Correction, and an Example

This discussion highlights the differences between using **S-parameter / Receiver** notation and **TrX** notation when referring to traces. The key to understanding the differences is realizing that **S-parameter / Receiver** notation ALWAYS refers to data that is NOT displayed.



- **Trace Settings** Normalization, Trace Math, Gating, Phase and Mag Offset, Electrical Delay, Time Domain.
- **Equation Editor** processing occurs on the **equation trace** immediately after error correction.
- **Referred Data/Trace** (used in the equation) is taken from the following locations:
  - When using **TrX** notation, data is taken immediately before formatting . These traces are always displayed

and include **Trace Settings**.

- When using **S-parameter** / **Receiver** notation, data is taken immediately after error correction. This data is NOT displayed and includes **NO** trace settings ([see example](#)).

See Equation Editor Notes at [GetData Method](#) or [GetDataByString Method](#).

### Error-correction and Equation Editor

Using **TrX** notation:

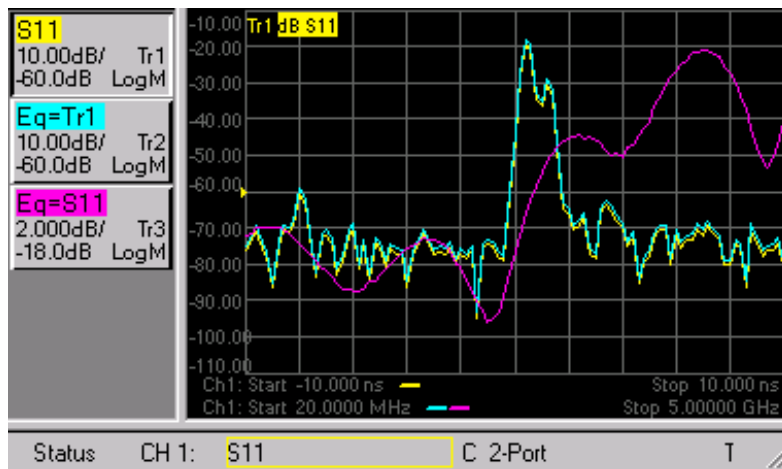
- The Trace Settings and Error-correction on the referred trace are used in the Equation trace.
- If error correction is NOT ON, then the raw, uncorrected data is used in the equation trace.
- To see if error correction is ON, make the trace active, then see the [Correction level in the status bar](#).
- Turning error correction ON/OFF on the equation trace has no meaning. The referred data that is used in the equation is ALWAYS what determines its level of correction.

Using **S-parameter** and **Receiver** notation:

- Because the data is not displayed, NO trace settings are used in the Equation trace.
- Correction can be turned ON/OFF if corrected data is available for the referred data. Exception: When using S-parameter and Receiver notation to refer to a trace on a channel that has been calibrated with a [Response Cal](#) or Receiver Cal, correction can NOT be turned ON, even though the Status Bar indicates otherwise. For example: Tr1 is an S11 measurement with a Response Cal. Tr2 is an equation trace that refers to S11. The Tr2 equation trace is NOT corrected, even though the Status Bar may indicate that it is corrected. However, if Tr2 refers to Tr1 (not S11), the Tr2 equation trace is corrected.

### Example

This example illustrates the differences when referring to a trace using **S-parameter** notation and **TrX** notation:



- **Tr1** is an S11 measurement with no equation, 2-port correction ON, and Time Domain transform ON.
- **Tr2** is an equation trace that refers to **Tr1**. Tr2 is corrected because Tr1 is corrected. Tr2 is transformed



because Tr1 is transformed. If transform is turned ON for Tr2, the data will be transformed AGAIN, which results in "unusual" data.

- **Tr3** is an equation trace that refers to **S11**. This is NOT the same as referring to Tr1. The S11 trace that is referred to is a different instance of S11 that is NOT displayed, and has NO trace settings. Notice that Tr3 data is NOT transformed, although Tr1 is transformed. Correction for **Tr3** can be turned ON and OFF because a calibration was performed on the channel in which the S11 trace resides.
- **Note:** X- axis annotation of the Equation trace is completely independent of the data that is presented. ONLY the **data values** from a referred trace are used. For example, notice that the Equation trace **Tr2** has Frequency on the X-axis although the referred trace **Tr1** is presented in Time.

## Functions and Constants used in Equation Editor

ALL trace data that is used in Equation Editor is unformatted, complex data.

In the following table,

- Function(scalar x) means that an automatic conversion from a complex number to its scalar magnitude is performed before passing the value to the function.
- Function(complex x) means that the entire complex value is used.
- **a, b, c, d** are arguments that are used in the function.

Function/Constant	Description
acos(scalar a)	returns the arc cosine of <b>a</b> in radians
asin(scalar a)	returns the arc sine of <b>a</b> in radians
atan(scalar a)	returns the arc tangent of <b>a</b> in radians
atan2	returns the phase of complex <b>a</b> = (re,im) in radians has the following two argument sets: <ul style="list-style-type: none"> <li>• atan2(complex a) - returns the phase in radians</li> <li>• atan2(scalar a, scalar b)</li> </ul>
conj(complex a)	takes <b>a</b> and returns the complex conjugate
cos(complex a)	takes <b>a</b> in radians and returns the cosine
cpx(scalar a, scalar b)	returns a complex value ( <b>a+ib</b> ) from two scalar values
e	returns the constant $\approx 2.71828...$

<code>exp(complex a)</code>	returns the exponential of <b>a</b>
<code>getNumPoints()</code>	returns the number of points for the current sweep
<code>im(complex a)</code>	returns the imag part of <b>a</b> as the scalar part of the result (zeroes the imag part)
<code>kfac(complex a, complex b, complex c, complex d )</code> when entered in EE: <code>kfac(S11,S21,S12,S22)</code>	k-factor: $k = (1 -  a ^2 -  d ^2 +  a*d-b*c ^2) / (2 *  b*c )$ returns a scalar result - the imaginary part of the complex result is always 0
<code>ln(complex a)</code>	returns the natural logarithm of <b>a</b>
<code>log10(complex a)</code>	returns the base 10 logarithm of <b>a</b>
<code>mag(complex a)</code>	returns $\sqrt{a.re*a.re+a.im*a.im}$
<code>max(complex a, complex b, ...)</code>	returns the complex value that has the largest magnitude of a list of values.
<code>max_hold(complex a)</code>	<b>New</b> - holds the current maximums of the sweep. Disable the equation to reset. <a href="#">See example</a>
<code>median(complex a, complex b,...)</code>	returns the median of a list of complex values <ul style="list-style-type: none"> <li>• The median is determined by sorting the values by magnitude, and returning the middle one.</li> <li>• If an even number of values is passed, then the smaller of the two middle values is returned.</li> </ul>
<code>min(complex a, complex b, ...)</code>	returns the complex value that has the smallest magnitude of a list of values.
<code>min_hold(complex a)</code>	<b>New</b> - holds the current minimums of the sweep. Disable the equation to reset. <a href="#">See example</a>
<code>mu1(complex a, complex b, complex c, complex d )</code> when entered in EE: <code>mu1(S11,S21,S12,S22)</code>	$\mu_1 = (1 -  a ^2) / ( d - \text{conj}(a) * (a*d-b*c)  +  b*c )$
<code>mu2( complex a, complex b, complex c, complex d )</code> when entered in EE: <code>mu1(S11,S21,S12,S22)</code>	$\mu_2 = (1 -  d ^2) / ( a - \text{conj}(d) * (a*d-b*c)  +  b*c )$

for both mu1 and mu2 (Usually written with the Greek character $\mu$ )	<ul style="list-style-type: none"> <li>conj is the complex conjugate. For scalars <b>a</b> and <b>b</b>, <math>\text{conj}(a+ib) = (a-ib)</math></li> <li>returns a scalar result - the imaginary part of the complex result is always 0</li> </ul>
phase(complex a)	returns atan2( <b>a</b> ) in degrees
PI	returns the numeric constant pi (3.141592), which is the ratio of the circumference of a circle to its diameter
pow(complex a,complex b)	returns <b>a</b> to the power <b>b</b>
re(complex a)	returns the scalar part of <b>a</b> (zeroes the imag part)
sin(complex a)	takes <b>a</b> in radians and returns the sine
sqrt(complex a)	returns the square root of <b>a</b> , with phase angle in the half-open interval $(-\pi/2, \pi/2]$
tan(complex a)	takes <b>a</b> in radians and returns the tangent
traceDataArray(complex a)	returns the entire set of points from a sweep. Function is intended to be used as an argument in an <a href="#">custom function</a> to allow access for data array processing.
xAxisArray()	<b>New</b> returns the x-axis values for the entire sweep.
xAxisIndex()	returns the current index in the sweep.
xAxisValue()	returns the current value of the x-axis index.

## Import Functions

You can create custom functions which are compiled into a DLL. Then import the functions for use in the Equation Editor.

See [Create custom functions for Equation Editor](#). This requires an internet connection.

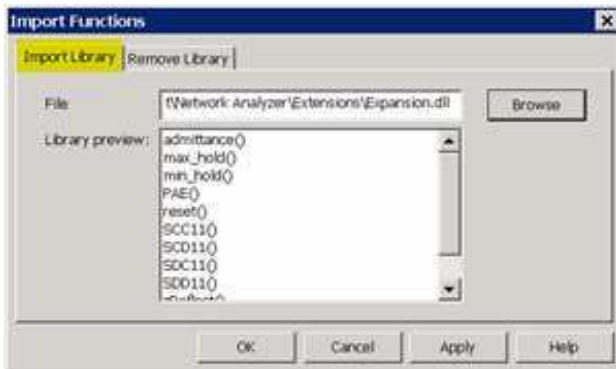
## Import Functions dialog box help

Imports and removes libraries that are used with Equation Editor. A library is a \*.DLL file that contains one or more custom functions.

Although not all functions are applicable to all channels or data sets, they will still appear in the "Function/Constants" list.

Each library is automatically loaded when the PNA applications starts. If a function is not found or if an error occurs while loading, the PNA will not attempt to reload the library when starting.

### Import Library tab



**Browse** click to navigate to the .DLL file on the PNA.

**Library Preview** Lists the functions that are contained in the library.

Click **OK** or **Apply** to load the library.

### Remove Library tab



**Left pane** Lists the imported libraries. These also appear in the Equation Editor main dialog and remain until removed from the PNA.

**Arrows** Click the relevant arrows to move some (>) or all (>>) libraries to the

**Right Pane** Lists the libraries to remove.

Click **OK** or **Apply** to remove the library.

## Operators used in Equation Editor

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
(	Open parenthesis
)	Close parenthesis
,	Comma - separator for arguments (as in S11, S22)
=	Equal (optional)
E	Exponent (as in 23.45E6)

## Example Equations

The following examples may help you get started with Equation Editor.

### Offset each data point in Tr2 from Tr1 by 2dB

Use the function: `pow(complex a, complex b)` -- returns **a** to the power **b**.

$$20\log(a) + 2 = 20\log(x)$$

$$\log(a) + 2/20 = \log(x) \text{ // divide all by 20.}$$

$$x = 10^{(\log(a) + 2/20)} \text{ // swap sides and take 10 to the power of both sides}$$

$$x = 10^{\log(a)} * 10^{(2/20)}$$

$$x = a * 10^{(2/20)}$$

The equation is entered into Tr2 as:

$$\text{Offset} = \text{Tr1} * \text{pow}(10, 2/20)$$

To offset by 5 dB

$$\text{Offset} = \text{Tr1} * \text{pow}(10, 5/20) .$$

### Balanced Match using a 2-port PNA

$$\text{SDD11} = (\text{S11} - \text{S21} - \text{S12} + \text{S22}) / 2$$

### Conversion loss

$$\text{B}_1 / \text{pow}(10, -15/20)$$

- B<sub>1</sub> is a receiver measurement;

- -15 is the input power in dBm

### Third-order intercept point (IP3 or TOI)

$$\text{Tr1} * \sqrt{\text{Tr1} / \text{Tr3}}$$

- Tr1 = input signal power
- Tr3 = intermodulation power (both traces measured with single receivers)

### Harmonics in dBc

$$\text{B}_1 / \text{Tr2}$$

- B\_1 is tuned to a harmonic frequency
- Tr2 = power at fundamental frequency, measured with B\_1 receiver

### PAE (Power Added Efficiency)

Pout - Pin / Pdc

Type the following equation into a new trace with an unratioed measurement, such as AI1. The data format is REAL:

$$\text{PAE} = 100 * (.001 * \text{pow}(\text{mag}(\text{Tr1}), 2) - (.001 * \text{pow}(\text{mag}(\text{Tr1}), 2) / \text{pow}(\text{mag}(\text{Tr2}), 2))) / (\text{Tr3} * \text{Tr4})$$

Where:

- Tr1 - a trace that measures unratioed B receiver.
- Tr2 - a corrected S21 trace (amplifier gain)
- Tr3 - a trace that measures [ADC voltage](#) (AI1) across a sensing resistor.
- Tr4 = an equation trace containing Isupp = (Tr3 / value of sensing resistor).

Data is displayed in Real format with units actually being watts.

### 1-port Insertion Loss

When it is not possible to connect both ends of a cable to the PNA, a 1-port insertion loss measurement can be made. However, the measured loss must be divided by 2 because the result includes the loss going down **and** coming back through the cable. This assumes that the device is terminated with a short to reflect all of the power. The 'divide by 2' operation is performed as follows using Equation Editor:

- Tr1 - an S11 trace
- Tr2 - an equation trace containing  $20 * \log_{10}(\text{Tr1}) / 2$

### Max and Min Hold

These two functions allow you to capture and display either the Maximum or Minimum values for each data point

over multiple sweeps.

**Maxhold (S21)** - displays the maximum value for each data point until reset. Reset by disabling, then enabling the equation. This example refers to an S21 trace that is not displayed.

## Saving Equation Editor Data

Equation data can be saved to the PNA hard drive in the following formats:

- [Citifile \(.cti\)](#) - Equation data is saved and recalled. The file header indicates the "underlying" s-parameter trace type.
- [Trace \(.prn\)](#) - read by Spreadsheet software. Can NOT be recalled by the PNA.
- [Print to File](#) (bmp, jpg, png) - saves image of PNA screen.

Equation data is NOT saved in [.SnP file format](#). When attempting to save an Equation trace in .SnP format, the "underlying" S-parameter data is saved; not Equation data.

---

### Last Modified:

11-Jun-2009	Added link to GetData
3-Sep-2008	Removed legacy content
22-Aug-2008	Added xAxisArray() Maxhold() and Minhold() function
6-May-2008	Added Import functions
3-Jul-2007	Added PAE and other notes
18-Jun-2007	Added examples

## Using Limit Lines

---

Limit lines allow you to compare measurement data to performance constraints that you define.

- [Overview](#)
- [Create and Edit Limit Lines](#)
- [Display and Test with Limit Lines](#)
- [Testing with Sufficient Data Points](#)

### Other Analyze Data topics

## Overview

Limit lines are visual representations on the PNA screen of the specified limits for a measurement. You can use limit lines to do the following:

- Give the operator **visual guides** when tuning devices.
- Provide **standard criteria** for meeting device specification.
- Show the **comparison** of data versus specifications.

Limit testing compares the measured data with defined limits, and provides optional **Pass or Fail** information for each measured data point.

You can have up to **100** discrete lines for each measurement trace allowing you to test all aspects of your DUT response.

Limit lines and limit testing are NOT available with **Smith Chart** or **Polar** display format. If limit lines are ON and you change to Smith Chart or Polar format, the analyzer will automatically disable the limit lines and limit testing.

## Create and Edit Limit Lines

You can create limit lines for all measurement traces. The limit lines are the same color as the measurement trace.

Limit lines are made up of discrete lines with four coordinates:

- BEGIN and END stimulus - X-axis values.
- BEGIN and END response - Y-axis values.



## How to create, edit, and test with Limit Lines

All limit line settings are made with the limit table. Use one of the following methods to show the limit table:

### Using front-panel HARDKEY [softkey] buttons

1. Press **ANALYSIS**
2. then **[Limits]**
3. then **[Limit Test]**

### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Analysis**
3. then **Limit Test**

## Programming Commands

### Limit Table

	TYPE	BEGIN STIMULUS	END STIMULUS	BEGIN RESPONSE	END RESPONSE
1	MIN	1.930000 GHz	1.990000 GHz	-5.000000 dB	-5.000000 dB
2	MAX	1.000000 GHz	1.500000 GHz	-60.000000 dB	-50.000000 dB
3	MAX	2.050000 GHz	3.000000 GHz	-50.000000 dB	-60.000000 dB
4	OFF	0.000000 Hz	0.000000 Hz	0.000000 dB	0.000000 dB

**Note:** To ADD a limit line to the table, change the last limit line to either MAX or MIN

1. In the **Type** area of the Limit Table, select **MIN** or **MAX** for Limit Line 1.
  - The MIN value will fail measurements BELOW this limit.
  - The MAX value will fail measurements ABOVE this limit.
2. Click **BEGIN STIMULUS** for Limit Segment 1. Enter the desired value.
3. Click **END STIMULUS** for Limit Segment 1. Enter the desired value.
4. Click **BEGIN RESPONSE** for Limit Segment 1. Enter the desired value.
5. Click **END RESPONSE** for Limit Segment 1. Enter the desired value.
6. Repeat Steps 1-5 for each desired limit line.

## Displaying and Testing with Limit Lines

After creating limit lines, you can then choose to **display** or **hide** them for each trace. The specified limits remain valid even if limit lines are not displayed.

Limit testing cannot be performed on memory traces.

You can choose to provide a visual and / or audible PASS / FAIL indication.

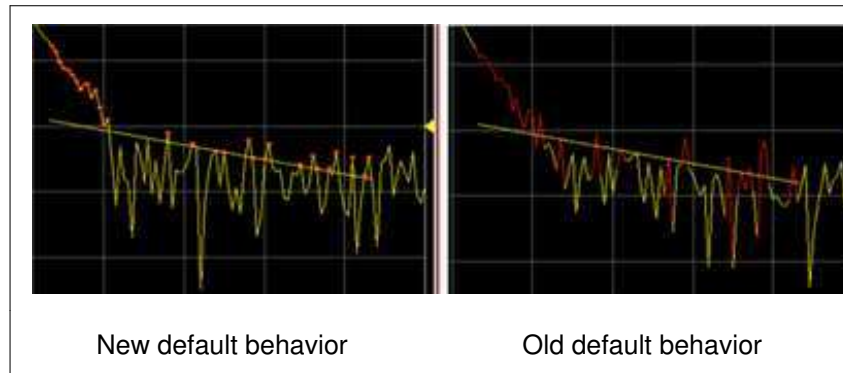
With limit testing turned ON:

- Any portion of the measurement trace that **fails** is **displayed in red**.

- Any portion of the measurement trace that does **NOT fail** remains unchanged and silent.

### Display failed trace points or trace segments

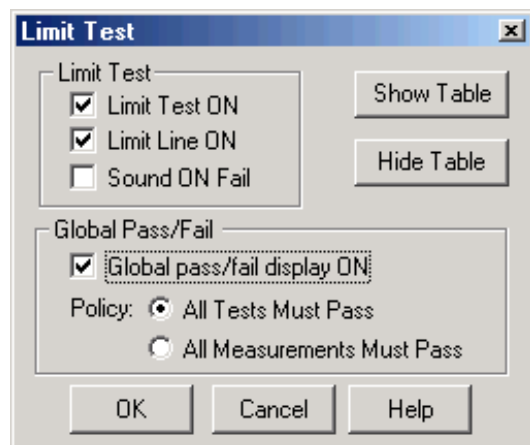
Beginning with PNA Rev. A.09.00, the new default behavior is to display the data points that fail limit line testing as red dots. This can be changed to the old default behavior (the failed limit line segment is colored red) with a Preference setting. [Learn how.](#)



### PASS is the default mode of Pass / Fail testing.

A data point will FAIL only if a measured point falls outside of the limits.

- If the limit line is set to OFF, the entire trace will PASS.
- If there is no measured data point at a limit line stimulus setting, that point will PASS.



## Limit Test dialog box help

**Show Table** Shows the table that allows you to create and edit limits.

**Hide Table** Makes the limits table disappear from the screen.

**Note:** To ADD a limit line to the table, change the last limit line to either MAX or MIN

### Limit Test

**Limit Test ON** Check the box to compare the data trace to the limits and display PASS or FAIL.

**Limit Line ON** Check the box to make the limits visible on the screen. (Testing still occurs if the limits are not visible.)

**Sound ON Fail** Check the box to make the PNA beep when a point on the data trace fails the limit test.

### Global Pass/Fail

The Pass/Fail indicator provides an easy way to monitor the status of ALL measurements.

**Global pass/fail display ON** Check to display the Global Pass/Fail status.

**Policy:** Choose which of the following must occur for the Global Pass/Fail status to display PASS:

- All Tests (with **Limit Test ON**) Must Pass - This setting reads the results from the Limit Tests. If all tests (with **Limit Test ON**) PASS, then the Global Pass/Fail status will PASS.
- All Measurements Must Pass - This more critical setting shows FAIL unless all measured data points fall within established test limits **and** Limit Test is ON. **Note:** In this mode, if one measurement does NOT have **Limit Test ON**, Global Pass/Fail will show FAIL.

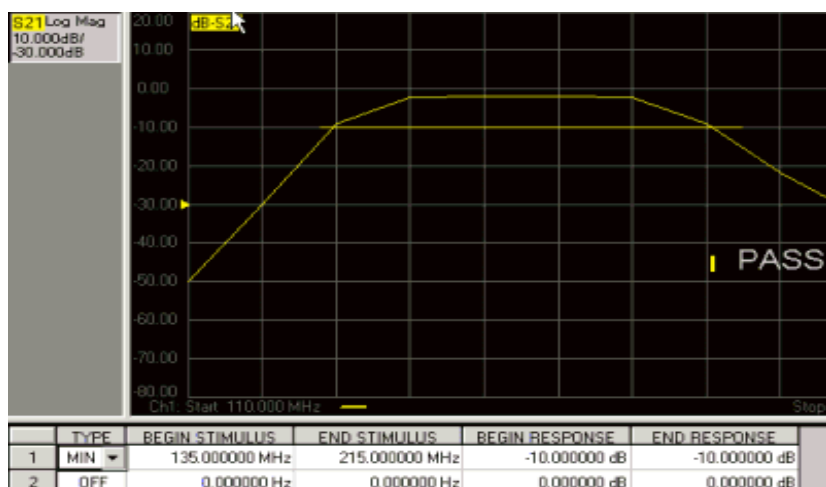
[Learn more about displaying and testing with Limits \(scroll up\)](#)

## Testing with Sufficient Data Points

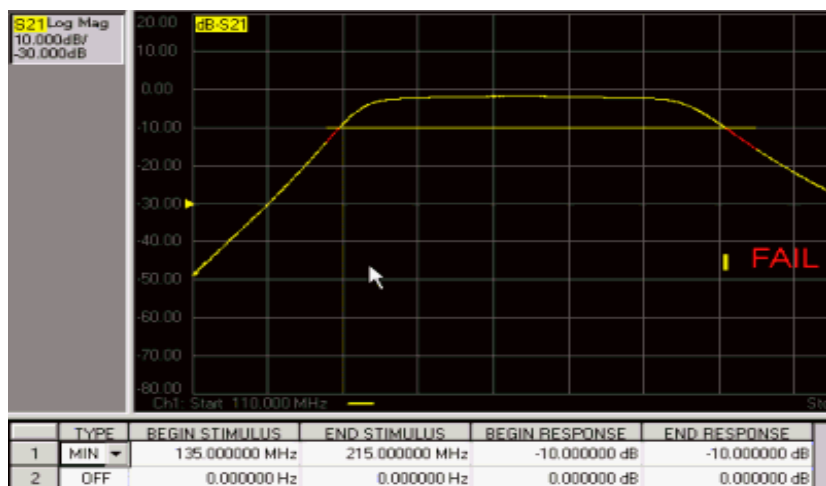
**Limits are checked only at the actual measured data points.** Therefore, It is possible for a device to be out of specification without a limit test failure indication if the data point density is insufficient.

The following image is a data trace of an actual filter using 11 data points (approximately one every vertical graticule). The filter is being tested with a minimum limit line (any data point under the limit line fails).

Although the data trace is clearly below the limit line on both sides of the filter skirts, there is a PASS indication because there is no data point being measured at these frequencies.



The following image shows the exact same conditions, except the number of data points is increased to 1601. The filter now fails the minimum limit test indicated by the red data trace.



Last Modified:

- 11-Aug-2009 Added failed dots
- 3-Sep-2008 Removed legacy content
- 2-Feb-2007 MX Added UI

## Save and Recall a File

---

The PNA allows you to save and recall files to and from an internal or external storage device in a variety of file formats.

- [How to Save PNA Instrument State](#)
- [How to Save PNA Measurement Data](#)
- [How to Recall a File](#)
- [About Instrument State and Calibration\) Files](#) ( .csa, .cst, .sta, .cal)
- [About Measurement Data Files](#) (.prn, .snp, .cti, .csv, .mdf)
- [Managing Files without a Mouse](#)

### Other Data Outputting topics

#### How to Save Instrument State and Calibration Files

Use one of the following methods:

##### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[Save]**, **[Save As]**, or **[Auto Save]**

##### Using a mouse with PNA Menus

1. Click **File**
2. then **Save** or **Save As**

[Learn all about PNA Instrument State files.](#)

**Save** Immediately saves the PNA state and possibly calibration data to the filename and extension you used when you last performed a Save. This file will be overwritten the next time you click **Save**. To prevent this, use one of the following methods.

**Save As** Starts the [Save As](#) dialog box.

**Auto Save** NOT available from the PNA menu. Saves state and calibration data to the internal hard disk in the C:/Program Files/Agilent/Network Analyzer/Documents folder. A filename is generated automatically using the syntax "**atxxx.csa**"; where xxx is a number that is incremented by one when a new file is Auto Saved.

[Programming Commands](#)



### Save (State and Calibration ) As dialog box help

**Save** Allows you to navigate to the directory where you want to save the file.

**File name** Displays the filename that you either typed in or clicked on in the directory contents box.

#### Save as type

The following file types save **Instrument states and Calibration data**. You can save, and later recall, instrument settings and calibration data for **all channels** currently in use on the PNA. These file types are only recognized by Agilent PNA Series analyzers.

[Learn more about these file types.](#)

- **\*.csa** - save Instrument state and actual Cal Set data (cal/state archive) **Default selection.**
- **\*.cst** - save Instrument state and a link to the Cal Set data.
- **\*.sta** - save Instrument state ONLY (**no** calibration data)
- **\*.cal** - save actual Calibration data ONLY (**no** Instrument state)

**Note:** To save the PNA screen as .bmp, .jpg, or .png graphics file types, click **File / Print to File.** [Learn more.](#)

**Save** Saves the file to the specified file name and directory.

### How to Save Measurement Data

Use one of the following methods:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[Save Data As]**

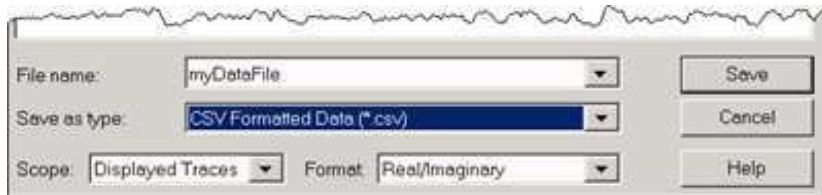
#### Using a mouse with PNA Menus

1. Click **File**
2. then **Save Data As**

**Save Data As** Saves the current trace(s) to the specified type of file.

**Note:** Beginning with A.09.20, FCA .S2P files CAN be saved using this dialog.

[Programming Commands](#)



### Save Data As dialog box help

**Note:** Before saving measurement data, always [trigger a single](#) measurement, and then allow the PNA channel to go into Hold. This ensures that the entire measurement trace is saved.

**Save** Allows you to navigate to the directory where you want to save the file.

**File name** Displays the filename that you either typed in or clicked on in the directory contents box.

#### Save as type

The following file types save **Measurement data** for use in spreadsheet or CAE programs: [\\*.prn](#), [\\*.SNP](#), [\\*.cti \(citifile\)](#), [\\*.csv](#).

- GCA, Swept IMD, and Swept IMDx data can be saved to a special csv format. Learn how ([GCA - Swept IMD](#))
- Citifiles and SNP files can be recalled and viewed in the PNA. [Learn more.](#)
- To save the PNA screen as .bmp, .jpg, or .png graphics file types, click **File / Print to File.** [Learn more.](#)

### Data Scope (CTI Data and Formatted, CSV, and MDIF)

Determines what traces are saved to a file.

- **Auto**
  - When correction is OFF, saves the specified trace.
  - When correction is ON, saves all corrected parameters associated with the calibrated ports in the Cal Set.
  - For GCA and Swept IMD channels, saves the active trace only.
- **Single Trace** - Saves the active trace.
- **Displayed Traces** - Saves all displayed traces for all channels.

### Format (CTI, CSV, SNP, MDIF)

- **Auto** - Data is saved in LogMag or LinMag if one of these is the currently selected display format. If format is other than these, then data is saved in Real/Imag.
- **Displayed Format (CSV and MDIF only)** - Data is saved in the format of the displayed trace.
- **LogMag, LinMag, Real/Imag** - Select output format.

- The imaginary portion for all **LogMag** and **LinMag** data is saved in degrees.
- Real/ Imag data is never smoothed.

**Note:** .prn files can only save the active trace in the displayed format.

**Save** Saves the file to the specified file name and directory.

### How to Recall (open) a file

Select a file from the 'most recently used' list. The list is saved when the PNA application exits.

Use one of the following methods:

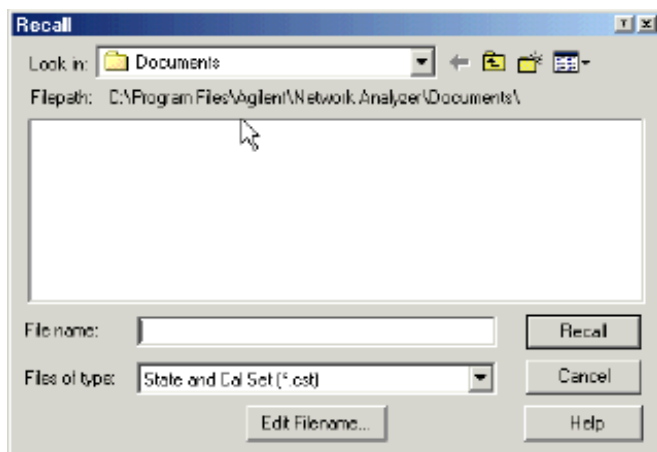
#### Using front-panel HARDKEY [softkey] buttons

1. Press **RECALL**
2. then **[Recall]**

#### Using a mouse with PNA Menus

1. Click **File**
2. then **Recall**

### Programming Commands





## Recall dialog box help

**Look in** Allows you to select the directory that contains the file that you want to recall.

**Filename** Displays the filename that you either typed in or clicked on in the directory contents box.

**Files of type** Allows you view and select files that are listed in categories of a file type. The following types of files can be recalled into the PNA: All [State files](#), Citi files, SNP files.

### Recalling PNA State files

When an Instrument State file is recalled into the PNA, the current state of the instrument is overwritten with the recalled state. A \*.cal file does not contain an instrument state, but only calibration data. [Learn more about Instrument States.](#)

See also [Power ON and OFF during Save / Recall, User Preset, and Preset.](#)

### Recalling Data files into the PNA

Citi files and SNP files can be recalled and viewed in the PNA.

1. Click **File** then **Recall**.
2. Select **Citifile Data** or **Snp**.
3. Select the file to recall
4. Click **Recall**.

**Note:** Citi files that were saved in [CW Time sweep](#) can NOT be recalled into the PNA

Recalled data is ALWAYS displayed on the PNA using [LogMag format](#), regardless of how the file was stored.

The channel is placed in Trigger Hold. If triggering is resumed, the data will be overwritten.

**SNP files** are recalled as traces into a single window and channel, beginning at the [highest available channel number allowed on the PNA](#). For multi-port SNP files (greater than 4 ports), if the number of S parameters in the file is beyond the [maximum number of traces in a window](#), then new windows will be created.

**Citi files** are recalled into the same window and channel configuration as when they were saved. However, the new recalled channel numbers begin with the [highest channel number allowed on the PNA](#) and decrement for each additional channel.

For example, when a citi file is saved, two traces are in window 1, channel 1 and two additional traces are in window 2, channel 2. When recalled into a factory preset condition (1 trace in window 1, channel 1), the first two recalled traces appear in window 2, highest channel number, and the second two traces appear in window 3, (highest channel number -1).

See also [Traces, Channels, and Windows on the PNA](#)

**Recall** Recalls the file displayed in the file name box.

## Instrument State / Calibration Files

You can save, and later recall, instrument settings and calibration data **for all channels** currently in use on the

PNA.

An **Instrument State** contains almost every PNA setting. The following PNA settings are NOT saved and recalled with Instrument State:

- [GPIB address](#)
- [RF power ON/OFF](#) (depends on current setting)
- [Test set I/O settings](#)

The following file types are used to save and recall instrument states and Cal Set information:

File Types				Information that is stored for each channel	
.csa	.cst		.sta	<b>Instrument State Information</b>	
				Channels/Traces	Averaging
				Windows	Markers
				Triggering	Math/memory
				Format	Limits
				Scale	More...
				Stimulus Information:	
				Frequency range	Alternate sweep
				Number of points	Port powers
				IF bandwidth	Source attenuators
		.cal		Sweep type	Receiver attenuators
				Sweep mode	Test Set port map
				<b>Cal Set Information</b>	
				GUID (Globally Unique Identifier) provides link to Cal Set	
				Name, Description, Modify date	
				Stimulus Information:	
				Frequency range	Alternate sweep
				Number of points	Port powers
				IF bandwidth	Source attenuators
				Sweep type	Receiver attenuators
				Sweep mode	Test Set port map
				Error Terms: Directivity, Crosstalk, Source match, Load match, Reflection tracking, Transmission tracking	

### File Type Descriptions and Recall

The following describes each file type, and what occurs when the file type is recalled.

#### \*.sta files

- Contain ONLY instrument state information.
- When recalled, they always replace the current instrument state immediately.

### \*.cst files

- Contain BOTH instrument state and a LINK to the Cal Sets.
- The **quickest and most flexible** method of saving and recalling a calibrated instrument state.
- Channels need not have cal data to save as .cst file.
- When recalled, the state information is loaded first. Then the PNA tries to [apply a Cal Set](#) as you would do manually. If the stimulus settings are different between the instrument state and the linked Cal Set, the usual choice is presented ([see Cal Sets](#)). If the linked Cal Set has been deleted, a message is displayed, but the state information remains in place.
- Because only a link to the Cal Set is saved, the Cal Set can be shared with other measurements.

**Note:** Before saving a .cst file, be sure that a User Cal Set (NOT a Cal Register) is being used for the calibration. Cal Registers are overwritten with new data whenever a calibration is performed, and may not be accurate cal data when the .cst file is recalled. [Learn more about Cal Sets.](#)

### \*.cal files

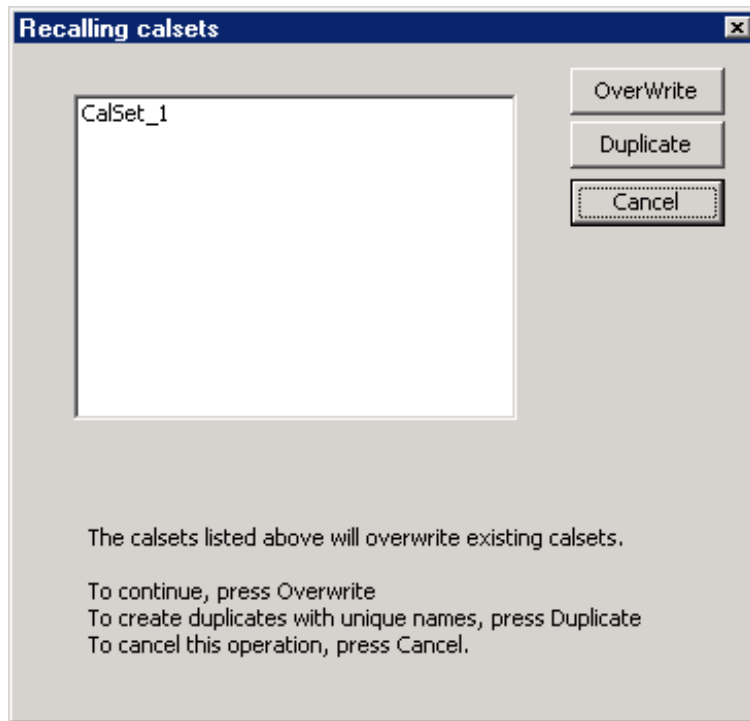
- Contain ONLY Cal Set information.
- When recalled, the Cal Set is NOT automatically applied. Apply the calibration data to a channel as you would [apply any Cal Set](#).
- [Learn about Recalling](#)

### \*.csa files

- Contain ALL instrument state and the actual Cal Set; not a link to the Cal Set.
- The **safest** method of saving and recalling a calibrated instrument state. However, the file size is larger than a \*.cst file, and the save and recall times are longer. In addition, because the actual Cal Set is saved, it is very difficult to share the cal data with other measurements.
- Channels need not be calibrated to save as .cst file.
- The Cal Set that is saved could have been a [Cal Register or a User Cal Set](#).
- [Learn about Recalling](#)

**Note:** \*.pcs files are the internal file format the PNA uses for storing cal sets. There is no reason for users to access or copy these files.

## Recalling Cal Sets



### Recalling Cal Sets dialog box help

Both **.cal** and **.csa** file types contain whole Cal Sets. When these file types are recalled, the PNA checks to see if the incoming Cal Set GUID matches an existing PNA Cal Set GUID. If it does, and if the rest of the Cal Set contents are different in any way, then both of these Cal Sets can NOT coexist in the PNA and you are offered the following choices.

Because all PNA channels are saved, there could be more than one Cal Set in either of these file types.

**Overwrite** The incoming Cal Set will replace the existing Cal Set.

**Duplicate** (Only available with .cal recalls.) Because the Cal Set is not automatically applied, you can choose to apply either the original or duplicate Cal Set. The original Cal Set remains in the .cal file.

**Cancel** Abandon the recall operation.

The PNA will offer a choice as described in each file type below. [Learn more about Cal Sets.](#)

## Measurement Data Files

Measurement data is saved as ASCII file types for use in a spreadsheet or CAE programs.

**Note:** Before saving measurement data, always [trigger a single](#) measurement, and then allow the PNA channel to go into Hold. This ensures that the entire measurement trace is saved.

The following file types can be saved by the PNA.

- [\\*.prn files](#)

- [\\*.SNP \(Touchstone\)](#)
- [\\*.cti \(Citifile\)](#)
- [\\*.csv](#)
- [\\*.mdf \(MDIF\)](#)

## \*.prn Files

Prn files have the following attributes:

- Comma-separated data which can be read into rows and columns by spreadsheet software, such as Microsoft® Excel. To avoid the "delimiting" dialog boxes, change the filename extension from .prn to .csv. Then open directly into Microsoft Excel.
- Contain formatted and corrected stimulus and response data for the current active trace ONLY.
- Are Output only - they cannot be read by the PNA.
- Beginning with PNA Rev 6.2, [FCA](#) and [Cal Set Viewer](#) data can be saved to \*.prn files

Example:

### "S<sub>11</sub> Log Mag"

"Frequency (Hz)",	"dB"
3.000000e+005 ,	-3.528682e+001 ,
4.529850e+007 ,	-2.817913e+001 ,
9.029700e+007 ,	-3.216808e+001 ,
1.352955e+008 ,	-3.101017e+001 ,

## .SNP Format (\*.s1p, \*.s2p, \*.s3p, \*.s4p, and so forth)

This file format is used by CAE programs such as Agilent's Microwave Design System (MDS) and Advanced Design System (ADS).

**Note:** Beginning with PNA Rev. A.09.20...

- \*.S2P data can be saved using the **File**, [Save Data As](#) dialog.
- FCA data can be saved to an S2P file using the **File**, [Save Data As](#) dialog.
- Balanced parameters can be saved to SNP files. See the ["Choose Ports " dialog](#).

- .SNP data can be [recalled](#) and viewed on the PNA, or read by the PNA [embed/de-embed](#) functions.
- The amount of data that is saved depends on the file type that you specify and the amount of data that is available:
- To save SNP data with an [external test set](#) enabled, at the File, [Save As](#) dialog, select **SNP File(\*.s\*p)**, then complete the ["Choose Ports " dialog](#).
- When [Fixturing](#) is enabled, all of the enabled data transforms (De-embedding, Port Z Conversion, and so forth) are applied to saved SNP files.

File Type	# of Ports	# of S-parameters saved
*.s1p	1	1 S-parameter
*.s2p	2	4 S-parameters
*.s3p	3	9 S-parameters
*.s4p	4	16 S-parameters
...	...	...
*.SNP	N	$N^2$ S-parameters

.SNP data is generally used to gather all S-parameters for a fully corrected measurement. The PNA uses the data that is available on the channel of the active measurement.

- If correction is applied, then valid data is returned for all corrected s-parameters.
- If requesting **less** data than is available, the [Choose ports for SNP data](#) dialog appears. Previous to PNA release 6.2, data was returned beginning with the first calibrated ports until your request is fulfilled.
- If correction is NOT applied, the PNA returns as much applicable raw data as possible using S-parameter measurements on the selected channel. Data that is not available is zero-filled. For example, if correction is NOT applied and the active measurement is S11, and an S21 measurement also exists on the channel, then data is returned for the S11 and S21 measurements. Data for S12 and S22 is not available and therefore returned as zeros in Real/Imaginary format. In Log Mag/Phase format, this appears as -200 dB and 45 degrees.
- **IMPORTANT** - ALL valid data is saved using the same format and settings (trace math, offset, delay, and so forth) as the active measurement. This can cause the data that is saved for the non-active measurements to be dramatically different from the data that is displayed. For example, when saving an S2P file, if the active S11 measurement is set to Data/Mem (data divided by memory), then ALL 4 S-parameters are saved using Data/Mem. The memory trace that is used in the Data/Mem operation is the same as that used in the active (S11) measurement.
- Before saving measurement data, always [trigger a single](#) measurement, and then allow the PNA channel to go into Hold. This ensures that the entire measurement trace is saved.
- Learn about [FCA parameters that are saved to an S2P file](#).

## .SNP Data Output

.SNP files contain header information, stimulus data, a response data pair for EACH S-parameter measurement. The only difference between .s1p, s2p, and so forth, is the number of S-parameters that are saved.

The following is a sample of **Header information**:

```
!Agilent Technologies,E8362B,US42340026,Q.03.54
!Agilent E8362B: Q.03.54
!Date: Friday, April 25, 2003 13:46:41
!Correction: S11(Full 2 Port SOLT,1,2) S21(Full 2 Port SOLT,1,2) S12(Full 2 Port
SOLT,1,2) S22(Full 2 Port SOLT,1,2)
!S2P File: Measurements:S11,S21,S12,S22:
# Hz S R I R 50
```

**Note:** Although the following shows Real / Imag pairs, the format could also be LogMag / Phase or LinMag / Phase

### \*.s1p Files

Each record contains 1 stimulus value and 1 S-parameter (total of 3 values)

Stim Real (Sxx) Imag(Sxx)

### \*.s2p Files

Each record contains 1 stimulus value and 4 S-parameters (total of 9 values)

Stim Real (S11) Imag(S11) Real(S21) Imag(S21) Real(S12) Imag(S12) Real(S22) Imag(S22)

### \*.s3p Files

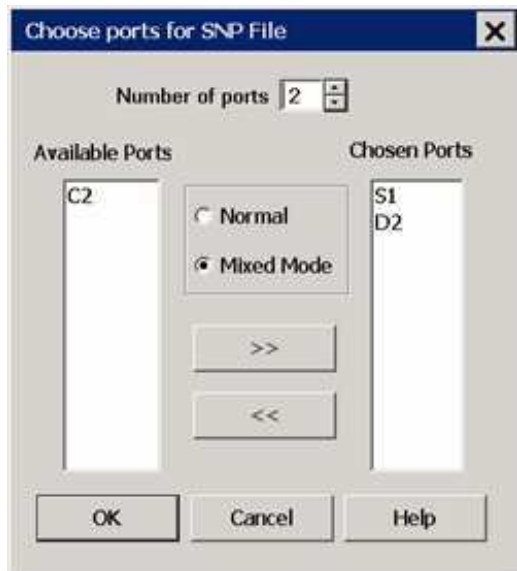
Each record contains 1 stimulus value and 9 S-parameters (total of 19 values)

Stim Real (S11) Imag(S11) Real(S12) Imag(S12) Real(S13) Imag(S13)  
Real (S21) Imag(S21) Real(S22) Imag(S22) Real(S23) Imag(S23)  
Real (S31) Imag(S31) Real(S32) Imag(S32) Real(S33) Imag(S33)

### \*.s4p Files (and so forth...)

Each record contains 1 stimulus value and 16 S-parameters (total of 33 values)

Stim Real (S11) Imag(S11) Real(S12) Imag(S12) Real(S13) Imag(S13) Real(S14) Imag(S14)  
Real (S21) Imag(S21) Real(S22) Imag(S22) Real(S23) Imag(S23) Real(S24) Imag(S24)  
Real (S31) Imag(S31) Real(S32) Imag(S32) Real(S33) Imag(S33) Real(S34) Imag(S34)  
Real (S41) Imag(S41) Real(S42) Imag(S42) Real(S43) Imag(S43) Real(S44) Imag(S44)



### Choose ports for SNP File dialog box help

This dialog appears when selecting File, [Save As](#), Trace SNP, and you request less data than is available, or you want data for more than 4 ports, or a balanced measurement is active. This dialog allows you to choose which S-parameter data to save.

**Number of ports** Select the number of ports for which data will be saved.

The following buttons appear when a [Balanced measurement](#) is displayed.

**Normal** Click to save normal (single-ended) port data.

**Mixed Mode** Click to save balanced (logical) port data. Choices are based on the [topology selection](#) for current active parameter:

- **SE-Bal:** Choose from S1, D2, C2 (Single-ended port 1, Differential port 2, Common port 2)
- **SE, SE, Bal:** Choose from S1, S2, D3, C3 (Single-ended port 1, Single-ended port 2, Differential port 3, Common port 3)
- **Bal-Bal:** Choose from D1, C1, D2, C2 (Differential port 1, Common port 1, Differential port 2, Common port 2)

For example, with SE-Bal topology, choose 2 ports, S1 for first, and D2 for second. The following 4 parameters are saved: Sss11, Ssd12, Sds21, Sdd22.

**Arrow buttons** Click to Add and Remove ports for the following columns:

**Available Ports** The PNA / External test set ports. There may NOT be valid data available for all of these ports. [Learn more.](#)

**Chosen Ports** When **OK** is clicked, SNP data is saved for these ports.

**OK** Becomes available when the number of **Chosen ports** = the **Number of ports** to save. Click to save to SNP file.

With **Number of ports** = 2, .s2p data is saved; with **Number of ports** = 3, .s3p data is saved, and so forth.

[Learn more about SNP files](#)



## .cti CitiFiles

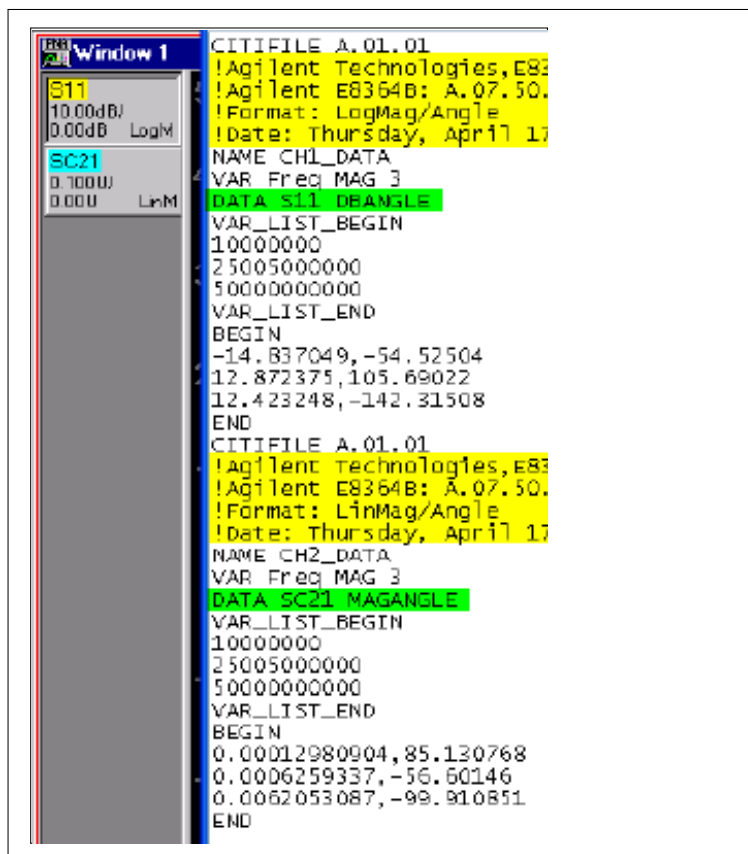
Citifile format is compatible with the Agilent 8510 Network Analyzer and Agilent's Microwave Design System (MDS).

You can do the following using citifiles :

- save the active trace, or all traces.
- save formatted or unformatted citifile data

### To save Formatted \*.cti data

1. Click **File** then **Save Data As**
2. Select **Citifile Formatted Data (\*.cti)**



The above image is a Citifile opened in Notepad. There are two traces in separate channels - one is an FCA trace. Each trace has 3 data points. The save settings = **Displayed Traces** Content, and **Auto** Format.

Format is identified by DBANGLE (log mag), MAGANGLE (Lin Mag), or RI (real, imaginary - NOT shown)

On the [data access map](#), Formatted data is taken from

location 2 or 4.

### To save Unformatted \*.cti data

1. Click **File**, then **Save Data As**
2. Select **Citifile Data Data (\*.cti)**

On the [data access map](#), Unformatted data is taken from the block just before Format.

Citifiles can be recalled and viewed in the PNA. [Learn more.](#)

### \*.csv Files

**Note:** 2D Gain Compression data is saved as \*.csv files using a different format than shown here. [Learn more.](#)

CSV files are read by spreadsheet programs such as Microsoft Excel.

### To save \*.csv files:

1. Click **File**, then **Save Data As**
2. Under **Save as type**, select **CSV Formatted Data**

\*.csv files contain: header information and the following **Comma-Separated Values**.

- Stimulus data
- Data pairs for EACH S-parameter

```
!CSV A,01.01
!Agilent Technologies, N5242A, US473000004, A.08.20.01
!Format: LogMag/Angle
!Date: Thursday, October 23, 2008 11:14:01

BEGIN CH1_DATA
!Freq(Hz), S11(DB), S11(DEG), S12(DB), S12(DEG), S21(DB), S21(DEG), S22(DB),
S22(DEG)
1e+009, 0.042540751, -0.024109257, 0.56931657, 0.042540751, -
0.024109257, 0.56931657,
...
...
...
END

BEGIN CH2_DATA
!Freq(Hz), S11(DB), S11(DEG), S12(DB), S12(DEG), S21(DB), S21(DEG), S22(DB),
S22(DEG)
1e+009, 0.042540751, -0.024109257, 0.56931657, 0.042540751, -
0.024109257, 0.56931657,
...
...
...
END
```

## \*.mdf Files

MDIF files are compatible with Agilent ADS (Advanced Design System). [Learn more at the Agilent website.](#)

### To save \*.mdf files:

1. Click **File**, then **Save Data As**
2. Under **Save as type**, select **MDIF Data**

\*.mdf files contain: header information and space-separated data:

- Stimulus data
- Real and Imaginary data pair for EACH S-parameter measurement

```
!MDF A.01.01
!Agilent Technologies, N5242A, US473000004, A.08.20.01
!Format: LogMag/Angle
!Date: Thursday, October 23, 2008 11:14:01

BEGIN CH1_DATA
%Freq (real) S[1,1] (complex) S[1,2] (complex) S[2,1] (complex) S[2,2] (complex)
1e+009 0.042540751 -0.024109257 0.56931657 0.042540751, -
0.024109257 0.56931657
...
...
...
END

BEGIN CH2_DATA
%Freq (Hz) S[1,1] (complex) S[1,2] (complex) S[2,1] (complex) S[2,2] (complex)
1e+009 0.042540751 -0.024109257 0.56931657 0.042540751 -
0.024109257 0.56931657
...
...
...
END
```

## Define Data Saves

### How to select Define Data Saves

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[Define Data Saves]**

#### Using a mouse with PNA Menus

1. Click **File**
2. then **Define Data Saves**

◀ **Programming Commands** ▶

## Define Data Saves dialog box help

**Note:** Although these settings are still supported, they are no longer necessary to save data files. The [Save Data As](#) dialog box has these settings.

The following settings survive an Instrument Preset and PNA Shutdown.

### CitiFile, CSV, and MDIF Contents

Determines what is saved to a .cti file.

**Auto** - Saves the active trace. Additional traces are saved if correction is ON. For Full 2-port calibration, 4 traces are saved; for Full 3-port calibration, 9 traces are saved, and so forth.

**Single Trace** - Saves the active trace.

**Displayed Traces** - Saves all displayed traces for all channels.

### Citifile and CSV Format

**Auto** - Data is saved in LogMag or LinMag if one of these is the currently selected display format. If format is other than these, then data is saved in Real/Imag.

**LogMag, LinMag, Real/Imag** - Select output format.

- The imaginary portion for all LogMag and LinMag data is saved in degrees.
- Real/ Imag data is never smoothed.

### SnP Format (.s1p, .s2p, .s3p)

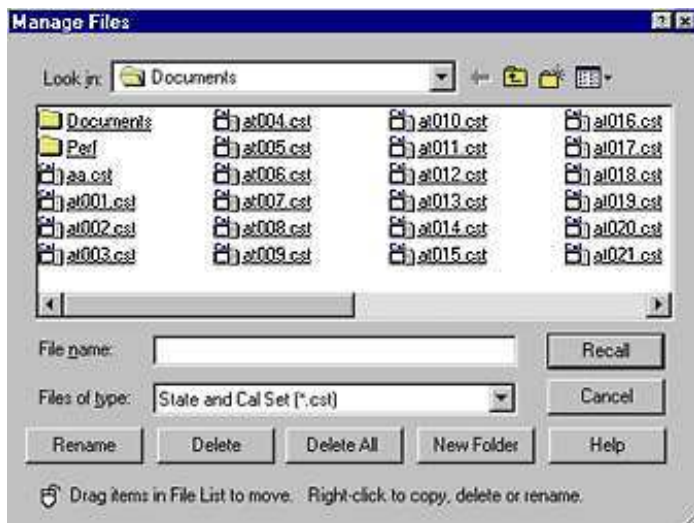
[Learn more about SnP files.](#)

**Auto** - Data is saved in LogMag or LinMag if one of these is the currently selected format. If format is other than these, then data is saved in Real/Imag.

**LogMag, LinMag, Real/Imag** - Select output format. The imaginary portion for all LogMag and LinMag data is output is in degrees.

## Manage Files without a Mouse

The Manage Files dialog box is designed to be used from the front panel. It performs the same function as Windows Explorer, but can be used without the use of a mouse or keyboard.



### Manage Files dialog box help

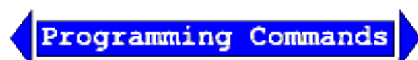
**Recall** Opens a Network Analyzer file already stored in memory.

**Rename** Renames a file that is selected in the open folder.

**Delete** Removes a selected file from the open folder.

**Delete All** Removes all files of the file type selected that appear in the open folder.

**New folder** Create a new folder and give it a name.



Last modified:

- 11-Nov-2009 Added snp fixturing note
- 3-Sep-2008 Removed legacy content
- 22-Aug-2008 Clarified invalid SNP data as -200 dB.
- 9-Jun-2008 Added no smoothing for real/imag cti files.
- 12-May-2008 Removed csv detail
- 17-Apr-2008 Added clarification and image to define data saves.
- 17-Oct-2007 Added note for MM
- 10/23/06 Added pcs note
- 9/18/06 MQ Added choose ports for snp
- 9/12/06 Added link to programming commands

## Drive Mapping

---

Drive mapping allows you to share disk drives between the PNA and an external computer. You can either map from the PNA, or from your PC, to the other.

- [From the PNA, map to a drive on an External PC](#)
- [From an External PC, map to a drive on the PNA](#)

### To prepare for Drive Mapping:

1. Both the PC and PNA must be connected to a shared computer network
2. You must know the full computer name of the PC (or analyzer) you are mapping **TO**. [Tell me how](#)
3. Your logon and password on the analyzer must be the same as that on the external PC. You can add your PC logon to the analyzer. [Tell me how](#)

**Note:** These procedures require a mouse and keyboard. Also, the external PC must have Windows NT 4.0 (or later).

### From the Analyzer, map to a drive on the External PC

1. On the external computer desktop, go to **Windows Explorer**. In the listing of drives, right click on the drive you want to share. Click **Sharing**.
2. In the dialog box, select **Shared As**. In the **Share Name** box, use the arrow key or type in a share name for the drive. For example: **C\$**. Click **OK**.
3. On the analyzer desktop, click **Windows Explorer**. From the **Tools** menu, click **Map Network Drive**. (To get to the analyzer desktop, click **View**, then click **Title Bars**)
4. If you would like to connect to your external PC using a different logon, click **Connect using a different Logon**. This logon must be registered on the analyzer and you must be currently logged on the external PC using this logon.
  1. In the **Connect as** box, type your logon name.  
The logon name and password must be exactly the same on both the external PC and the analyzer.
  2. In the **Password** box, type the logon password that you use on the external computer. Click **OK**. The logon name and password must be exactly the same on both the external PC and the analyzer.
5. In the **Folder** box, type /(full computer name of analyzer)/share name (from step 2). (For example: **/SLT1234/C\$** )
6. Click **Finish**.

## From an External PC, map to a drive on the Analyzer

1. On the analyzer desktop, click **Windows Explorer**. Right click on the drive you want to share. Click on **Sharing...**
2. In the dialog box, select **Shared this folder**. In the **Share Name** box, type in a share name for the drive. For example: **C\$**. Click **OK**.
3. On the external PC desktop, click **Windows Explorer**. From the **Tools** menu, click **Map Network Drive**.
4. If the current logon on your PC is different from the current logon on the analyzer, click **Connect using a different Logon** to connect to using the current analyzer logon, .This logon must be registered on the external PC. To see the current logon on either the PC or analyzer, hold **Ctrl - Alt**, and press **Delete**.
  1. In the **Connect as** box, type the logon currently being used by the analyzer.
  2. In the **Password** box, type the logon password that you use on the external computer. Click **OK**
5. In the **Folder** box, type *//computername ([prep1](#))/share name* (from step 2). (For example: *//SLT1234/C\$* )
6. Click **Finish**.

## Print a Displayed Measurement

---

The analyzer allows you to print a displayed measurement to a printer or to a file. The printer can be either networked or local.

- [Connecting a Printer](#)
- [Printing](#)

### Other Outputting Data topics

## Connecting a Printer

You can connect a printer to one of the PNA USB ports or to the LAN connector.

**CAUTION:** Do NOT connect your printer to the 25-pin female port labeled **Ext. Test Set Interface**. Voltage levels of signal lines may damage the printer's I/O.

### To Add a Printer

**Note:** If you try to print from the PNA application and the **Add Printer Wizard** appears, click **Cancel** and add the printer using the following procedure.

1. From the PNA application, click **View** then click **Minimize Application**
2. On the Windows taskbar, click **Start**, point to **Settings**, then click **Printers**.
3. Double-click **Add Printer**.
4. Follow the instructions in the **Add Printer** Wizard.

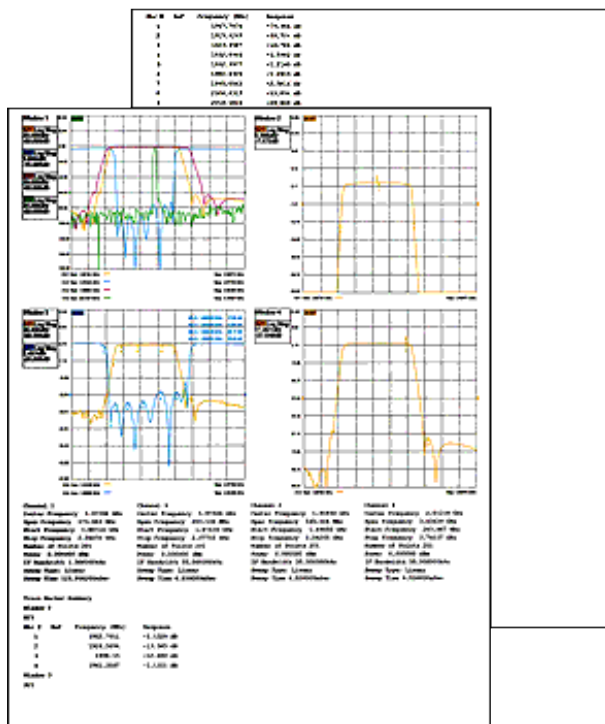
For more information, refer to Microsoft Windows Help or your printer documentation.

## Printing

- [Print a Hardcopy](#)
- [Page Setup](#)
- [Print to File](#)

The measurement information on the screen can be printed to any local or networked printer that is connected to the PNA. The graphic below shows an example of how a screen-capture image appears when printed. The [Page Setup](#) settings allows you to customize the printed form of the measurement information.





## How to Print a Hardcopy

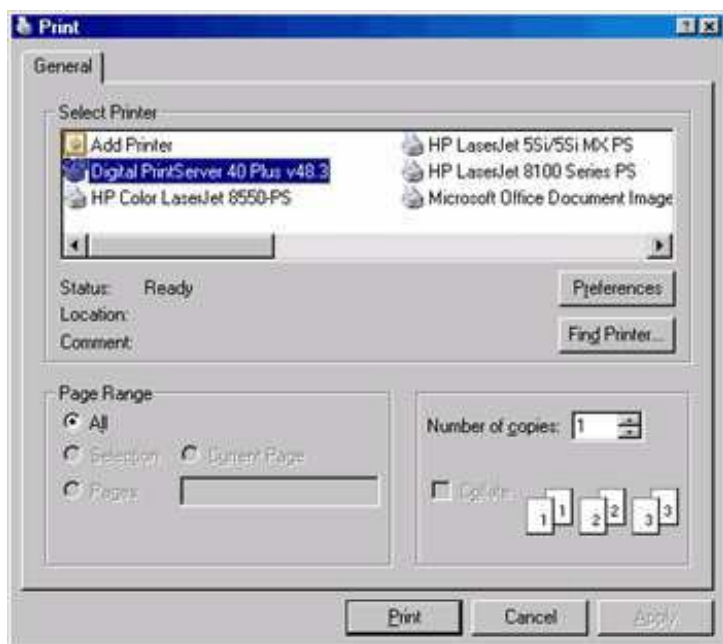
### Using front-panel HARDKEY [softkey] buttons

1. Press **PRINT**
2. then **[Print]**

### Using a mouse with PNA Menus

1. Click **File**
2. then **Print**
3. then **Print**

No programming commands are available for this feature.



**Note:** For information on the choices in the Print dialog box, see Windows Help.

## Page Setup

The Page Setup dialog allows flexibility in the appearance that measurement data is printed. After setting up the page, click **File**, then **Print...** to obtain a hard-copy.

### How to select Page Setup

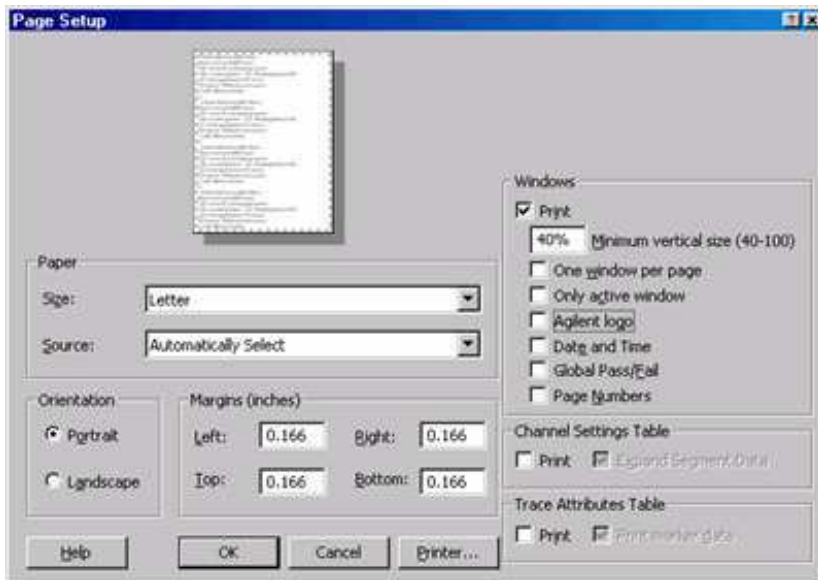
#### Using front-panel HARDKEY [softkey] buttons

1. Press **PRINT**
2. then **[Page Setup]**

#### Using a mouse with PNA Menus

1. Click **File**
2. then **Print**
3. then **Print Options**

◀ **Programming Commands** ▶



## Page Setup dialog box help

### Paper, Orientation, and Margins

These settings do NOT survive a PNA shutdown.

See Windows Help for information on these settings.

The following PNA-specific settings DO survive a PNA shutdown.

### Windows

**Minimum vertical size** Adjust to change the amount of a page that the measurement window fills. The adjustment range is from 40 to 100%.

**One window per page** Check to print one window per page. Clear to print all selected windows without a forced page break.

**Only active window** Check to print only the active window. Clear to print all windows.

**Agilent logo** Check to print the Agilent logo to the header.

**Date and Time** Check to add the current date and time to the header.

**Global Pass/Fail** Check to add the Global Pass/Fail status to the header.

**Page Numbers** Check to add page numbers (1 of n) to the header.

### Channel Settings Table

**Print** Check to print the channel settings table.

**Expand segment data** Check to print segment sweep data.

### Trace Attributes Table

**Print** Check to print the Trace Attributes Table. The Trace Attributes are measurement type, correction factors ON or OFF, smoothing, options, and marker details. The Trace Attributes are listed by Trace ID# for each window.

Each Trace ID# can have multiple entries depending on the number of markers associated with the trace. The marker details are marker number, position and response. If there are multiple markers on a trace, the trace attributes are only shown for the first marker. However, the trace attributes for the first marker apply to all other markers on that trace.

The options column can have one or more options. **D** for Delay, **M** for Marker, **G** for Gating. Multiple options selected would appear as follows: DMG.

**Print marker data** Check to print all marker data. The amount of data depends on how many markers are created.

## Print to a File

The analyzer can save a screen-capture image in any of the following formats:

- **.png** (preferred format)
- **.bmp** (bitmap)
- **.jpg**

The analyzer automatically saves the file to the current path. If not previously defined, the analyzer automatically selects the default path C:/Program Files/Agilent/Network Analyzer/Documents/

A .bmp file, like a .prn file, can be imported into software applications such as Microsoft Excel, Word, or Paint to display a screen-capture image.

[See Save and Recall files for more information.](#)

### How to Print to a File

#### Using front-panel HARDKEY [softkey] buttons

1. Press **PRINT**
2. then **[Print to File]**

#### Using a mouse with PNA Menus

1. Click **File**
2. then **Print**
3. then **Print to File**

◀ **Programming Commands** ▶

Last modified:

- 3-Sep-2008    Removed legacy content
- 23-Apr-2008    Added preference survive note.
- 10/19/06    Modified for new print dialog



## Programming Guide

---

**Two ways** to find programming commands:

**1.** [From a simulated PNA User Interface](#)

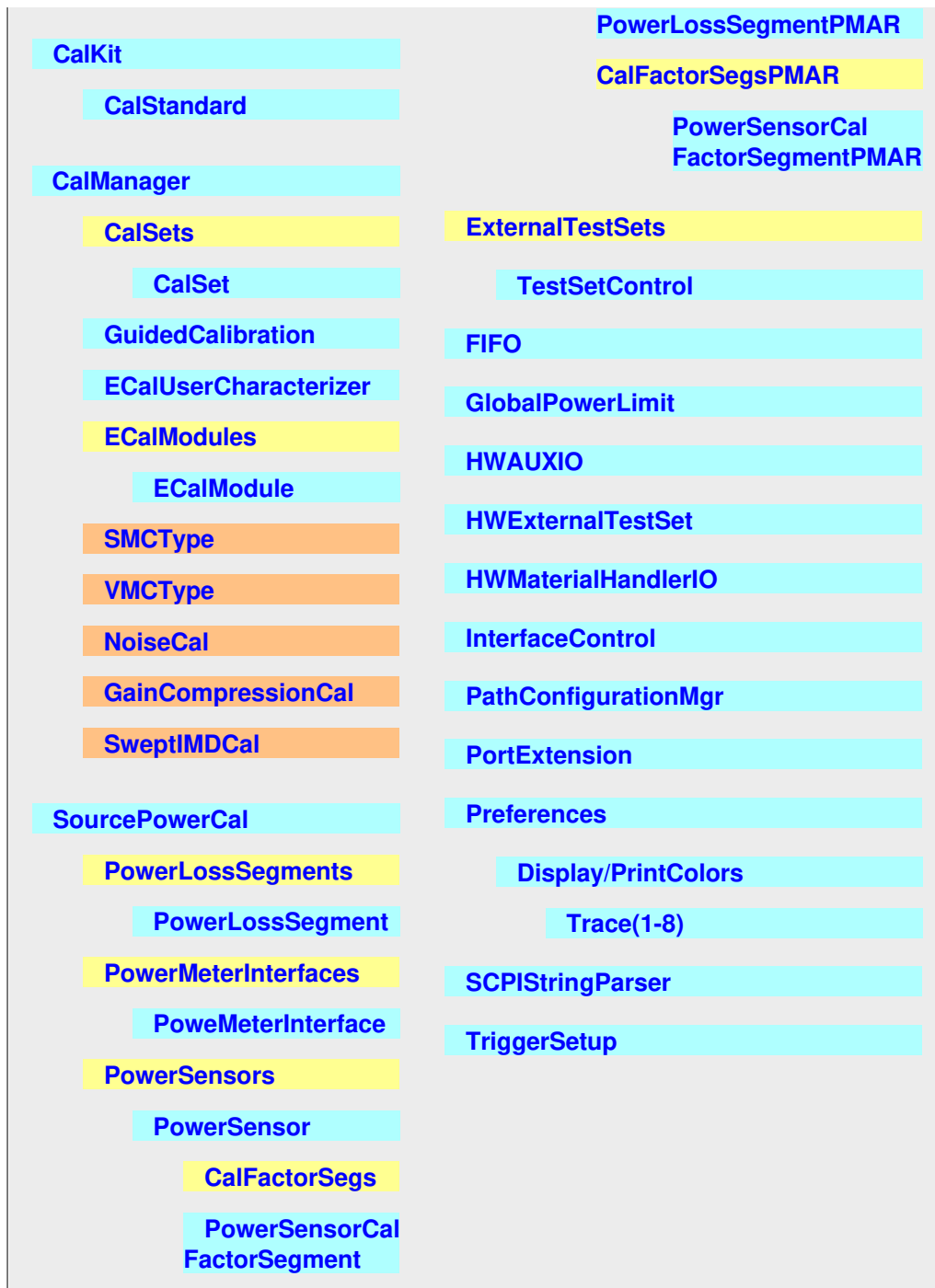
File Trace/Chan Response Marker/Analysis Stimulus Utility

	GPIB / SCPI	COM
<b>2.</b>	<a href="#">Command Tree</a>	<a href="#">COM Object Model</a>
<b>See Also</b>	<a href="#">Example Programs</a> <a href="#">Learning about GPIB</a>	<a href="#">Example Programs</a> <a href="#">Learning about COM</a>

- [New Programming Commands](#)
- [Remotely Specifying a Source Port](#)
- [Shut Down or Restart the PNA Remotely](#)
- [LXI Compliance](#)
- [VEE Examples](#) with runtime installed.
- [Using Macros](#)
- [Code Translator App.](#)
- [Superseded / Replacement Commands](#)
- [Data Access Map](#)
- See more PNA programming information and examples at: <http://na.tm.agilent.com/pna/programming/>

## PNA Object Model





**Legend:**

Object

Collection

Interface



Last Modified:

19-Feb-2010	Added PSAT, PNOP, GDAperture (9.2)
27-Aug-2009	Added External Devices
7-Apr-2009	Added ECalModules/Module
13-Nov-2007	Replaced image with text Added NFA, ENRFile, and GCA

## Application Object

### Description

The Application object is the highest object in the PNA [object model](#). This object presents methods and properties that affect the entire analyzer, rather than a specific channel or measurement. For example, the application object provides the GetIDString method. There's only one ID string for the instrument, unrelated to the channel or parameter being measured. Likewise, the TriggerSignal Property is global to the instrument. You can elect to use an internally generated (free run) trigger or a manual trigger. Either way, that type of trigger generation will be used on all measurements, on all channels. Therefore, it is under the Application object.

### Accessing the Application object

This object is unique in that you must **create** this object rather than just get a handle to it.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)
```

Replace <analyzerName> with the full computer name of your PNA. For example, "My PNA". See [Change Computer Name](#).

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Getting a Handle to an Object](#)
- [Example Programs](#)
- [Superseded commands](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Interface	Description
	<a href="#">See History</a>	
<a href="#">ActivateWindow</a>	IApplication	Makes a window object the Active Window.
<a href="#">AllowAllEvents</a>	IApplication	Monitors all events
<a href="#">AllowEventCategory</a>	IApplication	Monitors an event category
<a href="#">AllowEventMessage</a>	IApplication	Monitors an event
<a href="#">AllowEventSeverity</a>	IApplication	Monitors an event severity level
<a href="#">BuildHybridKit</a>	IApplication	Defines the user kit as port1kit + port2kit.
<a href="#">Channel</a>	IApplication	Returns a handle to the channel object for the supplied channel number.

<a href="#">Configure</a>	IAApplication9	Restarts as an "N-port" PNA using the specified multiport test set.
<a href="#">CreateCustomMeasurementEx</a>	IAApplication3	Creates a new custom measurement with initialization.
<a href="#">CreateCustomMeasurement</a>	IAApplication	<b>Superseded with</b> <a href="#">CreateCustomMeasurementEx Method</a>
<a href="#">CreateMeasurement</a>	IAApplication	Creates a new measurement.
<a href="#">CreateSParameter</a>	IAApplication	<b>Superseded with</b> <a href="#">Create SParameterEX Method</a>
<a href="#">CreateSParameterEx</a>	IAApplication	Creates a new S-Parameter measurement with a 3-port load.
<a href="#">DeleteShortCut</a>	IAApplication	Removes a macro (shortcut) from the list of macros
<a href="#">DisallowAllEvents</a>	IAApplication	Monitors NO events
<a href="#">DoPrint</a>	IAApplication	Prints the screen to the active Printer.
<a href="#">ExecuteShortcut</a>	IAApplication	Executes a macro (shortcut) stored in the analyzer.
<a href="#">GetAuxIO</a>	IAApplication	Returns a handle to the AuxIO interface
<a href="#">GetCalManager</a>	IAApplication	Returns a handle to the CalManager interface
<a href="#">GetExternalTestSetIO</a>	IAApplication	Returns a handle to the ExternalTestSet IO interface
<a href="#">GetIPConfigurationStruct</a>	IAApplication14	Returns an NA_IPConfiguration data structure which contains information about the current status of the PNA computer networking configuration.
<a href="#">GetMaterialHandlerIO</a>	IAApplication	Returns a handle to the MaterialHandlerIO interface
<a href="#">GetShortcut</a>	IAApplication	Returns the title and path of the specified macro (shortcut).
<a href="#">LANConfigurationInitialize</a>	IAApplication13	Resets the PNA LAN configuration.
<a href="#">LaunchCalWizard</a>	IAApplication	Launches the Cal Wizard
<a href="#">LaunchDialog</a>	IAApplication10	Launches the specified dialog box.
<a href="#">ManualTrigger</a>	IAApplication	Triggers the analyzer when TriggerSignal = naTriggerManual.
<a href="#">Preset</a>	IAApplication	Resets the analyzer to factory defined default settings.
<a href="#">PrintToFile</a>	IAApplication	Saves the screen data to bitmap (.bmp) file of the screen.
<a href="#">PutShortcut</a>	IAApplication	Puts a Macro (shortcut) file into the analyzer.
<a href="#">Quit</a>	IAApplication	Ends the Network Analyzer application.
<a href="#">Recall</a>	IAApplication	Recalls a measurement state, calibration state, or both from the hard drive into the analyzer.

<a href="#">RecallKits</a>	IApplication	Recalls the calibration kits definitions that were stored with the SaveKits command.
<a href="#">Reset</a>	IApplication	Removes all existing windows and measurements.
<a href="#">RestoreCalKitDefaults</a>	IApplication	Restores the factory defaults for the specified kit.
<a href="#">RestoreCalKitDefaultsAll</a>	IApplication	Restores the factory defaults for all kits.
<a href="#">Save</a>	IApplication	Saves instrument state and calibration files to disk
<a href="#">SaveCitiDataData</a>	IApplication5	Saves UNFORMATTED trace data to .cti file. <b>Superseded</b> with <a href="#">SaveData</a>
<a href="#">SaveCitiFormattedData</a>	IApplication5	Saves FORMATTED trace data to .cti file. <b>Superseded</b> with <a href="#">SaveData</a>
<a href="#">SaveData</a>	IApplication18	Saves trace data to files on disk.
<a href="#">SaveKits</a>	IApplication	Saves all cal kits to disk.
<a href="#">SetFailOnOverRange</a>	IApplication	Causes over range values to return an error code
<a href="#">SetIPConfiguration</a>	IApplication14	Modifies settings of the PNA's computer networking configuration.
<a href="#">ShowStatusBar</a>	IApplication	Shows and Hides the Status Bar.
<a href="#">ShowStimulus</a>	IApplication	Shows and Hides Stimulus information.
<a href="#">ShowTitleBars</a>	IApplication	Shows and Hides the Title Bars.
<a href="#">ShowToolbar</a>	IApplication	Shows and Hides the specified Toolbar.
<a href="#">UserPreset</a>	IApplication7	Performs a User Preset.
<a href="#">UserPresetLoadFile</a>	IApplication7	Loads an existing instrument state file (.sta or .cst) to be used for User Preset.
<a href="#">UserPresetSaveState</a>	IApplication7	Saves the current instrument settings as UserPreset.sta.

Properties	Description
------------	-------------

<a href="#">ActiveCalKit</a>	IApplication	Returns a pointer to the kit identified by kitNumber.
<a href="#">ActiveChannel</a>	IApplication	Returns a handle to the Active Channel object.
<a href="#">ActiveMeasurement</a>	IApplication	Returns a handle to the Active Measurement object.
<a href="#">ActiveNAWindow</a>	IApplication	Returns a handle to the Active Window object.

<a href="#"><u>ArrangeWindows</u></a>	IAApplication	Sets or returns the arrangement of all the windows.
<a href="#"><u>AuxiliaryTriggerCount</u></a>	IAApplication11	Returns the number of Aux trigger input / output connector pairs in the instrument.
<a href="#"><u>CalKitType</u></a>	IAApplication	Sets or returns the calibration kit type for to be used for calibration or for kit modification. Shared with the CalKit object.
<a href="#"><u>Capabilities</u></a>	IAApplication4	Return capabilities of the remote PNA.
<a href="#"><u>Channels</u></a>	IAApplication	Collection for iterating through the channels
<a href="#"><u>CoupledMarkers</u></a>	IAApplication	Sets (or reads) coupled markers ON and OFF
<a href="#"><u>DisplayAutomationErrors</u></a>	IAApplication2	Enables or disables automation error messages from being displayed on the screen. U
<a href="#"><u>DisplayGlobalPassFail</u></a>	IAApplication6	Shows or hides the dialog which displays global pass/fail results.
<a href="#"><u>E5091Testsets</u></a>	IAApplication8	Collection to control the E5091A testset.
<a href="#"><u>ENRFile</u></a>	IAApplication13	Manages Noise ENR files.
<a href="#"><u>ExternalALC</u></a>	IAApplication	Sets or returns the source of the analyzer leveling control.
<a href="#"><u>ExternalDevices</u></a>	IAApplication16	Collection to control External Devices
<a href="#"><u>ExternalTestsets</u></a>	IAApplication9	Collection to control External Test sets.
<a href="#"><u>FIFO</u></a>	IAApplication15	Controls FIFO settings
<a href="#"><u>GlobalPowerLimit</u></a>	IAApplication17	Controls Global Power Limit settings
<a href="#"><u>GPIBAddress</u></a>	IAApplication8	Sets and returns the PNA GPIB address.
<a href="#"><u>GPIBMode</u></a>	IAApplication	Makes the analyzer the system controller or a talker/listener.
<a href="#"><u>GridLineType</u></a>	IAApplication17	Set and return the line type of the window grid (solid   dotted)
<a href="#"><u>IDString</u></a>	IAApplication	Returns the model, serial number and software revision of the analyzer
<a href="#"><u>InterfaceControl</u></a>	IAApplication8	Control the Interface control features.
<a href="#"><u>LANConfiguration</u></a>	IAApplication13	Returns the current status of the PNA computer networking configuration.

<a href="#">LXIDeviceIDState</a>	IAApplication14	Displays the LAN Status dialog with LAN Status Indicator showing IDENTIFY.
<a href="#">LocalLockoutState</a>	IAApplication4	Prevents use of the mouse, keyboard, and front panel while your program is running.
<a href="#">Measurement</a>	IAApplication	Create and manage measurements
<a href="#">Measurements</a>	IAApplication	Collection for iterating through the Application measurements.
<a href="#">MessageText</a>	IAApplication	Returns text for the specified eventID
<a href="#">NaWindows</a>	IAApplication	Collection for iterating through the Application windows.
<a href="#">NoiseSourceState</a>	IAApplication13	Sets and Reads the ON   OFF state of the noise source
<a href="#">NumberOfPorts</a>	IAApplication	Returns the number of hardware source ports on the PNA
<a href="#">Options</a>	IAApplication	Returns the options on the analyzer
<a href="#">PathConfigurationManager</a>	IAApplication11	Provides access to hardware configuration.
<a href="#">Port Extensions</a>	IAApplication	<b>Superseded with</b> <a href="#">Fixturing Object</a>
<a href="#">Preferences</a>	IAApplication5	Preferences for saving citifiles.
<a href="#">ScpiStringParser</a>	IAApplication	Provides the ability to send a SCPI command from within the COM command.
<a href="#">SecurityLevel</a>	IAApplication4	Turns ON or OFF the display of frequency information.
<a href="#">SICL</a>	IAApplication5	Allows control of the PNA via SICL
<a href="#">SICLAddress</a>	IAApplication8	Sets and returns the PNA SICL address
<a href="#">SourcePowerCalibrator</a>	IAApplication2	Allows capability for performing source power calibrations.
<a href="#">SourcePowerState</a>	IAApplication	Turns Source Power ON and OFF.
<a href="#">SystemImpedanceZ0</a>	IAApplication	Sets the analyzer impedance value.
<a href="#">SystemName</a>	IAApplication	Returns the full computer name of the PNA.
<a href="#">Touchscreen</a>	IAApplication12	Enables and disables touchscreen.
<a href="#">TriggerDelay</a>	IAApplication	Sets or returns the delay time for a trigger.
<a href="#">TriggerSetup</a>	IAApplication4	Controls triggering for the entire PNA application.

<a href="#">TriggerSignal</a>	IApplication	<b>Superseded with</b> <a href="#">Source Property</a>
<a href="#">TriggerType</a>	IApplication	<b>Superseded with</b> <a href="#">Scope Property</a>
<a href="#">UserPresetEnable</a>	IApplication7	'Checks' and 'clears' the enable box on the User Preset dialog box.
<a href="#">VelocityFactor</a>	IApplication	Sets the velocity factor to be used with Electrical Delay, Port Extensions, and Time Domain marker distance calculations.
<a href="#">Visible</a>	IApplication	Makes the Network Analyzer application visible or not visible.
<a href="#">WindowState</a>	IApplication	Sets or returns the window setting of Maximized, Minimized, or Normal.  Shared with the NAWindow Object

Events	Interface	Description
<a href="#">OnCalEvent</a>	IApplication	Triggered by a calibration event.
<a href="#">OnChannelEvent</a>	IApplication	Triggered by a channel event.
<a href="#">OnDisplayEvent</a>	IApplication	Triggered by a display event.
<a href="#">OnHardwareEvent</a>	IApplication	Triggered by a hardware event.
<a href="#">OnMeasurementEvent</a>	IApplication	Triggered by a measurement event.
<a href="#">OnSCPIEvent</a>	IApplication	Triggered by a SCPI event.
<a href="#">OnSystemEvent</a>	IApplication	Triggered by a system event.
<a href="#">OnUserEvent</a>	IApplication	For future use

## IApplication History

Interface	Introduced with PNA Rev:
IApplication	1.0
IApplication2	3.0
IApplication3	3.2
IApplication4	3.5
IApplication5	4.0
IApplication6	5.0
IApplication7	5.0
IApplication8	5.2
IApplication9	6.0
IApplication10	7.20
IApplication11	7.20
IApplication12	7.21
IApplication13	8.0
IApplication14	8.2
IApplication15	8.34
IApplication16	9.0
IApplication17	9.0
IApplication18	9.2

---

Last Modified:

17-Oct-2007    Updated IPathConfigMgr Prop



## AuxiliaryTrigger Object

---

### Description

These properties setup Auxiliary triggering on a channel.

### Accessing the object

Use `chan.AuxTrigger (n)` to access the object.

where **n**= the connector pair to be used for Auxiliary Triggering.

- PNA-X models: Use **1** or **2**
- All other PNA models: Use **1** - **These models do NOT have an Aux Input; only an Output. Therefore, the following 'Input' commands will return an error when sent to PNA models other than the PNA-X.**

Use [app.AuxiliaryTriggerCount](#) to determine the number of auxiliary trigger pairs on the rear panel of a PNA.

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim AuxTrig as AuxTrigger
AuxTrig = chan.AuxTrigger(2)
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Triggering in the PNA](#)
- [Example Programs](#)

Methods	Interface	Description
	<a href="#">See History (below)</a>	

None

Properties		Description
<a href="#">Delay</a> (Input)	IAuxTrigger	Specifies the delay that should be applied by the PNA after the Aux trigger input is received and before the acquisition is made
<a href="#">Enable</a>	IAuxTrigger	Turns ON / OFF the trigger output.

<a href="#"><u>HandshakeEnable</u></a> (Input)	IAuxTrigger	Turns handshake ON / OFF.
<a href="#"><u>Number</u></a>	IAuxTrigger	Reads the number of the Aux I/O pair being used.
<a href="#"><u>TriggerInPolarity</u></a> (Input)	IAuxTrigger	Specifies the polarity of the trigger IN signal to which the PNA will respond.
<a href="#"><u>TriggerInType</u></a> (Input)	IAuxTrigger	Specifies the type of Aux trigger input being supplied to the PNA
<a href="#"><u>TriggerOutDuration</u></a>	IAuxTrigger	Specifies the width of the pulse or the time that the Aux trigger output will be asserted
<a href="#"><u>TriggerOutInterval</u></a>	IAuxTrigger	Specifies how often a trigger output signal is sent.
<a href="#"><u>TriggerOutPolarity</u></a>	IAuxTrigger	Specifies the polarity of the trigger output signal being supplied by the PNA.
<a href="#"><u>TriggerOutPosition</u></a>	IAuxTrigger	Specifies whether the Aux trigger out signal is sent Before or After the acquisition.

#### IAuxTrigger History

Interface	Introduced with PNA Rev:
IAuxTrigger	7.2

---

#### Last Modified:

6-Apr-2009	Replaced N5242A with PNA-X
13-Aug-2008	Fixed example

## BalancedMeasurement Object

---

### Description

These properties set the measurement type that is used with balanced topologies.

Use the [BalancedTopology Object](#) to set the topology and port mappings for the DUT,

### Accessing the BalancedMeasurement object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim balMeas As BalancedMeasurement
Set balMeas = app.ActiveMeasurement.BalancedMeasurement
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About Balanced Measurements](#)
- [Example Programs](#)

(**Bold** Methods or Properties provide access to a child object)

Method	Description
--------	-------------

None

Property	Interface <a href="#">See History</a>	Description
----------	--	-------------

[BalancedMode](#)      IBalancedMeasurement      Sets and returns whether the balanced transform is ON or OFF.

[BalancedTopology](#)      IBalancedMeasurement      Sets and returns the topology of a balanced DUT.

[BBalMeasurement](#)      IBalancedMeasurement      Sets and returns the measurement for the Balanced - Balanced topology.

[SBalMeasurement](#)      IBalancedMeasurement      Sets and returns the measurement for the Single-Ended - Balanced topology.

[SSBMeasurement](#) IBalancedMeasurement Sets and returns the measurement for the Single-Ended - Single-Ended - Balanced topology

#### IBalancedMeasurement History

Interface	Introduced with PNA Rev:
IBalancedMeasurement	5.0

## BalancedStimulus Object

---

### Description

These properties set the values that are unique to iTMSA - Opt 460.

All other properties for iTMSA use the standard PNA commands.

### Accessing the BalancedStimulus object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim balStim As BalancedStimulus
Set balStim = app.ActiveMeasurement.BalancedMeasurement.BalancedStimulus
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About iTMSA](#)
- [Example iTMSA Program](#)

(**Bold** Methods or Properties provide access to a child object)

Method		Description
None		

Property	Interface <a href="#">See History</a>	Description
<a href="#">BalPort1PhaseOffset</a>	IBalancedStimulus	Sets balanced port 1 phase offset
<a href="#">BalPort1PowerOffset</a>	IBalancedStimulus	Sets balanced port 1 power offset
<a href="#">BalPort1StartPhase</a>	IBalancedStimulus2	Sets Phase start value for port 1
<a href="#">BalPort1StopPhase</a>	IBalancedStimulus2	Sets Phase stop value for port 1
<a href="#">BalPort2PhaseOffset</a>	IBalancedStimulus	Sets balanced port 2 phase offset
<a href="#">BalPort2PowerOffset</a>	IBalancedStimulus	Sets balanced port 2 power offset

<a href="#">BalPort2StartPhase</a>	IBalancedStimulus2	Sets Phase start value for port 2
<a href="#">BalPort2StopPhase</a>	IBalancedStimulus2	Sets Phase start value for port 2
<a href="#">Mode</a>	IBalancedStimulus	Sets Stimulus mode for balanced measurements
<a href="#">PhaseAsFixture</a>	IBalancedStimulus	Sets the state of phase offset as a fixture
<a href="#">PhaseSwpAsFixture</a>	IBalancedStimulus2	Enable Phase Sweep as fixture
<a href="#">PhaseSwpState</a>	IBalancedStimulus2	Enable Phase Sweep
<a href="#">PowerAsFixture</a>	IBalancedStimulus	Sets the state of power offset as a fixture

#### IBalancedStimulus History

Interface	Introduced with PNA Rev:
IBalancedStimulus	8.2
IBalancedStimulus2	8.5

---

#### Last Modified:

27-Feb-2009	Added IBal2 - Phase Sweep
15-May-2008	MX New topic

## BalancedTopology Object

---

### Description

The [DUTTopology](#) property sets and returns the topology of a balanced DUT.

The following methods **set** the port mappings for the DUT.

The remaining properties **return** the port mappings for the DUT.

Use the [BalancedMeasurement object](#) to set the measurement type.

### Accessing the BalancedTopology object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel

Dim balTopology as BalancedTopology
Set balTopology = chan.BalancedTopology
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About Balanced Measurements](#)
- [Example Programs](#)

Method	Interface	Description
	<a href="#">See History</a>	
<a href="#">SetBBPorts</a>	IBalancedTopology	Sets the physical port mappings for the Balanced - Balanced DUT topology.
<a href="#">SetSBPorts</a>	IBalancedTopology	Sets the physical port mappings for the Single-Ended - Balanced DUT topology.
<a href="#">SetSSBPorts</a>	IBalancedTopology	Sets the physical port mappings for the Single-Ended - Single-Ended - Balanced DUT topology.

Property	Interface	Description
<a href="#">BB_BalPort1Negative</a>	IBalancedTopology	Returns the PNA port number that is connected to the Negative side of the DUT's logical Port 1 .

<a href="#"><u>BB_BalPort1Positive</u></a>	IBalancedTopology	Returns the first positive balanced port number in the Balanced - Balanced topology
<a href="#"><u>BB_BalPort2Negative</u></a>	IBalancedTopology	Returns the second negative balanced port number in the Balanced - Balanced topology.
<a href="#"><u>BB_BalPort2Positive</u></a>	IBalancedTopology	Returns the second positive balanced port number in the Balanced - Balanced topology.
<a href="#"><u>DUTTopology</u></a>	IBalancedTopology	Sets and returns the device topology setting.
<a href="#"><u>SB_BalPortNegative</u></a>	IBalancedTopology	Returns the negative balanced port number in the Single-Ended - Balanced topology.
<a href="#"><u>SB_BalPortPositive</u></a>	IBalancedTopology	Returns the positive balanced port number in the Single-Ended - Balanced topology.
<a href="#"><u>SB_SEPort</u></a>	IBalancedTopology	Returns the single ended port number in the Single-Ended - Balanced topology.
<a href="#"><u>SSB_BalPortNegative</u></a>	IBalancedTopology	Returns the negative balanced port number in the Single-Ended - Single-Ended - Balanced topology.
<a href="#"><u>SSB_BalPortPositive</u></a>	IBalancedTopology	Returns the positive balanced port number in the Single-Ended - Single-Ended - Balanced topology
<a href="#"><u>SSB_SEPort1</u></a>	IBalancedTopology	Returns the first single ended port in the Single-Ended - Single-Ended - Balanced topology.
<a href="#"><u>SSB_SEPort2</u></a>	IBalancedTopology	Returns the second single ended port in the Single-Ended - Single-Ended - Balanced topology.

#### BalancedTopology History

Interface	Introduced with PNA Rev:
IBalancedTopology	5.0



## CalFactorSegments Collection

---

### Description

A collection object that provides a mechanism for iterating through the segments of a power sensor cal factor table. The Cal Factor table can contain up to 100 segments.

### Accessing the CalFactorSegments collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim calFact As CalFactorSegments
Set calFact = app.SourcePowerCalibrator.PowerSensors(1).CalFactorSegments
```

### See Also:

- [PowerSensorCalFactorSegment Object](#)
- [About Source Power Cal](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Add</a>	Adds a PowerSensorCalFactorSegment object to the collection
<a href="#">Item</a>	Use to get a handle to a PowerSensorCalFactorSegment object in the collection.
<a href="#">Remove</a>	Removes an object from the collection.

Properties	Description
<a href="#">Count</a>	Returns the number of objects in the collection.
<a href="#">Parent</a>	Returns a handle to the Parent object ( <a href="#">PowerSensor</a> ) of this collection.

## Calibrator Object

---

### See Also

- [Example Programs](#)
- [Calibrator Methods and Properties](#)
- [ICalData Interface](#) for putting and getting typed Calibration data.
- [Superseded commands](#)

### Description

The Calibrator object, a child of the channel, is used to perform an **Unguided** calibration.

**Note:** You can NOT perform a full 3 or 4-port using the Calibrator object. You must use the [GuidedCalibration object](#).

There must be a measurement present for the calibrator to use or you will receive a "no measurement found" error. Therefore, to perform a 2-port cal, you must have any S-parameter measurement on the channel. For a 1-port measurement, you must have the measurement (S11 or S22) on the channel. The same is true for a response measurement.

There are a number of approaches to calibration with the calibrator object:

- You can collect data yourself and download it to the ACQUISITION buffer. The acquisition buffer holds the actual measured data for each standard. See the PNA [data map](#).
  1. Calibrator.[SetCalInfo](#)
  2. Connect a standard
  3. Trigger a sweep
  4. Retrieve the data for the standard
  5. Download the data - calibrator.[putStandard](#)
  6. Repeat for each standard
  7. Calibrator.[CalculateErrorCoefficients](#)
- You can tell the calibrator to acquire a standard. In this case, the calibrator collects the data and places it in the ACQUISITION buffer.
  1. Calibrator.[SetCalInfo](#)
  2. Connect a standard
  3. Calibrator.[AcquireCalStandard2](#)
  4. Repeat for each standard
  5. Calibrator.[CalculateErrorCoefficients](#)

- You can put previously-retrieved error terms in the error correction buffer.
  - [PutErrorTerm](#)
  - Repeat for each term
  - Measurement.[Calttype](#) = pick one
- You can also "piece together" a 2-port cal from two 1-port cals (S11 and S22) and four response (thru) cals. The system will detect that all the standards needed for a 2-port cal have been acquired even though they may not have gathered at the same time.

### Accessing the Calibrator object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim cal As ICalibrator
Set cal = app.ActiveChannel.Calibrator
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Learn about reading and writing Calibration data.](#)

Methods	Interface	Description
	<a href="#">See History</a>	
<a href="#">AcquireCalConfidenceCheckECAL</a>	ICalibrator	<b>Superseded with</b> AcquireCalConfidenceCheckECALEx
<a href="#">AcquireCalConfidenceCheckECALEx</a>	<b>ICalibrator4</b>	Transfers ECAL confidence data into analyzer memory
<a href="#">AcquireCalStandard</a>	ICalibrator	<b>Superseded with</b> AcquireCalStandard2
<a href="#">AcquireCalStandard2</a>	ICalibrator	Causes the analyzer to measure a calibration standard. Also provides for sliding load.
<a href="#">CalculateErrorCoefficients</a>	ICalibrator	Generates Error Terms from standard and actual data in the error correction buffer.
<a href="#">DoECAL1Port</a>	ICalibrator	<b>Superseded with</b> DoECAL1PortEx
<a href="#">DoECAL1PortEx</a>	<b>ICalibrator4</b>	Completes a 1 port ECAL
<a href="#">DoECAL2Port</a>	ICalibrator	<b>Superseded with</b> DoECAL2PortEx

<a href="#">DoECAL2PortEx</a>	<b>ICalibrator4</b>	Completes a 2 port ECAL
<a href="#">DoneCalConfidenceCheckECAL</a>	ICalibrator	Concludes an ECAL confidence check
<a href="#">DoReceiverPowerCal</a>	<b>ICalibrator5</b>	Perform a receiver power cal.
<a href="#">DoResponseCal</a>	<b>ICalibrator9</b>	Perform a response (normalization) cal.
<a href="#">GetCalKitTypeString</a>	<b>ICalibrator8</b>	Returns information about the attached modules
<a href="#">GetECALModuleInfo</a>	ICalibrator	<b>Superseded with</b> <a href="#">GetECALModuleInfoEx</a>
<a href="#">GetECALModuleInfoEx</a>	<b>ICalibrator4</b>	Returns information about the attached module
<a href="#">getErrorTerm</a>	ICalibrator	<b>Superseded with</b> <a href="#">GetErrorTermByString</a>
<a href="#">getStandard</a>	ICalibrator	<b>Superseded with</b> <a href="#">GetStandardByString</a>
<a href="#">putErrorTerm</a>	ICalibrator	<b>Superseded with</b> <a href="#">PutErrorTermByString</a>
<a href="#">putStandard</a>	ICalibrator	<b>Superseded with</b> <a href="#">PutStandardByString</a>
<a href="#">SaveCalSets</a>	ICalibrator	<b>Superseded with</b> <a href="#">CalSet.Save</a>
<a href="#">setCalInfo</a>	ICalibrator	Specifies the type of calibration and prepares the internal state for the rest of the calibration.
Properties	Interface	Description
<a href="#">AcquisitionDirection</a>	ICalibrator	Specifies the direction in a 2-Port cal using one set of standards.
<a href="#">CalKitType</a>	ICalibrator10	Sets and returns the name of the Cal Kit to use for unguided cal.
<a href="#">CalKitTypes</a>	ICalibrator10	Returns the names of the first 50 mechanical cal kits in your PNA that can be used for unguided calibrations.
<a href="#">ECALCharacterization</a>	<b>ICalibrator2</b>	<b>Superseded with</b> <a href="#">ECALCharacterizationEx</a>
<a href="#">ECALCharacterizationEx</a>	<b>ICalibrator4</b>	Specifies which set of characterization data within an ECal module will be used for ECal operations with that module.
<a href="#">ECALCharacterizationIndexList</a>	<b>ICalibrator6</b>	Returns a list of characterizations stored in the specified ECal module.

## [ECAL Isolation](#)

**ICalibrator** Specifies whether the acquisition of the ECal calibration should include isolation or not.

## [ECALModuleNumberList](#)

**ICalibrator6** Returns a list of index numbers to be used for referring to the ECal modules that are currently attached to the PNA.

## [ECALPortMap](#)

**ICalibrator3** **Superseded with** [ECALPortMapEx](#)

## [ECALPortMapEx](#)

**ICalibrator4** Specifies which ports of the ECal module are connected to which ports of the PNA.

## [IsECALModuleFound](#)

**ICalibrator** **Superseded with** [IsECALModuleFoundEx](#)

## [IsECALModuleFoundEx](#)

**ICalibrator4** **Superseded with** [ECALCharacterizationIndexList](#) and [ECALModuleNumberList](#)

## [IsolationAveragingIncrement](#)

**ICalibrator7** Value to increase the channel's averaging factor.

## [OrientECALModule](#)

**ICalibrator3** Specifies if the PNA should perform orientation of the ECal module during calibration.

## [Simultaneous2PortAcquisition](#)

**ICalibrator** Allows the use of 2 sets of standards at the same time.

### ICalibrator History

Interface	Introduced with PNA Rev:
ICalibrator	1.0
ICalibrator2	3.1
ICalibrator3	3.1
ICalibrator4	3.5
ICalibrator5	5.0
ICalibrator6	5.26
ICalibrator7	7.21
ICalibrator8	8.1
ICalibrator9	9.1
ICalibrator10	9.2

### ICalData Interface

#### Description

Contains methods for putting Calibration data in and getting Calibration data out of the analyzer using typed data. This interface transfers data more efficiently than variant data. However, this interfaces is only usable from VB6, C, & C++. All other programming languages must use the [ICalSet interface](#).

There is also an [ICalData Interface](#) on the CalSet Object

[Learn about reading and writing Calibration data.](#)

Methods	Description
<a href="#">getErrorTermComplex</a>	Retrieves error term data
<a href="#">getStandardComplex</a>	Retrieves calibration data from the acquisition data buffer (before error-terms are applied).
<a href="#">putErrorTermComplex</a>	Puts error term data
<a href="#">putStandardComplex</a>	Puts calibration data into the acquisition data buffer (before error-terms are applied).
Properties	Description

None

#### ICalData History

Interface	Introduced with PNA Rev:
ICalData	1.0

## CalKit Object

---

### Description

The calkit object provides the properties and methods to access and modify a calibration kit.

### Accessing a CalKit object

The **active** cal kit is the kit that is selected for use in **Unguided** calibrations. To get a handle to the active kit, use the [app.ActiveCalKit](#) property. To access the CalKit object for a specific cal kit, you must first make that kit the active kit using [app.CalKitType](#).

The CalKit object behaves differently from other objects in that you can only have a handle to **one** cal kit -- the active cal kit. Therefore, when you change the CalKitType from either the [Application object](#) or the [CalKit object](#), you may also be changing the object to which you may have other references.

For example, the following example specifies two CalKit type objects and in turn, assigns them to two different variables: ck1 and ck2.

```
Dim app As AgilentPNA835x.Application
Dim ck1 As calKit
Dim ck2 As calKit

Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)
app.CalKitType = naCalKit_User1
Set ck1 = app.ActiveCalKit
ck1.Name = "My CalKit1"

app.CalKitType = naCalKit_User2
Set ck2 = app.ActiveCalKit
ck2.Name = "My CalKit2"

Print "ck1: " & ck1.Name
Print "ck2: " & ck2.Name
```

When the pointer to each of these kits is read (printed), they each have a pointer to the last kit to be assigned to the active cal kit:

```
ck1: My CalKit2
ck2: My CalKit2
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Description
---------	-------------

<a href="#">getCalStandard</a>	Returns a handle to a calibration standard for modifying its definitions.
<a href="#">GetStandardsForClass</a>	Returns the calibration standard numbers for a specified calibration class.
<a href="#">SetStandardsForClass</a>	Sets the calibration standard numbers for a specified calibration class

Properties	Description
------------	-------------

<a href="#">CalKitType</a>	Sets or returns the calibration kit type to be used for calibration or for kit modification. Shared with the Application object.
<a href="#">Name</a>	Sets and returns the name of the cal kit
<a href="#">PortLabel</a>	Labels the ports for the kit; only affects the cal wizard annotation.
<a href="#">StandardForClass</a>	<b>Superseded with</b> Use <a href="#">GetStandardForClass</a> and <a href="#">SetStandardForClass</a> . Maps a standard device to a cal class.

## ICalKit History

Interface	Introduced with PNA Rev:
-----------	--------------------------

ICalKit	1.0
---------	-----



## CalManager Object

---

### Description

Use this interface to list, save, and delete Cal Sets.

### Accessing the CalManager object

Get a handle to a the CalManager with the app.GetCalManager Method.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim mgr as ICalManager
Set mgr = app.GetCalManager
```

### See Also:

- PNA Automation Interfaces
- The PNA Object Model
- Example Programs
- **Superseded** commands

(**Bold** Methods or Properties provide access to a child object)

Methods	Interface	Description
	See History	
AllowChannelToSweepDuringCalAcquisition	ICalManager5	Specifies the channel to sweep during a Calibration.
CreateCalSet	ICalManager	Creates a new Cal Set
<b>CreateCustomCal</b>	ICalManager2	Creates an FCA cal object.
<b>CreateCustomCalEx</b>	ICalManager5	Creates a custom cal object.
DeleteCalSet	ICalManager	Deletes a Cal Set
DisplayNAWindowDuringCalAcquisition	ICalManager5	Set the 'show' state of the window to be displayed during
DisplayOnlyCalWindowDuringCalAcquisition	ICalManager5	Clears the flags for windows to be shown during calibrat
EnumerateCalSets	ICalManager4	Returns an array of Cal Set names being stored on the I
GetCalSetByGUID	ICalManager	Get a handle to a Cal Set

GetCalSetCatalog	ICalManager	<b>Superseded with</b> EnumerateCalSets
GetCalSetUsageInfo	ICalManager	Returns the Cal Set ID and Error Term ID currently in use.
GetCalTypes	ICalManager2	Query for a list of available calibration types.
GetEcalUserCharacterizer	ICalManager6	Returns the ECalUserCharacterizer object.
GetRequiredEtermNames	ICalManager2	Returns an array of strings specifying the error terms required.
SaveCalSets	ICalManager	<b>Superseded with</b> CalSet.Save
SweepOnlyCalChannelDuringCalAcquisition	ICalManager5	Clears ALL flags for channels to sweep during calibration.

## Properties

<b>Cal Sets</b>	ICalManager	Collection for iterating through all the Cal Sets in the analyzer.
<b>ECalModules Collection</b>	ICalManager7	Collection of ECal Modules that are connected to the PNA.
<b>GuidedCalibration</b>	ICalManager3	Used to perform a Guided Calibration.

## ICalManager History

Interface	Introduced with PNA Rev:
ICalManager	2.0
CalManager2	3.1
CalManager3	3.5
CalManager4	5.0
ICalManager5	8.0
ICalManager6	8.3
ICalManager7	8.5

## CalSet Object

See [ICalData Interface](#) for putting and getting typed Cal Set data.

### Description

Use this interface to query and or change the contents of a Cal Set.

### Accessing the CalSet object

Get a handle to a CalSet object by using the CalSets collection. This is done through the CalManager object with the app.[GetCalManager](#) Method.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim calst As ICalSet
Set calst = app.GetCalManager.CalSets.Item(1)
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Reading and Writing Calibration data](#)
- [Example Programs](#)
- [Superseded commands](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">CloseCalSet</a>	ICalSet	<b>Obsolete</b> - No longer necessary.
<a href="#">ComputeErrorTerms</a>	ICalSet	Computes error terms for the CalType specified by a preceding OpenCal Set call.
<a href="#">Copy</a>	ICalSet	Creates a new Cal Set and copies the current Cal Set data into it.
<a href="#">getErrorTerm</a>	ICalSet	<b>Superseded with</b> <a href="#">getErrorTermByString</a>
<a href="#">getErrorTermByString</a>	ICalSet2	Returns variant error term data by specifying the string name of the error term.
<a href="#">getErrorTermList</a>	ICalSet	<b>Superseded with</b> <a href="#">getErrorTermList2</a>
<a href="#">getErrorTermList2</a>	ICalSet2	Returns a list of error term names found in a calset.

<a href="#">GetGUID</a>	ICalSet	Returns the GUID identifying a Cal Set
<a href="#">getStandard</a>	ICalSet	<b>Superseded with</b> <a href="#">getStandardByString</a>
<a href="#">getStandardByString</a>	ICalSet2	Returns variant standard acquisition data by specifying the string name of the standard.
<a href="#">getStandardsList</a>	ICalSet	<b>Superseded with</b> <a href="#">getStandardList2</a>
<a href="#">getStandardList2</a>	ICalSet2	Returns a list of standard names found in a Cal Set.
<a href="#">HasCalType</a>	ICalSet	Verifies that the Cal Set object contains the error terms required to apply the specified CalType to an appropriate measurement.
<a href="#">OpenCalSet</a>	ICalSet	<b>Obsolete</b> - No longer necessary.
<a href="#">putErrorTerm</a>	ICalSet	<b>Superseded with</b> <a href="#">putErrorTermByString</a>
<a href="#">putErrorTermByString</a>	ICalSet2	Writes variant error term data by specifying the string name of the error term.
<a href="#">putStandard</a>	ICalSet	<b>Superseded with</b> <a href="#">putStandardByString</a>
<a href="#">putStandardByString</a>	ICalSet2	Writes variant standard acquisition data by specifying the string name of the standard.
<a href="#">Save</a>	ICalSet	Saves the current Cal Set to disk.
<a href="#">StringToNACalClass</a>	ICalSet	Converts string values from GetStandardsList into enumeration data
<a href="#">StringToNAErrorTerm2</a>	ICalSet	Converts string values from GetErrorTermList into enumeration data
Properties		Description
<a href="#">AlternateSweep</a>	ICalSet3	Reads sweep either alternate or chopped.
<a href="#">Attenuator</a>	ICalSet3	Returns the value of the attenuator control for the specified port number.
<a href="#">AttenuatorMode</a>	ICalSet3	Returns the mode of operation (auto or manual) of the attenuator control for the specified port number.
<a href="#">CouplePorts</a>	ICalSet3	Returns state of couple ports (ON or OFF)
<a href="#">CWFrequency</a>	ICalSet3	Returns CW Frequency

<a href="#"><u>Description</u></a>	ICalSet	Set or return the descriptive string assigned to the Cal Set
<a href="#"><u>DwellTime</u></a>	ICalSet3	Returns the dwell time for the channel.
<a href="#"><u>FrequencyOffsetCWOverride</u></a>	ICalSet3	Reads state of CW Override (ON or OFF)
<a href="#"><u>FrequencyOffsetDivisor</u></a>	ICalSet3	Reads Frequency Offset Divisor value
<a href="#"><u>FrequencyOffsetFrequency</u></a>	ICalSet3	Reads Offset Frequency
<a href="#"><u>FrequencyOffsetMultiplier</u></a>	ICalSet3	Reads Frequency Offset Multiplier value
<a href="#"><u>FrequencyOffsetState</u></a>	ICalSet3	Reads Frequency Offset state (ON or OFF)
<a href="#"><u>IFBandwidth</u></a>	ICalSet3	Reads IF Bandwidth of the channel
<a href="#"><u>LastModified</u></a>	ICalSet3	Reads the time stamp of when the file was last modified
<a href="#"><u>Name</u></a>	ICalSet4	Sets and returns the Cal Set name.
<a href="#"><u>NumberOfPoints</u></a>	ICalSet3	Returns the Number of Points of the channel.
<a href="#"><u>PowerSlope</u></a>	ICalSet3	Returns the Power Slope value.
<a href="#"><u>ReceiverAttenuator</u></a>	ICalSet3	Returns the value of the specified receiver attenuator control.
<a href="#"><u>StartFrequency</u></a>	ICalSet3	Returns the start frequency of the channel.
<a href="#"><u>StartPower</u></a>	ICalSet3	Returns the start power of the PNA when sweep type is set to Power Sweep.
<a href="#"><u>StimulusValues</u></a>	ICalSet3	Returns x-axis values for stimulus or response frequencies
<a href="#"><u>StopFrequency</u></a>	ICalSet3	Returns the stop frequency of the channel.
<a href="#"><u>StopPower</u></a>	ICalSet3	Returns the stop power of the PNA when sweep type is set to Power Sweep.
<a href="#"><u>SweepGenerationMode</u></a>	ICalSet3	Returns the method being used to generate a sweep: analog or stepped.
<a href="#"><u>SweepTime</u></a>	ICalSet3	Returns the sweep time of the analyzer.
<a href="#"><u>SweepType</u></a>	ICalSet3	Returns the type of X-axis sweep that is performed on a channel.
<a href="#"><u>TestPortPower</u></a>	ICalSet3	Returns the RF power level for the channel.

## ICalSet History

Interface	Introduced with PNA Rev:
ICalSet	2.0
ICalSet2	3.0
ICalSet3	3.2
ICalSet4	6.0

## ICalData Interface

### Description

Use this interface as an alternative to the ICalSet Interface to avoid using variants when transmitting data to and from the Cal Set

[Learn about reading and writing Calibration data.](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">getErrorTermComplex</a>	ICalData2	<b>Superseded with</b> <a href="#">getErrorTermComplexByString</a>
<a href="#">getErrorTermComplexByString</a>	ICalData3	Returns typed error term data by specifying the string name of the error term.
<a href="#">getStandardComplex</a>	ICalData2	<b>Superseded with</b> <a href="#">getStandardComplexByString</a>
<a href="#">getStandardComplexByString</a>	ICalData3	Returns typed standard acquisition data by specifying the string name of the standard.
<a href="#">putErrorTermComplex</a>	ICalData2	<b>Superseded with</b> <a href="#">putErrorTermComplexByString</a>
<a href="#">putErrorTermComplexByString</a>	ICalData3	Writes typed error term data by specifying the string name of the error term.
<a href="#">putStandardComplex</a>	ICalData2	<b>Superseded with</b> <a href="#">putStandardComplexByString</a>
<a href="#">putStandardComplexByString</a>	ICalData3	Writes typed standard acquisition data by specifying the string name of the standard.
Properties	Description	

---

None

## History

Interface	Introduced with PNA Rev:
-----------	--------------------------

The original ICalData Interface was introduced with PNA 1.0 on the [Calibrator](#) Object.

ICalData2	2.0
-----------	-----

ICalData3	3.1
-----------	-----

---

Last modified:

Nov. 1, 2006    New start and stop freq commands added

## Cal Sets Collection

---

### Description

A collection object that provides a mechanism for iterating through all the Cal Sets in the analyzer. There is no ordering to the items in the collection. Therefore make no assumptions about the formatting of the collection.

For the Item and Remove methods, you can specify either the Cal Set string name, or the integer item of the Cal Set in the collection.

### Accessing the CalSets collection

Get a handle to the CalSets collection through the CalManager object with the app.[GetCalManager](#) Method.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim calsts As CalSets
Set calsts = app.GetCalManager.CalSets
```

### See Also:

- [Cal Set Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Item</a>	Returns a handle to a Cal Set object in the collection.
<a href="#">Remove</a>	Deletes the Cal Set residing at position index in the collection.

Properties	Description
<a href="#">Count</a>	Returns the number of Cal Sets in the collection.

---

Last Modified:

30-Oct-2007    added item and remove note.



## CalStandard Object

### Description

Contains all of the settings that are required to modify a calibration standard.

For more information, read [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(Application Note 1287-11\)](#)

### Accessing the CalStandard object

Get a handle to a standard with the calkit.[GetCalStandard](#) Method.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim std As ICalStandard
Set std = app.ActiveCalKit.GetCalStandard(1)
std.Delay = 0.00000003
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Reading and Writing Calibration data](#)
- [Example Programs](#)

### Methods

None

Properties	Interface	Description
	<a href="#">See History</a>	
<a href="#">C0</a>	ICalStandard	Sets and Returns the C0 (C-zero) value (the first capacitance value) for the calibration standard, when the Type is set to "naOpen".
<a href="#">C1</a>	ICalStandard	Sets and Returns the C1 value (the second capacitance value) for the calibration standard, when the Type is set to "naOpen".
<a href="#">C2</a>	ICalStandard	Sets and Returns the C2 value (the third capacitance value) for the calibration standard, when the Type is set to "naOpen".
<a href="#">C3</a>	ICalStandard	Sets and Returns the C3 value (the fourth capacitance value) for the calibration standard, when the Type is set to "naOpen".
<a href="#">Delay</a>	ICalStandard	Sets and Returns the electrical delay value for the calibration standard.

<a href="#"><u>L0</u></a>	ICalStandard	Sets and Returns the L0 (L-zero) value (the first inductance value) for the calibration standard, when the Type is set to "naShort".
<a href="#"><u>L1</u></a>	ICalStandard	Sets and Returns the L1 value (the second inductance value) for the calibration standard, when the Type is set to "naShort"..
<a href="#"><u>L2</u></a>	ICalStandard	Sets and Returns the L2 value (the third inductance value) for the calibration standard, when the Type is set to "naShort"..
<a href="#"><u>L3</u></a>	ICalStandard	Sets and Returns the L3 value (the third inductance value) for the calibration standard, when the Type is set to "naShort"..
<a href="#"><u>Label</u></a>	ICalStandard	Sets and Returns the label for the calibration standard.
<a href="#"><u>loss</u></a>	ICalStandard	Sets and Returns the insertion loss for the calibration standard.
<a href="#"><u>Maximum Frequency</u></a>	ICalStandard	Sets and Returns the maximum frequency for the calibration standard.
<a href="#"><u>Medium</u></a>	ICalStandard	Sets and Returns the media type of the calibration standard.
<a href="#"><u>Minimum Frequency</u></a>	ICalStandard	Sets and Returns the minumum frequency for the calibration standard.
<a href="#"><u>Type</u></a>	ICalStandard	Sets and Returns the type of calibration standard. Selections are: <b>naOpen, naShort, naLoad, naThru, naArbitraryImpedance and naSliding.</b>
<a href="#"><u>TZReal</u></a>	<b>ICalStandard2</b>	Sets and Returns the TZReal value (the Real Terminal Impedance value) for the calibration standard, when the Type is set to "naArbitraryImpedance".
<a href="#"><u>TZImag</u></a>	<b>ICalStandard2</b>	Sets and Returns the TZImag value (the Imaginary Terminal Impedance value) for the calibration standard, when the Type is set to "naArbitraryImpedance".
<a href="#"><u>Z0</u></a>	ICalStandard	Sets and Returns the characteristic impedance for the calibration standard.

#### ICalStandard History

Interface	Introduced with PNA Rev:
CalStandard	1.0
CalStandard2	3.0

# Capabilities Object

## Description

These properties return capabilities of the remote PNA.

## Accessing the Capabilities object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim cap As Capabilities
Set cap = app.Capabilities
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [ICapabilities History](#)
- [Example Programs](#)

Methods	Interface
	<a href="#">See History</a>

<a href="#">GetPortNumber Method</a>	ICapabilities4 Returns the port number for the specified string port name.
--------------------------------------	--

Properties	Description
<a href="#">CpuRevision</a>	ICapabilities6 Returns the CPU speed of the PNA
<a href="#">DspRevision</a>	ICapabilities6 Returns the DSP Revision number
<a href="#">DspFpgaRevision</a>	ICapabilities6 Returns the DSP FPGA Revision number
<a href="#">FirmwareMajorRevision</a>	ICapabilities Returns integer portion of firmware revision number.
<a href="#">FirmwareMinorRevision</a>	ICapabilities Return decimal portion of firmware revision number.
<a href="#">FirmwareSeries</a>	ICapabilities Returns the Alpha portion of the firmware revision number.
<a href="#">GPIBPortCount</a>	ICapabilities3 Returns the number of GPIB ports (1or 2)
<a href="#">InternalTestsetPortCount</a>	ICapabilities Returns the number of PNA test ports.

<a href="#"><u>IsFrequencyOffsetPresent</u></a>	ICapabilities	Returns the presence of Frequency Offset Option 080 (True or False).
<a href="#"><u>IsReceiverStepAttenuatorPresent</u></a>	ICapabilities	Returns the presence of receiver step attenuators (True or False).
<a href="#"><u>IsReferenceBypassSwitchPresent</u></a>	ICapabilities	Returns the presence of the reference switch (True or False).
<a href="#"><u>MaximumFrequency</u></a>	ICapabilities	Returns the maximum frequency of the PNA.
<a href="#"><u>MaximumNumberOfChannels</u></a>	ICapabilities2	Returns the maximum possible number of Channels
<a href="#"><u>MaximumNumberOfPoints</u></a>	ICapabilities	Returns the maximum possible number of data points.
<a href="#"><u>MaximumNumberOfTracesPerWindow</u></a>	ICapabilities2	Returns the maximum possible number of traces per window
<a href="#"><u>MaximumNumberOfWindows</u></a>	ICapabilities2	Returns the maximum possible number of windows
<a href="#"><u>MaximumReceiverStepAttenuator</u></a>	ICapabilities	Returns the maximum amount of receiver attenuation.
<a href="#"><u>MaximumSourceALCPower</u></a>	ICapabilities	Returns the maximum amount of source ALC power.
<a href="#"><u>MaximumSourceStepAttenuator</u></a>	ICapabilities	Returns the maximum amount of source attenuation.
<a href="#"><u>MinimumFrequency</u></a>	ICapabilities	Returns the minimum frequency of the PNA.
<a href="#"><u>MinimumNumberOfPoints</u></a>	ICapabilities	Returns the minimum possible number of data points.
<a href="#"><u>MinimumReceiverStepAttenuator</u></a>	ICapabilities	Returns the minimum amount of receiver attenuation.
<a href="#"><u>MinimumSourceALCPower</u></a>	ICapabilities	Returns the minimum amount of source ALC power.
<a href="#"><u>ReceiverCount</u></a>	ICapabilities	Returns the number of receivers in the PNA.
<a href="#"><u>ReceiverStepAttenuatorStepSize</u></a>	ICapabilities	Returns the step size of the attenuator.
<a href="#"><u>SourceStepAttenuatorStepSize</u></a>	ICapabilities5	Returns a value indicating the step size of the source attenuator.
<a href="#"><u>SourceCount</u></a>	ICapabilities	Returns the number of sources.
<a href="#"><u>SourcePortCount</u></a>	ICapabilities4	Returns the number of source ports.
<a href="#"><u>SourcePortNames</u></a>	ICapabilities4	Returns the string names of source ports.

## **ICapabilities History**

I Interface	Introduced with PNA Rev:
ICapabilities	3.5
ICapabilities2	5.23
ICapabilities3	6.0
ICapabilities4	7.20
ICapabilities5	8.04
ICapabilities6	9.10

## Channel Object

See [SourcePowerCalData Interface](#) for putting and getting typed source power calibration data.

### Description

The channel object is like the engine that produces data. Channel settings consist of stimulus values like frequency, power, IF bandwidth, and number of points.

### Accessing the Channel object

You can get a handle to a channel in a number of ways. But first you have to make sure that the channel exists. When you first startup the analyzer, there is one S11 measurement on channel 1. Thus there is only one channel in existence. You can do the following:

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim chan As IChannel
Set chan = app.ActiveChannel
```

or

```
Set chan = app.Channels(2)
```

The first method returns the channel object that is driving the active measurement. If there is no measurement, there may not be a channel. Once a channel is created, it does not go away. So if there once was a measurement (hence a channel), the channel will still be available.

If there is no channel you can create one in a couple ways. You can do the following:

```
Pna.CreateMeasurement( ch1, "S11", port1, window2)
```

or

```
Pna.Channels.Add(2)
```

The latter will have no visible effect on the analyzer. It will simply create channel 2 if it does not already exist.

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)
- [Superseded commands](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Interface	Description
	<a href="#">See History</a>	
<a href="#">Abort</a>	IChannel	Aborts the current measurement sweep on the channel.

<a href="#">ApplySourcePowerCorrectionTo</a>	IChannel11	Copies an existing Source Power Calibration to another channel.
<a href="#">AveragingRestart</a>	IChannel	Clears and restarts averaging of the measurement data.
<a href="#">Continuous</a>	IChannel	The channel continuously responds to trigger signals.
<a href="#">CopyToChannel</a>	IChannel	Sets up another channel as a copy of this objects channel.
<a href="#">GetConverter</a>	IChannel7	Returns a handle to a SweptIMD for converter object
<a href="#">GetErrorCorrection</a>	IChannel8	Returns the channel error correction state.
<a href="#">GetNumberOfGroups</a>	IChannel3	Returns the <a href="#">number of groups</a> a channel has yet to acquire.
<a href="#">GetPortNumber</a>	IChannel13	Returns the port number for the specified string port name.
<a href="#">GetRxLevelingConfiguration</a>	IChannel17	Returns a handle to a RxLevelingConfiguration object
<a href="#">getSourcePowerCalData</a>	IChannel	<b>Superseded with</b> <a href="#">Get SourcePowerCalDataEx</a>
<a href="#">getSourcePowerCalDataEx</a>	IChannel4	Returns requested source power calibration data, if it exists.
<a href="#">GetSupportedALCModes</a>	IChannel10	Returns a list of supported ALC modes
<a href="#">GetXAxisValues</a>	IChannel	Returns the channel's X-axis values into a dimensioned Variant array.
<a href="#">GetXAxisValues2</a>	IChannel	Returns the channel's X-axis values into a dimensioned NON-Variant array.
<a href="#">Hold</a>	IChannel	Puts the Channel in Hold - not sweeping.
<a href="#">Next_IFBandwidth</a>	IChannel	A function that returns the Next higher IF Bandwidth value.
<a href="#">NumberOfGroups</a>	IChannel	Sets the Number of trigger signals the channel will receive.
<a href="#">Preset</a>	IChannel	Resets the channel to factory defined settings.
<a href="#">PreviousIFBandwidth</a>	IChannel	Returns the previous IF Bandwidth value.
<a href="#">putSourcePowerCalData</a>	IChannel	<b>Superseded with</b> <a href="#">Put SourcePowerCalDataEx Method</a>
<a href="#">putSourcePowerCalDataEx</a>	IChannel4	Inputs source power calibration data to this channel for a specific source port.
<a href="#">SelectCalSet</a>	IChannel	Specifies the Cal Set to use for the Channel
<a href="#">Single</a>	IChannel	Channel responds to one trigger signal from any source (internal, external, or manual). Then channel switches to Hold.

<b>Properties</b>	<b>Interface</b>	<b>Description</b>
-------------------	------------------	--------------------

<a href="#"><u>ALCLevelingMode</u></a>	IChannel10	Set or return the ALC leveling mode.
<a href="#"><u>AlternateSweep</u></a>	IChannel	Sets sweeps to either alternate or chopped.
<a href="#"><u>Attenuator</u></a>	IChannel	Sets or returns the value of the attenuator control for the specified port number.
<a href="#"><u>AttenuatorMode</u></a>	IChannel	Sets or returns the mode of operation of the attenuator control for the specified port number.
<a href="#"><u>AuxiliaryTrigger</u></a>	IChannel10	Used to configure Auxiliary Triggering
<a href="#"><u>Averaging</u></a>	IChannel	Turns trace averaging ON or OFF for all measurements on the channel.
<a href="#"><u>AveragingCount</u></a>	IChannel	Returns the number of sweeps that have been averaged into the measurements.
<a href="#"><u>AveragingFactor</u></a>	IChannel	Specifies the number of measurement sweeps to combine for an average.
<a href="#"><u>AverageMode</u></a>	IChannel16	Sets Point or Sweep averaging.
<a href="#"><u>BalancedTopology</u></a>	IChannel6	Sets and returns the topology of a balanced DUT.
<a href="#"><u>Calibrator</u></a>	IChannel4	Used to perform an Unguided calibration.
<a href="#"><u>CalSet</u></a>	IChannel	Change the contents of a Cal Set
<a href="#"><u>centerFrequency</u></a>	IChannel	Sets or returns the center frequency of the channel. Shared with the Segment Object
<a href="#"><u>channelNumber</u></a>	IChannel	Returns the Channel number. Shared with the Measurement Object
<a href="#"><u>CoupleChannelParams</u></a>	IChannel5	Turns ON and OFF Time Domain Trace Coupling.
<a href="#"><u>CouplePorts</u></a>	IChannel	Turns ON and OFF port power coupling.
<a href="#"><u>CustomChannelConfiguration</u></a>	IChannel12	Provides access to custom application objects.
<a href="#"><u>CWFrequency</u></a>	IChannel	Set the Continuous Wave (CW) frequency.
<a href="#"><u>DwellTime</u></a>	IChannel	Sets or returns the dwell time for the channel. Shared with the Segment Object
<a href="#"><u>ErrorCorrection</u></a>	IChannel7	Attempts to sets error correction ON or OFF for all of the measurements on the channel.
<a href="#"><u>ExternalTriggerDelay</u></a>	IChannel6	Sets or returns the external trigger delay value for the channel.



<a href="#">FastCWPointCount</a>	IChannel16	Enables Fast CW sweep and sets the number of data points for the channel.
<a href="#">Fixturing</a>	IChannel6	Port Ext, Embedding, and De-embedding functions.
<a href="#">FOM Collection</a>	IChannel9	Configure Frequency Offset Measurements
<a href="#">FrequencyOffsetDivisor</a>	IChannel2	
<a href="#">FrequencyOffsetFrequency</a>	IChannel2	
<a href="#">FrequencyOffsetMultiplier</a>	IChannel2	<b>Superseded</b> with <a href="#">FOM</a> and <a href="#">FOMRange</a>
<a href="#">FrequencyOffsetCWOverride</a>	IChannel2	
<a href="#">FrequencyOffsetState</a>	IChannel2	
<a href="#">FrequencySpan</a>	IChannel	Sets or returns the frequency span of the channel. Shared with the Segment Object.
<a href="#">IFBandwidth</a>	IChannel	Sets or returns the IF Bandwidth of the channel. Shared with the Segment Object.
<a href="#">IFConfiguration</a>	IChannel4	Control the IF gain and source path settings for the H11 Option.
<a href="#">IsContinuous</a>	IChannel3	Returns whether or not a channel is in <a href="#">continuous</a> mode.
<a href="#">IsHold</a>	IChannel3	Returns whether or not a channel is in <a href="#">hold</a> mode.
<a href="#">MeasurementClass</a>	IChannel15	Returns the measurement class name.
<a href="#">NumberOfPoints</a>	IChannel	Sets or returns the Number of Points of the channel. Shared with the Segment Object.
<a href="#">Parent</a>	IChannel	Returns a handle to the parent object of the channel.
<a href="#">PathConfiguration</a>	IChannel10	Provides access to path configuration switches and setting.
<a href="#">PathConfigurationManager</a>	IChannel10	Provides access to path configuration file management.
<a href="#">PointSweepState</a>	IChannel16	Turns point sweep ON or OFF for all measurements on the channel.
<a href="#">PowerSlope</a>	IChannel	Sets or returns the Power Slope value.
<a href="#">PulseGenerator</a>	IChannel10	Provides access to pulse generator configuration.

<a href="#"><b>PulseMeasurementControl</b></a>	IChannel20	Provides access to pulse measurement settings.
<a href="#">R1InputPath</a>	IChannel2	Throws internal reference switch ( <a href="#">option 081</a> ).
<a href="#">ReceiverAttenuator</a>	IChannel	Sets or returns the value of the specified receiver attenuator control.
<a href="#">ReduceIFBandwidth</a>	IChannel5	Sets or returns the state of the <a href="#">Reduced IF Bandwidth at Low Frequencies</a> setting.
<a href="#"><b>Segments</b></a>	IChannel	Collection for iterating through the sweep segments of a channel.
<a href="#">SourcePortCount</a>	IChannel13	Returns the number of source ports.
<a href="#">SourcePortMode</a>	IChannel9	Sets the state of the PNA sources. (AUTO   ON   OFF)
<a href="#">SourcePortNames</a>	IChannel13	Returns the string names of source ports.
<a href="#">SourcePowerCalPowerOffset</a>	IChannel4	Sets or returns a power level offset from the PNA test port power.
<a href="#">SourcePowerCorrection</a>	IChannel	Turns source power correction ON or OFF for a specific source port.
<a href="#">StartFrequency</a>	IChannel	Sets or returns the start frequency of the channel. Shared with the Segment Object
<a href="#">StartPower</a>	IChannel	Sets the start power of the analyzer when sweep type is set to Power Sweep.
<a href="#">StartPowerEx</a>	IChannel13	Sets and reads the power sweep start power value for a specific port.
<a href="#">StopFrequency</a>	IChannel	Sets or returns the stop frequency of the channel. Shared with the Segment Object
<a href="#">StopPower</a>	IChannel	Sets the Stop Power of the analyzer when sweep type is set to Power Sweep.
<a href="#">StopPowerEx</a>	IChannel13	Sets and reads the power sweep stop power value for a specific port.
<a href="#">SweepDelay</a>	IChannel19	Sets the time to wait just before acquisition begins for each sweep.
<a href="#">SweepGenerationMode</a>	IChannel	Sets the method used to generate a sweep: continuous ramp (analog) or discrete steps (stepped).
<a href="#">SweepSpeedMode</a>	IChannel14	Set or returns the sweep speed mode.
<a href="#">SweepTime</a>	IChannel	Sets the Sweep time of the analyzer.

<a href="#">SweepType</a>	IChannel	Sets the type of X-axis sweep that is performed on a channel.
<a href="#">TestPortPower</a>	IChannel	Sets or returns the RF power level for the channel. Shared with the Segment Object
<a href="#">TriggerMode</a>	IChannel	Determines the measurement that occurs when a trigger signal is sent to the channel.
<a href="#">UserRangeMax</a>	IChannel	Sets the stimulus stop value for the specified User Range.
<a href="#">UserRangeMin</a>	IChannel	Sets the stimulus start value for the specified User Range.
<a href="#">XAxisPointSpacing</a>	IChannel	Sets X-Axis point spacing for the active channel.

### IChannel History

Interface	Introduced with PNA Rev:
IChannel	1.0
IChannel2	3.0
IChannel3	4.0
IChannel4	4.0
IChannel5	4.2
IChannel6	5.0
IChannel7	5.2
IChannel8	6.0
IChannel9	7.0
IChannel10	7.2
IChannel11	7.5
IChannel12	8.0
IChannel13	8.2
IChannel14	8.33
IChannel15	8.33
IChannel16	8.35

IChannel17	8.55
IChannel19	9.20
IChannel20	9.20

## ISourcePowerCalData Interface

### Description

Contains methods for putting source power calibration data in and getting source power calibration data out of the analyzer using typed data. The methods in this interface transfer data more efficiently than methods that use variant data. However, this interfaces is only usable from VB6, C, & C++. All other programming languages must use the methods on the Channel Object.

**Note:** Interface **ISourcePowerCalData** is abbreviated as **ISPCD** in the following table.

Methods	Interface	Description
	<a href="#">See History</a>	
<a href="#">getSourcePowerCalDataScalar</a>	ISPCD	<b>Superseded with</b> - use <a href="#">PutSourcePowerCalDataScalarEx Method</a>
<a href="#">getSourcePowerCalDataScalarEx</a>	<b>ISPCD2</b>	Returns requested source power calibration data, if it exists.
<a href="#">putSourcePowerCalDataScalar</a>	ISPCD	<b>Superseded with</b> - use <a href="#">PutSourcePowerCalDataEx Method</a>
<a href="#">putSourcePowerCalDataScalarEx</a>	<b>ISPCD2</b>	Inputs source power calibration data to a channel, for a specific source port.
Properties	Description	

None

### ISourcePowerCalData History

Interface	Introduced with PNA Rev:
ISourcePowerCalData	2.0
ISourcePowerCalData2	4.0

## Channels Collection

---

### Description

A collection object that provides a mechanism for iterating through the channels

Collections are, by definition, unordered lists of like objects. You cannot assume that Channels.Item(1) is always Channel 1.

### Accessing the Channels collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim chans As Channels
Set chans = app.Channels
```

### See Also:

- [Channel Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface	Description
<a href="#">Add</a>	IChannels	An alternate way to create a measurement.
<a href="#">Hold</a>	IChannels	Places all channels in Hold trigger mode.
<a href="#">Item</a>	<b>IChannels2</b>	Use to get a handle on a channel in the collection.
<a href="#">Resume</a>	<b>IChannels2</b>	Resumes the trigger mode of all channels that was in effect before sending the channels.Hold method.

Properties		Description
<a href="#">Count</a>	IChannels	Returns the number of channels in the analyzer.
<a href="#">Parent</a>	IChannels	Returns a handle to the current Application.
<a href="#">UnusedChannelNumbers</a>	<b>IChannels2</b>	Returns an array of channel numbers that are NOT in use.
<a href="#">UsedChannelNumbers</a>	<b>IChannels2</b>	Returns an array of channel numbers that are in use.



## Converter Object (Opt 087 and FCA (Opt 082 / 083)

---

### Description

Contains the methods and properties to setup the mixer for an IMDx channel.

This object and IMixer can be used to configure FCA measurements.

Use the SweptIMD object to configure the IMDx properties.

Use the SweptIMDCal Object to perform a calibration on the SweptIMDx channel.

### Accessing the Converter Interface

```
Dim app as AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application")
Dim chan As IChannel
Set chan = app.ActiveChannel
Dim convert
Set convert = chan.GetConverter
```

### See Also:

- IMDx for Converters
- PNA Automation Interfaces
- The PNA Object Model

Methods	Interface	Description
	See History	
Apply	Converter	Applies mixer settings.
AssignSourceToRole	Converter	Assigns a configured source to the specified role.
Calculate	Converter	Automatically calculate Input and Output frequencies for mixer setup.
DiscardChanges	Converter	Cancels changes that have been made to the Converter setup.
GetSourceByRole	Converter	Returns the name of a source that is assigned to the specified role.
GetSourceRoles	Converter	Returns the defined role names ("RF2", "LO1").
LoadFile	Converter	Loads a previously-configured mixer attributes file (.mxr )

SaveFile	Converter	Saves the settings for the mixer/converter test setup to a mixer attributes file.
----------	-----------	---

## Properties

## Description

ActiveXAxisRange	Converter	Sets or returns the swept frequency range to display on the X-axis.
<b>ConverterEmbeddedLO</b>	Converter2	Provides access to the ConverterEmbeddedLO Object
EnablePhase	Converter3	Sets and returns the state of SMC + Phase measurements and calibrations.
InputDenominator	Converter	Sets or returns the denominator value of the Input Fractional Multiplier.
InputFixedFrequency	Converter	Sets or returns the mixer fixed Input frequency value.
InputNumerator	Converter	Sets or returns the numerator value of the Input Fractional Multiplier.
InputPower	Converter	Sets or returns the value of the Input Power.
InputRangeMode	Converter	Sets or returns the Input sweep mode.
InputStartFrequency	Converter	Sets or returns the start frequency of the mixer input.
InputStopFrequency	Converter	Sets or returns the stop frequency of the mixer input.
IsInputGreaterThanLO	Converter	Specifies whether to use the Input frequency that is greater than the LO or less
LODenominator	Converter	Sets or returns the denominator value of the LO Fractional Multiplier.
LOFixedFrequency	Converter	Sets or returns the fixed frequency of the specified LO.
LOName	Converter	Sets or returns the LO name.
LONumerator	Converter	Sets or returns the numerator value of the LO Fractional Multiplier.
LOPower	Converter	Sets or returns the value of the LO Power.
LORangeMode	Converter	Sets or returns the LO sweep mode to fixed or swept.
LOStage	Converter	Returns the number of stages.
LOStartFrequency	Converter	Sets or returns the start frequency of the specified LO.
LOStartPower	Converter	Sets or returns the start value of a LO Power sweep.
LOStopFrequency	Converter	Sets or returns the start frequency of the specified LO.
LOStopPower	Converter	Sets or returns the stop value of a LO Power sweep.



NormalizePoint	Converter3	Sets or returns the data point used for normalizing an SMC phase measurement.
OutputFixedFrequency	Converter	Sets or returns the fixed frequency of the mixer output.
OutputRangeMode	Converter	Sets or returns the Output sweep mode.
OutputSideband	Converter	Sets or returns the value of the output sideband.
OutputStartFrequency	Converter	Sets or returns the start frequency of the mixer output.
OutputStopFrequency	Converter	Sets or returns the stop frequency of the mixer output.

### Converter History

Interface	Introduced with PNA Rev:
Converter	8.55
Converter2	9.0
Converter3	9.2

---

Last Modified:

2-Feb-2009    New topic

## E5091Testsets Collection

---

### Description

Two testsets can be connected and controlled by the PNA at any time.

The item number in the testsets collection is set by the DIP switches on the testset rear-panel. The valid item numbers are 1 and 2. If the testset DIP switches are set to 1, then item number in the collection is 1, and so forth. See your E5091A documentation for more information.

If the specified testset is not connected to USB or not ON, then setting [Enabled](#) = True will return an error. All other properties can be set when the testset is not connected.

### Accessing the E5091Testsets collection

Child of the **Application** Object. Get a handle to one of the [E5091Testset objects](#) by specifying an item of the collection.

```
Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")
Dim testsets As E5091Testsets
Set testsets = pna.E5091Testsets
Dim tset1 As E5091Testset
Set tset1 = testsets(1)
```

### See Also:

- [E5091Testset Control COM Example](#)
- [E5091Testset Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)

Methods	Description
---------	-------------

Item	Use to get a handle to a testset in the collection.
------	---

Properties	Description
------------	-------------

Count	Returns the number of items in a collection of objects.
-------	---

Parent	Returns a handle to the current naNetworkAnalyzer application.
--------	--

### E5091Testsets History

Interface	Introduced with PNA Rev:
-----------	--------------------------

IE5091Testsets	5.2
----------------	-----

## E5091Testset Object

### Description

There can be two test sets connected and controlled by the PNA at any time.

The item number in the testsets collection is set by the DIP switches on the test set rear-panel. The valid item numbers are 1 and 2. If the test set DIP switches are set to 1, then item number in the collection is 1, and so forth. See your E5091A documentation for more information.

If the specified test set is not connected to USB or not ON, then setting [Enabled](#) = True will return an error. All other properties can be set when the test set is not connected.

### Accessing the E5091Testset object

Child of the **Application** Object. Get a handle to a E5091Testset object by specifying an item of the collection.

```
Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")
Dim testsets As E5091Testsets
Set testsets = pna.E5091Testsets
Dim tset1 As E5091Testset
Set tset1 = testsets(1)
```

### See Also:

- [E5091Testset Control COM Example](#)
- [E5091 TestSet Control](#)
- [E5091Testsets Collection](#)
- [TestsetControl Object](#) (for different test sets)
- [The PNA Object Model](#)

Methods	Description
---------	-------------

None

Properties	Description
------------	-------------

<a href="#">ControlLines</a>	Sets the control lines of the specified E5091A.
<a href="#">Enabled</a>	Enables and disables (ON/OFF) the port mapping and control line output of the specified testset.
<a href="#">ID</a>	Returns the test set ID number.
<a href="#">NumberOfPorts</a>	Reads the number of ports (7 or 9) that are on the specified E5091A test set.

[OutputPort](#)

Switches an input to one of the valid outputs on the specified E5091A.

[ShowProperties](#)

Turns ON and OFF the display of the test set control status bar.

#### E5091Testset History

Interface		Introduced with PNA Rev:
IE5091Testset		5.2

## ECalUserCharacterizer Object

### Description

Controls the settings used to perform an ECal User Characterization. An S-Parameter channel must already be calibrated. These commands will then measure the ECal module with adapters, cables, or fixtures to be included in the User Characterization, allow descriptive text to be entered, then save the User Characterization to the ECal module.

Up to 12 User Characterizations can be stored in an ECal module.

You can NOT perform a **remote** User Characterization of a 4-port ECal module using a 2-port PNA. This can only be done from the front panel user interface.

### Accessing the ECalUserCharacterizer Interface

Access the Interface through the ICalManager Object.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application")

Dim mgr as ICalManager
Set mgr = app.GetCalManager
Dim ecalCharacterizer
Set ecalCharacterizer = mgr.GetECalUserCharacterizer()
```

### See Also:

- Example:** Perform an ECal User Characterization
- PNA Automation Interfaces
- The PNA Object Model
- About User Characterization

Methods	Interface	Description
	See History	
AcquireStep	ECalUserCharacterizer	Measure the ECal module.
GenerateSteps	ECalUserCharacterizer	Returns the number of steps required to complete the calibration.
GetStepDescription	ECalUserCharacterizer	Returns the description of the specified step in the calibration process.
Initialize	ECalUserCharacterizer	<b>Superseded</b> with InitializeEx Method
InitializeEx	ECalUserCharacterizer2	Initiates a User Characterization of an ECal module.
SaveToDiskMemory	ECalUserCharacterizer2	Saves the User Characterization to PNA disk memory.

SaveToECal      ECalUserCharacterizer      Saves the User Characterization to the ECal module.

## Properties

## Description

CharacterizationNumber	ECalUserCharacterizer	Sets and reads the number to which the user characterization will be saved.
ConnectorType	ECalUserCharacterizer	Sets or queries the connector type for the specified port.
ECalID	ECalUserCharacterizer	Select the model and serial number of the ECal module to be characterized.
PortDescription	ECalUserCharacterizer	Sets and reads the description of the adapters, cable, or fixture to be characterized.
UserDescriptionofPNA	ECalUserCharacterizer	Sets and reads a user description of the PNA used to perform the characterization.
UserName	ECalUserCharacterizer	Sets and reads the description of the person and/or company who is performing the characterization.
ValidConnectorTypes	ECalUserCharacterizer	Returns a list of connector names that are valid for use with user-characterization.

## IECalUserCharacterizer History

Interface	Introduced with PNA Rev:
IECalUserCharacterizer	8.33
IECalUserCharacterizer2	9.00

Last Modified:

2-Nov-2008      MX New topic

## EmbeddedLO Object

---

### Description

Provides access to the properties that allow measurement of mixers that contain an embedded LO.

### Accessing the EmbeddedLO Interface

Access the Interface through the IMixer Object.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application")
app.Preset

' FCA Measurements can't share the channel with standard measurements
' Because preset creates a single measurement in channel 1, we first delete the
standard measurement
Dim standardMeas As IMeasurement
Set standardMeas = app.ActiveMeasurement
standardMeas.Delete

' Create a Measurement object, in this case using the IMeasurement interface
Dim meas As IMeasurement
Set meas = app.CreateCustomMeasurementEx(1, "SMC_Forward.SMC_ForwardMeas", "SC21")

' See if this measurement object supports IMixer
Dim mixer As IMixer
Dim embeddedLO
Set embeddedLO = mixer.EmbeddedLO
```

See an example program that shows how to create and calibrate a standard SMC or VMC measurement or a fixed output SMC measurement.

### See Also:

[PNA Automation Interfaces](#)

[The PNA Object Model](#)

[Making Embedded LO Measurements](#)



Methods	Interface	Description
	See History	

ResetLOFrequency	IEmbeddedLO	Reset LO Delta frequency.
ResetTuningParameters	IEmbeddedLO	Resets the tuning parameters to their defaults.

Properties	Description
------------	-------------

BroadbandTuningSpan	IEmbeddedLO	Set broadband sweep span.
<b>EmbeddedLODiagnostic</b>	IEmbeddedLO	Provides access to the status of tuning sweeps.
IsOn	IEmbeddedLO	Set and return Embedded LO ON   OFF.
LOFrequencyDelta	IEmbeddedLO	Sets and returns LO delta frequency.
MaxPreciseTuningIterations	IEmbeddedLO	Sets and returns precise tuning iterations.
NormalizePoint	IEmbeddedLO	Sets and returns tuning point.
PreciseTuningTolerance	IEmbeddedLO	Sets and returns precise tuning tolerance.
TuningIFBW	IEmbeddedLO	Sets and returns the IF Bandwidth for tuning sweeps.
TuningMode	IEmbeddedLO	Sets and returns the method used to determine the embedded LO Frequency.
TuningSweepInterval	IEmbeddedLO	Set how often a tuning sweep is performed.

#### IEmbeddedLO History

Interface	Introduced with PNA Rev:
IEmbeddedLO	7.21

## EmbeddedLODiagnostic Object

---

### Description

Allows access to the properties that provide information about the broadband and precise tuning of an embedded LO.

### Accessing the EmbeddedLODiagnostic Interface

Access the Interface through the EmbeddedLO Object.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application")
app.Preset

' FCA Measurements can't share the channel with standard measurements
' Because preset creates a single measurement in channel 1, we first delete the
standard measurement
Dim standardMeas As IMeasurement
Set standardMeas = app.ActiveMeasurement
standardMeas.Delete

' Create a Measurement object, in this case using the IMeasurement interface
Dim meas As IMeasurement
Set meas = app.CreateCustomMeasurementEx(1, "SMC_Forward.SMC_ForwardMeas", "SC21")

' See if this measurement object supports IMixer
Dim mixer As IMixer
Dim embeddedLO
Set embeddedLO = mixer.EmbeddedLO

Dim embeddedLODiagnostic
Set embeddedLODiagnostic = embeddedLO.EmbeddedLODiagnostic
```

See an example program that shows how to create and calibrate a standard SMC or VMC measurement or a fixed output SMC measurement.

### See Also:

[PNA Automation Interfaces](#)

[The PNA Object Model](#)

[Making Embedded LO Measurements](#)

[EmbeddedLO Object](#)

Methods	Interface	Description
	See History	

Clear	IELODiag	Clear current diagnostic information.
-------	----------	---------------------------------------

Properties		Description
------------	--	-------------

IsMarkerOn	IELODiag	Was a marker was used for a tuning sweep?
------------	----------	---

LODeltaFound	IELODiag	Returns the LO frequency delta from this tuning sweep.
--------------	----------	--

NumberOfSweeps	IELODiag	Get number of tuning sweeps.
----------------	----------	------------------------------

MarkerAnnotation	IELODiag	Get the marker annotation.
------------------	----------	----------------------------

MarkerPosition	IELODiag	Get the marker X-axis position.
----------------	----------	---------------------------------

Parameter	IELODiag	Returns the tuning sweep parameter name.
-----------	----------	--

StatusAsString	IELODiag	Get result of the last tuning sweeps.
----------------	----------	---------------------------------------

StepData	IELODiag	Get a tuning sweep data.
----------	----------	--------------------------

StepTitle	IELODiag	Returns the tuning sweep title.
-----------	----------	---------------------------------

XAxisAnnotation	IELODiag	Get the tuning sweep X axis annotation.
-----------------	----------	---

XAxisStart	IELODiag	Get the Start sweep value.
------------	----------	----------------------------

XAxisStop	IELODiag	Get the Stop sweep value.
-----------	----------	---------------------------

YAxisAnnotation	IELODiag	Get the tuning sweep Y axis annotation.
-----------------	----------	---

## History

Interface	Introduced with PNA Rev:
-----------	--------------------------

IEmbeddedLODiagnostic	7.21
-----------------------	------

## ENRFile Object

---

### Description

Provide commands for creating or editing an ENR file. This is rarely necessary as ENR files, which contain factory calibrated data, are typically provided by the manufacturer of the noise source.

[Learn more about Noise Figure Application](#)

### Accessing the ENRFile object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim enr As ENRFile
Set enr = app.ENRFile
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Program](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">GetENRData</a>	IENRFile	Read the ENR calibration data from PNA memory.
<a href="#">PutENRData</a>	IENRFile	Write the ENR calibration data to PNA memory.
<a href="#">LoadENRFile</a>	IENRFile	Recalls an ENR file from disk into PNA Memory.
<a href="#">SaveENRFile</a>	IENRFile	Saves an ENR file from PNA memory to disk.

Properties	Interface <a href="#">See History</a>	Description
<a href="#">ENRID</a>	IENRFile	Sets and returns ID of ENR table.
<a href="#">ENRSN</a>	IENRFile	Sets and returns the serial number of the noise source.

### IENRFile History

Interface	Introduced with PNA Rev:
-----------	-----------------------------

IENRFile	8.0
----------	-----

---

Last Modified:

2-Aug-2007    MX New topic

## Equation Object

---

### Description

Provide commands for creating an equation.

[Learn more about Equation Editor](#)

### Accessing the Equation object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)
```

```
Dim eq As Equation
Set eq = app.Equation
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Methods	Interface	Description
	<a href="#">See History</a>	

None

Properties	Interface	Description
	<a href="#">See History</a>	

<a href="#">Text</a>	IEquation	Sets the Equation
<a href="#">State</a>	IEquation	Sets the Equation enabled state
<a href="#">Valid</a>	IEquation	Returns whether the equation is presently valid.

### Example Program using these commands:

```

Dim na
Dim meas
Set na = CreateObject("AgilentPNA835x.Application")
Set meas = na.ActiveMeasurement
'Define the measurement
meas.Equation.Text = "mysillyequ=sqrt(AR1_1)"
'Check to see if the equation is valid
valid_e = meas.Equation.Valid
MsgBox valid_e
'Turn on the Equation Editor
meas.Equation.State = True

```

## IEquation History

Interface	Introduced with PNA Rev:
IEquation	6.03

Last Modified:

4-Dec-2007    Added example

## ExternalTestsets Collection

---

### Description

ExternalTestsets collection provides access to a TestsetControl object. Only one external testset can be controlled by the PNA at any time.

### Accessing the ExternalTestsets collection

The ExternalTestsets collection is a property of the main **Application** Object. You can obtain a handle to a testset by specifying an item in the collection.

### Visual Basic Example

```
Dim pna
Dim testsets As ExternalTestsets
Dim tset1 As TestsetControl
Set pna = CreateObject("AgilentPNA835x.Application")
Set testsets = pna.ExternalTestsets
Set tset1 = testsets(1)
' make COM calls on tset1 object
End Sub
```

### See Also:

[ExternalTestset Control COM Example](#)

[About External TestSet Control](#)

[TestsetControl Object](#)

[The PNA Object Model](#)

Methods	Description
<a href="#">Add</a>	Adds a testset to the collection and loads a test set configuration file.
<a href="#">Item</a>	Use to get a handle to a testset in the collection.
<a href="#">TestsetCatalog</a>	Returns a list of supported test sets.

Properties	Description
<a href="#">Count</a>	Returns the number of items in a collection of objects.
<a href="#">Parent</a>	Returns a handle to the current naNetworkAnalyzer application.

### ExternalTestsets History



Interface	Introduced with PNA Rev:
IExternalTestsets	6.0
IExternalTestsets	6.2

## FIFO Object

---

### Description

These properties control the First IN, First OUT (FIFO) buffer settings for the PNA-X and N5264A. The 4 GB FIFO data buffer is available with Option 118 on the [PNA-X](#) and [N5264A](#).

### Accessing the FIFO object

```
Dim app as AgilentPNA835x.Application
Dim fifo as IFIFO
Set fifo = app.IFIFO
```

### See Also:

- [About FIFO](#)
- [FIFO example program](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Methods		Description
<a href="#">Clear</a>	IFIFO	Clears the FIFO buffer

Properties	Interface <a href="#">See History</a>	Description
<a href="#">Data Property</a>	IFIFO	Reads the next specified number of data points from the FIFO buffer.
<a href="#">DataCount Property</a>	IFIFO	Returns the total number of data points in the FIFO buffer.
<a href="#">DataInCompactForm</a>	IFIFO	Reads data from the FIFO buffer. Same as Data but in a compact form.
<a href="#">State</a>	IFIFO	Turns FIFO ON and OFF

### IFIFO History

Interface	Introduced with PNA Rev:
-----------	--------------------------

IFIFO	8.35
-------	------

---

Last Modified:

6-Oct-2008    MX New topic

## Fixturing Object

---

### Description

Contains the properties for Embedding and De-embedding test fixtures.

### Accessing the Fixturing object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim fixt as Fixturing
Set fixt = chan.Fixturing
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About Fixturing](#)
- [Example Programs](#)

Methods		Description
<a href="#">AutoPortExtMeasure</a>	IFixturing2	Measures either an OPEN or SHORT standard.
<a href="#">AutoPortExtReset</a>	IFixturing2	Clears old port extension delay and loss data.

Properties	Interface <a href="#">See History</a>	Description
<a href="#">AutoPortExtConfig</a>	IFixturing2	Sets the frequency span that is used to calculate Automatic Port Extension.
<a href="#">AutoPortExtDCOffset</a>	IFixturing2	Specifies whether or not to include DC Offset as part of Automatic port extension.
<a href="#">AutoPortExtLoss</a>	IFixturing2	Specifies whether or not to include loss correction as part of Automatic Port Extension.
<a href="#">AutoPortExtSearchStart</a>	IFixturing2	Set the start frequency for custom user span.
<a href="#">AutoPortExtSearchStop</a>	IFixturing2	Set the stop frequency for custom user span.

<a href="#"><u>AutoPortExtState</u></a>	IFixturing2	Enables and disables automatic port extensions on the specified port.
<a href="#"><u>CmnModeZConvPortImag</u></a>	IFixturing2	Sets imaginary value for common port impedance conversion.
<a href="#"><u>CmnModeZConvPortReal</u></a>	IFixturing2	Sets real value for common port impedance conversion.
<a href="#"><u>CmnModeZConvState</u></a>	IFixturing2	Turns ON/OFF common port impedance conversion.
<a href="#"><u>CmnModeZConvPortZ0</u></a>	IFixturing2	Sets impedance value for common port impedance conversion.
<a href="#"><u>DiffPortMatch_C</u></a>	IFixturing2	Sets Capacitance value of the differential matching circuit.
<a href="#"><u>DiffPortMatch_G</u></a>	IFixturing2	Sets Conductance value of the differential matching circuit.
<a href="#"><u>DiffPortMatch_L</u></a>	IFixturing2	Sets Inductance value of the differential matching circuit.
<a href="#"><u>DiffPortMatch_R</u></a>	IFixturing2	Sets Resistance value of the differential matching circuit.
<a href="#"><u>DiffPortMatchMode</u></a>	IFixturing2	Sets type of circuit to embed.
<a href="#"><u>DiffPortMatchUserFilename</u></a>	IFixturing2	Specifies the 4-port touchstone file for user-defined differential matching circuit.
<a href="#"><u>DiffPortMatchState</u></a>	IFixturing2	Turns ON/OFF differential matching circuit function.
<a href="#"><u>DiffZConvPortImag</u></a>	IFixturing2	Sets imaginary value for differential port impedance conversion.
<a href="#"><u>DiffZConvPortReal</u></a>	IFixturing2	Sets real value for differential port impedance conversion.
<a href="#"><u>DiffZConvPortZ0</u></a>	IFixturing2	Sets impedance value for differential port impedance conversion.
<a href="#"><u>DiffZConvState</u></a>	IFixturing2	Turns ON/OFF differential port impedance conversion.
<a href="#"><u>Embed4PortA</u></a>	IFixturing2	Returns PNA portA connections.
<a href="#"><u>Embed4PortB</u></a>	IFixturing2	Returns PNA portB connections.
<a href="#"><u>Embed4PortC</u></a>	IFixturing2	Returns PNA portC connections.

<a href="#"><u>Embed4PortD</u></a>	IFixturing2	Returns PNA portD connections.
<a href="#"><u>Embed4PortList</u></a>	IFixturing2	Specifies all PNA port connections.
<a href="#"><u>Embed4PortNetworkFilename</u></a>	IFixturing2	Specifies *.s4p filename.
<a href="#"><u>Embed4PortNetworkMode</u></a>	IFixturing2	Specify embed, de-embed, or none.
<a href="#"><u>Embed4PortState</u></a>	IFixturing2	Turns ON or OFF 4-port Network Embed/De-embed.
<a href="#"><u>Embed4PortTopology</u></a>	IFixturing2	Specifies the PNA / DUT topology.
<a href="#"><u>EnablePowerCompensation</u></a>	IFixturing5	Compensates source power for combined loss through all fixturing functions.
<a href="#"><u>FixturingState</u></a>	IFixturing	Turns Fixturing ON and OFF on this channel.
<a href="#"><u>Port2PdeembedCktModel</u></a>	IFixturing	Sets and returns the 2 port De-embedding circuit model for the specified port number.
<a href="#"><u>Port2PdeembedState</u></a>	IFixturing	Turns 2 port de-embedding ON and OFF on this channel.
<a href="#"><u>PortArbzImag</u></a>	IFixturing3	Sets and returns the imaginary impedance value for the specified single-ended port number.
<a href="#"><u>PortArbzReal</u></a>	IFixturing3	Sets and returns the real impedance value for the specified single-ended port number.
<a href="#"><u>PortArbzState</u></a>	IFixturing	Turns single-ended port impedance ON and OFF on the specified channel.
<a href="#"><u>PortArbzZ0</u></a>	IFixturing3	Sets and returns the real and imaginary impedance value for the specified single-ended port number.
<a href="#"><u>PortCoupleToSystemMedia</u></a>	IFixturing4	Couples to system Media type
<a href="#"><u>PortCoupleToSystemVelocity</u></a>	IFixturing4	Couples to system Velocity Factor
<a href="#"><u>PortDelay</u></a>	IFixturing	Sets and returns the Port Delay value for the specified port number.
<a href="#"><u>PortDistance</u></a>	IFixturing4	Sets Port Ext in distance
<a href="#"><u>PortDistanceUnit</u></a>	IFixturing4	Sets distance units

<a href="#"><u>PortExtState</u></a>	IFixturing	Turns Port Extension ON and OFF on this channel.
<a href="#"><u>PortExtUse1</u></a>	IFixturing	Sets and returns the USE1 ON/OFF state for the Loss1 and Freq1 values for the specified port number.
<a href="#"><u>PortExtUse2</u></a>	IFixturing	Sets and returns the USE2 ON/OFF state for the Loss2 and Freq2 values for the specified port number.
<a href="#"><u>PortFreq1</u></a>	IFixturing	Sets and returns the 1st Port Frequency value for the specified port number.
<a href="#"><u>PortFreq2</u></a>	IFixturing	Sets and returns the 2nd Port Frequency value for the specified port number.
<a href="#"><u>PortLoss1</u></a>	IFixturing	Sets and returns the 1st Port Loss value for the specified port number.
<a href="#"><u>PortLoss2</u></a>	IFixturing	Sets and returns the 2nd Port Loss value for the specified port number.
<a href="#"><u>PortLossDC</u></a>	IFixturing	Sets and returns the Port Loss at DC value for the specified port number.
<a href="#"><u>PortMatching_C</u></a>	IFixturing	Sets and returns the Capacitance, 'C' value for the specified port number.
<a href="#"><u>PortMatching_G</u></a>	IFixturing	Sets and returns the Conductance, 'G' value for the specified port number.
<a href="#"><u>PortMatching_L</u></a>	IFixturing	Sets and returns the Inductance, 'L' value for the specified port number.
<a href="#"><u>PortMatching_R</u></a>	IFixturing	Sets and returns the Resistance, 'R' value for the specified port number.
<a href="#"><u>PortMatchingCktModel</u></a>	IFixturing	Sets and returns the Port Matching circuit model for the specified port number.
<a href="#"><u>PortMatchingState</u></a>	IFixturing	Turns Port Matching ON and OFF on this channel.
<a href="#"><u>PortMedium</u></a>	IFixturing4	Sets Media per port
<a href="#"><u>PortVelocityFactor</u></a>	IFixturing4	Set Velocity Factor per port

<a href="#"><u>PortWGCutoffFreq</u></a>	IFixturing4	Sets waveguide cutoff frequency per port
<a href="#"><u>strPort2Pdeembed_S2PFile</u></a>	IFixturing	Sets and returns the 2 port De-embedding 'S2P' file name for the specified port number.
<a href="#"><u>strPortMatch_S2PFile</u></a>	IFixturing	Sets and returns the Port Matching 'S2P' file name for the specified port number.

#### IFixturing History

Interface	Introduced with PNA Rev:
IFixturing	5.0
IFixturing2	5.2
IFixturing3	5.25
IFixturing4	8.50
IFixturing5	9.20



## FOM Collection

---

### Description

The FOM collection provides access to the source and receiver range objects which are used for configuring frequency offset measurements.

The FOM range items are typically numbered as follows:

1. Primary
2. Source
3. Receivers
4. Source2 (if present)

### Accessing the FOM Collection and [FOMRange](#) objects

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel

Dim ifom as FOM
Set ifom = chan.FOM

ifom.item(2).Coupled = false
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About FOM](#)
- [Example Programs](#)

Method	Interface	Description
<a href="#">See History</a>		

[Item](#) IFOM

Property	Interface	Description
<a href="#">DisplayRange</a>	IFOM	Sets the range to be displayed on the PNA x-axis.
<a href="#">FOMRange</a>	IFOM	Object
<a href="#">RangeCount</a>	IFOM	Returns the number of FOM ranges available on the PNA.
<a href="#">State</a>	IFOM	Turns Frequency Offset ON and OFF.

#### FOM History

Interface	Introduced with PNA Rev:
IFOM	7.10

Last Modified:

8-Mar-2007    Modified Access

## FOMRange Object

---

### Description

The FOM Range object provides access to the properties and methods for configuring a specific Range for frequency offset measurements.

### Accessing an FOMRange object

Get a handle to a FOM Range by specifying an item in the [FOM collection](#).

The FOM range items are typically numbered as follows:

1. Primary
2. Source
3. Receivers
4. Source2 (if present)

```
Dim app as AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim ranges as FOM
Set ranges = app.ActiveChannel.FOM
ranges.item(2).Coupled = False
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About FOM](#)
- [Example Programs](#)

Method	Interface	Description
	<a href="#">See History</a>	

None

Property	Interface	Description
<a href="#">Coupled</a>	IFOMRange	Sets and returns the state of coupling (ON or OFF) of this range to the primary range.
<a href="#">CWFrequency</a>	IFOMRange	Set the Continuous Wave (CW) frequency.
<a href="#">Divisor</a>	IFOMRange	Sets and returns the Divisor value to be used when coupling this range to the primary range.
<a href="#">Multiplier</a>	IFOMRange	Sets and returns the Multiplier value to be used when coupling this range to the primary range.
<a href="#">Name</a>	IFOMRange	Returns the name of this FOM range object.
<a href="#">Offset</a>	IFOMRange	Sets and returns the offset value to be used when coupling this range to the primary range.
<a href="#">rangeNumber</a>	IFOMRange	Returns the index number of the range within the <a href="#">FOM collection</a> .
<a href="#">Segments</a>	IFOMRange	Collection - Used to add segment sweep capability to a range.
<a href="#">StartFrequency</a>	IFOMRange	Sets or returns the start frequency of this FOM Range.
<a href="#">StopFrequency</a>	IFOMRange	Sets or returns the stop frequency of this FOM Range.
<a href="#">Sweep Type</a>	IFOMRange	Sets the type of range sweep.

**Note:** Use the [Start Power](#) and [Stop Power](#) settings from the [channel object](#).

#### FOM History

Interface	Introduced with PNA Rev:
IFOM	7.10

Last Modified:

7-Jan-2008    Added Start/Stop power note

7-Mar-2007    Modified Receivers

## Gain Compression Object

---

### Description

Controls the Gain Compression Application settings.

### Accessing the GainCompression object

```
Dim app as AgilentPNA835x.Application
app.CreateCustomMeasurementEx(1, "Gain Compression", "CompIn21", 1)
Dim GCA
Set GCA = app.ActiveChannel.CustomChannelConfiguration
```

### See Also:

- **Example Program** [Create and Cal a Gain Compression Measurement](#)
- [GainCompressionCal Object](#)
- [About Gain Compression Application](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

**Note:** Set the Start/Stop Frequency and Start/Stop Power Settings using the [Channel Object](#).

Method	Interface	Description
<a href="#">See History</a>		
<a href="#">GetRaw2DData</a>	IGainCompression	Reads Gain Compression data from specified location.
<a href="#">GetDataIm</a>	IGainCompression	Reads Imaginary part of specified frequency or power points.
<a href="#">GetDataRe</a>	IGainCompression	Reads REAL part of specified frequency or power points.
<a href="#">SetPortMap</a>	IGainCompression	Maps the PNA ports to the DUT ports
Property	Interface	Description

<a href="#"><u>AcquisitionMode</u></a>	IGainCompression	Set and read the method by which gain compression data is acquired.
<a href="#"><u>CompressionAlgorithm</u></a>	IGainCompression	Set and read the algorithm method used to compute gain compression.
<a href="#"><u>CompressionBackoff</u></a>	IGainCompression	Set and read value for the BackOff compression algorithm.
<a href="#"><u>CompressionDeltaX</u></a>	IGainCompression	Set and read the 'X' value in the delta X/Y compression algorithm.
<a href="#"><u>CompressionDeltaY</u></a>	IGainCompression	Set and read the 'Y' value in the delta X/Y compression algorithm.
<a href="#"><u>CompressionInterpolation</u></a>	IGainCompression	Sets whether or not to interpolate the final power level when the measured compression level deviates from the specified level.
<a href="#"><u>CompressionLevel</u></a>	IGainCompression	Set and read the decrease in gain which indicates that the amplifier is compressing.
<a href="#"><u>DeviceInputPort</u></a>	IGainCompression	Read the PNA port number which is connected to the DUT input.
<a href="#"><u>DeviceOutputPort</u></a>	IGainCompression	Read the PNA port number which is connected to the DUT output.
<a href="#"><u>EndOfSweepOperation</u></a>	IGainCompression	Set and read the action which should be taken at the end of the last frequency or power sweep in the measurement.
<a href="#"><u>InputLinearPowerLevel</u></a>	IGainCompression	Set and read the input power level that should produce linear gain.
<a href="#"><u>MaximumNumberOfPoints</u></a>	IGainCompression	Returns the maximum possible number of data points.
<a href="#"><u>NumberOfFrequencyPoints</u></a>	IGainCompression	Set and read the number of data points in each frequency sweep.
<a href="#"><u>NumberOfPowerPoints</u></a>	IGainCompression	Set and read the number of data points in each power sweep.
<a href="#"><u>ReverseLinearPowerLevel</u></a>	IGainCompression	Set and read the reverse power level to the DUT.

<a href="#"><u>SafeSweepCoarsePowerAdjustment</u></a>	IGainCompression	Set and read the Safe Sweep COARSE power adjustment.
<a href="#"><u>SafeSweepEnable</u></a>	IGainCompression	Set and read the (ON   OFF) state of Safe Sweep mode.
<a href="#"><u>SafeSweepFinePowerAdjustment</u></a>	IGainCompression	Set and read the Safe Sweep FINE power adjustment.
<a href="#"><u>SafeSweepFineThreshold</u></a>	IGainCompression	Set and read the compression level in which Safe Sweep changes from the COARSE power adjustment to the FINE power adjustment.
<a href="#"><u>SafeSweepMaximumLimit</u></a>	IGainCompression4	When the DUT Output reaches this value, the input power to the DUT is no longer incremented at that frequency.
<a href="#"><u>SaturationLevel</u></a>	IGainCompression3	Set and read the deviation dB from the maximum Pout.
<a href="#"><u>SearchFailures</u></a>	IGainCompression	Read number of points that did not achieve compression.
<a href="#"><u>SearchSummary</u></a>	IGainCompression	Returns if a compression search is complete, and if the search was a success or failure.
<a href="#"><u>SmartSweepMaximumIterations</u></a>	IGainCompression	Set and read the maximum number of iterations to be used to find the compression level in a SMART sweep.
<a href="#"><u>SmartSweepSettlingTime</u></a>	IGainCompression	Set and read SMART sweep settling time.
<a href="#"><u>SmartSweepShowIterations</u></a>	IGainCompression	Set and read whether to show results for each SMART sweep iteration.
<a href="#"><u>SmartSweepTolerance</u></a>	IGainCompression	Set and read the level of tolerance to be used to find the compression level in a SMART sweep.
<a href="#"><u>TotalIterations</u></a>	IGainCompression2	Returns the total number of iteration required for the SMART Sweep.
<a href="#"><u>TotalNumberOfPoints</u></a>	IGainCompression	Set and read the total number of data points.(Freq x Power)

## IGainCompression History



Interface	Introduced with PNA Rev:
IGainCompression	8.0
IGainCompression2	8.2
IGainCompression3	9.0
IGainCompression4	9.2

---

Last Modified:

3-Sep-2009	Added GC3 (9.0)
10-Jun-2009	Added SearchSummary
12-May-2008	Added 8.2
11-Sep-2007	MX New topic

## Gain CompressionCal Object

---

### Description

Sets properties that are unique to a Gain Compression Cal (opt 086).

The remaining commands to perform a GCA Cal use the [Guided Calibration commands](#).

### Accessing the GainCompressionCal object

```
Dim app as AgilentPNA835x.Application
Set GCACal = pna.GetCalmanager.CreateCustomCalEx\(channelNum\)
Set GCACalExtension = GCACal.CustomCalConfiguration
GCACalExtension.PowerLevel = 5
```

### See Also:

- **Example Program** [Create and Cal a Gain Compression Measurement](#)
- [GainCompression Object](#)
- [About Gain Compression Application](#)
- [The PNA Object Model](#)
- [PNA Automation Interfaces](#)

Method	Interface	Description
<a href="#">See History</a>		
None		

Property	Interface	Description
<a href="#">PowerLevel</a>	IGainCompressionCal	Set and read the power level at which to perform the Source Power Cal.
<a href="#">PowerSensorCalKitType</a>	IGainCompressionCal2	Set and read the cal kit to be used for calibrating at the port 1 reference plane.
<a href="#">PowerSensorConnectorType</a>	IGainCompressionCal2	<b>Superseded</b> by <a href="#">PowerSensorConnectorType</a> on the GuidedCal Object.

**IGainCompressionCal History**

Interface	Introduced with PNA Rev:
IGainCompressionCal	8.0
IGainCompressionCal2	8.04

---

Last Modified:

- 12-Oct-2009    Superseded property (9.1)
- 11-Apr-2008    Added 8.04 properties
- 27-Nov-2007    MX New topic

## Gating Object

### Description

Contains the methods and properties that control Time Domain Gating.

### Accessing the Gating Object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim gate As Gating
Set gate = app.ActiveMeasurement.Gating
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Time Domain Topics](#)
- [Example Programs](#)

### Methods

None

Properties	Interface (see History)	Description
<a href="#">Center</a>	IGating	Sets or returns the Center time. Shared with the Transform Object
<a href="#">CoupledParameters</a>	<b>IGating2</b>	Select Gating parameters to couple
<a href="#">Shape</a>	IGating	Specifies the shape of the gate filter.
<a href="#">Span</a>	IGating	Sets or returns the Span time. Shared with the Transform Object
<a href="#">Start</a>	IGating	Sets or returns the Start time. Shared with the Transform Object
<a href="#">State</a>	IGating	Turns an Object ON and OFF.

<a href="#">Stop</a>	IGating	Sets or returns the Stop time. Shared with the Transform Object
<a href="#">Type</a>	IGating	Specifies the type of gate filter used.

### History

Interface	Introduced with PNA Rev:
IGating	1.0
IGating2	4.2

## GuidedCalibration Object

---

### Description

Contains the methods and properties used to perform a Guided Calibration.

A Guided Calibration must be performed on the Active Channel. To activate a channel, activate any measurement on that channel. Do this using [meas.Activate](#), which requires you already have a handle to the measurement.

**Note:** ECal orientation is performed using the [OrientECALModule\\_Property](#) and [ECALPortMapEx\\_Property](#) on the [Calibrator Object](#).

### Accessing the GuidedCalibration object

For standard S-parameter channels:

```
Dim app as AgilentPNA835x.Application
Dim CalMgr
Set CalMgr = App.GetCalManager
Dim guidedCal
Set guidedCal = CalMgr.GuidedCalibration
```

To calibrate an [Application](#) channel, see [CreateCustomCalEx](#) Method.

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods		Description
<a href="#">AcquireStep</a>	IGuidedCalibration	Acquire data for a cal step.
<a href="#">ApplyDeltaMatchFromCalSet Method</a>	IGuidedCalibration2	Apply a cal as Delta Match Cal.
<a href="#">GenerateErrorTerms</a>	IGuidedCalibration	Generates the error terms for the calibration.
<a href="#">GenerateGlobalDeltaMatchSequence</a>	IGuidedCalibration2	Initiates a global delta match calibration.
<a href="#">GenerateSteps</a>	IGuidedCalibration	Request to generate a connection list and return the number of steps required.
<a href="#">GetCompatibleCalKits</a>	IGuidedCalibration5	Returns the list of cal kits for the connector type.
<a href="#">GetIsolationPaths</a>	IGuidedCalibration3	Gets the list of port pairings for which isolation standards will be measured during calibration.

<a href="#">GetStepDescription</a>	IGuidedCalibration	Query description of a step.
<a href="#">Initialize</a>	IGuidedCalibration	Initial setup with channel context for the remote cal object.
<a href="#">SetIsolationPaths</a>	IGuidedCalibration3	Sets the list of port pairings for which isolation standards will be measured during calibration.
<a href="#">SetupMeasurementsForStep</a>	IGuidedCalibration4	Show the Cal Window, or custom Cal Window, before acquiring a Cal standard.

## Properties

## Interface

## Description

[See History](#)

<a href="#">CalKitType</a>	IGuidedCalibration	Sets the cal kit for the port.
<a href="#">CompatibleCalKits</a>	IGuidedCalibration	<b>Superseded</b> with <a href="#">GetCompatibleCalKits Method</a>
<a href="#">ConnectorType</a>	IGuidedCalibration	Sets the connector type for the port.
<a href="#">CustomCalConfiguration</a>	IGuidedCalibration4	Provides access to additional Properties and Methods which extends the GuidedCal Object.
<a href="#">IsolationAveragingIncrement</a>	IGuidedCalibration3	Value by which to increment the channel's averaging factor during measurement of isolation standards.
<a href="#">PathCalMethod</a>	IGuidedCalibration3	Specifies the calibration method for each port pair.
<a href="#">PathThruMethod</a>	IGuidedCalibration3	Specifies the calibration <b>THRU</b> method for each port pair.
<a href="#">PortsNeedingDeltaMatch</a>	IGuidedCalibration2	Returns port numbers that need delta match cal.
<a href="#">PowerCalibrationLevel</a>	IGuidedCalibration6	Sets power level for power cal in several applications.
<a href="#">PowerSensorCalKitType</a>	IGuidedCalibration6	Sets Cal Kit for power cal in several applications.
<a href="#">PowerSensorConnectorType</a>	IGuidedCalibration6	Sets Power sensor connector for power cal in several applications.
<a href="#">ThruCalMethod</a>	IGuidedCalibration	<b>Superseded with</b> <a href="#">PathCalMethod</a> and <a href="#">PathThruMethod</a>

---

<a href="#">ThruPortList</a>	IGuidedCalibration	Sets the thru connection port pairs.
<a href="#">UseCalWindow</a>	IGuidedCalibration	Turns Cal window ON or OFF
<a href="#">ValidConnectorTypes</a>	IGuidedCalibration	Gets Valid Connector Types.

#### IGuidedCalibration History

Interface	Introduced with PNA Rev:
IGuidedCalibration	5.0
IGuidedCalibration2	5.25
IGuidedCalibration3	7.11
IGuidedCalibration4	8.0
IGuidedCalibration5	9.0
IGuidedCalibration6	9.1

---



---

#### Last Modified:

25-Aug-2009	Added GetCompatibleCalKits
22-Jul-2009	Added CustomCalConfiguration
9-Nov-2007	Added Setup command and Activate note



## HWAuxIO Object

### Description

Contains the methods and properties that control the rear panel Auxiliary Input / Output connector.

**Note:** PNA-X models do NOT have this connector. However, the get/put Input/Output voltage commands can be used on the PNA-X to control ADC voltages on the Power I/O connector: Sending other Control:AUX commands to a PNA-X may result in unusual behavior.

### Accessing the HWAuxIO object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim AuxIO As HWAuxIO
Set AuxIO = app.GetAuxIO
```

### See Also:

- [Pinout of the Aux IO Connector](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">get_InputVoltage</a>	IHWAuxIO	<b>Superseded</b> by <a href="#">get InputVoltageEX</a>
<a href="#">get_OutputVoltage</a>	IHWAuxIO	Reads ADC output voltages.
<a href="#">get_OutputVoltageMode</a>	<b>IHWAuxIO2</b>	Reads mode setting for either DAC output.
<a href="#">get_PortCData</a>	IHWAuxIO	Reads a 4-bit value from Port C
<a href="#">put_OutputVoltage</a>	IHWAuxIO	Writes voltages to the DAC/Analog Output 1 and Output 2
<a href="#">put_OutputVoltageMode</a>	<b>IHWAuxIO2</b>	Writes the mode setting for either DAC output.
<a href="#">put_PortCData</a>	IHWAuxIO	Writes a 4-bit value to Port C
Properties	Description	

---

<a href="#">FootSwitch</a>	IHWAuxIO	Reads the Footswitch Input
<a href="#">FootswitchMode</a>	<b>IHWAuxIO3</b>	Determines the action that occurs when the footswitch is pressed.
<a href="#">InputVoltageEX</a>	<b>IHWAuxIO5</b>	Reads the ADC input voltage
<a href="#">PassFailLogic</a>	IHWAuxIO	Sets and reads the logic of the PassFail line Shared with the HWMaterialHandler Object
<a href="#">PassFailMode</a>	IHWAuxIO	Sets and reads the mode of the PassFail line Shared with the HWMaterialHandler Object
<a href="#">PassFailPolicy</a>	<b>IHWAuxIO4</b>	Sets the policy used to determine how global pass/fail is computed. Shared with the HWMaterialHandler Object
<a href="#">PassFailScope</a>	IHWAuxIO	Sets and reads the scope of the PassFail line Shared with the HWMaterialHandler Object
<a href="#">PassFailStatus</a>	<b>IHWAuxIO4</b>	Returns the most recent pass/fail status value. Shared with the HWMaterialHandler Object
<a href="#">PortCLogic</a>	HWAuxIO	Sets and reads the logic mode of Port C
<a href="#">PortCMode</a>	HWAuxIO	Sets and reads the mode of Port C
<a href="#">SweepEndMode</a>	HWAuxIO	Sets and reads the event that causes the Sweep End line to go to a false state. Shared with the HWMaterialHandler Object

#### **IHWAuxIO History**

Interface	Introduced with PNA Rev:
IHWAuxIO	2.0
IHWAuxIO2	3.0
IHWAuxIO3	3.0
IHWAuxIO4	5.0
IHWAuxIO5	7.5

Last Modified:

- 5-Aug-2008 Updated for InputVoltageEX command
- 10-Jul-2007 Added new command
- 29-Jun-2007 Updated for PNA-X ADC commands

## HWExternalTestSetIO Object

---

### Description

Contains the methods and properties that control the rear panel External Test Set Input / Output connector

### Accessing the HWExternalTestSetIO object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim ExtTS As HWExternalTestSetIO
Set ExtTS = app.GetExternalTestSetIO
```

### See Also:

- [Pinout of the Aux IO Connector](#)
- [Pinout for the External Test Set Connector](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">ReadData</a>	Reads data and generates the appropriate timing signals
<a href="#">ReadRaw</a>	Reads data, but does NOT generate appropriate timing signals
<a href="#">WriteData</a>	Writes data and generates the appropriate timing signals
<a href="#">WriteRaw</a>	Writes data, but does NOT generate the appropriate timing signals

Properties	Description
<a href="#">Interrupt</a>	Returns the state of the Interrupt line
<a href="#">SweepHoldOff</a>	Returns the state of the Sweep Holdoff line

### IHWExternalTestSetIO History

Interface	Introduced with PNA Rev:
-----------	--------------------------

HWExternalTestSetIO	2.0
---------------------	-----

## HWMaterialHandlerIO Object

### Description

Contains the methods and properties that control the rear panel Material Handler Input / Output connector.

### Accessing the HWMaterialHandlerIO object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim MatHdlr As HWMaterialHandlerIO
Set MatHdlr = app.GetMaterialHandlerIO
```

### See Also:

- [Pinout for the Material HandlerIO Connector](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface	Description
<a href="#">get_Input1</a>	HWMaterialHandlerIO	Reads a hardware latch that captures low to high transition on Input1
<a href="#">get_Output</a>	HWMaterialHandlerIO	Returns the last value written to the selected output pin.
<a href="#">get_Port</a>	HWMaterialHandlerIO	Returns the value from the specified "readable" port.
<a href="#">put_Output</a>	HWMaterialHandlerIO	Writes a TTL HI or TTL Low to output pins 3 or 4.
<a href="#">put_Port</a>	HWMaterialHandlerIO	Writes a value to the specified port.

Properties		Description
<a href="#">IndexState</a>	<b>HWMaterialHandlerIO2</b>	Determines the control of Material Handler connector Pin 20
<a href="#">ReadyForTriggerState</a>	<b>HWMaterialHandlerIO2</b>	Determines the control of Material Handler connector Pin 21

<a href="#"><u>PassFailLogic</u></a>	HWMaterialHandlerIO	Sets and reads the logic of the PassFail line Shared with the HWAuxIO Object
<a href="#"><u>PassFailMode</u></a>	HWMaterialHandlerIO	Sets and reads the mode for the PassFail line Shared with the HWAuxIO Object
<a href="#"><u>PassFailPolicy</u></a>	<b>HWMaterialHandlerIO2</b>	Sets the policy used to determine how global pass/fail is computed. Shared with the HWAuxIO Object
<a href="#"><u>PassFailScope</u></a>	HWMaterialHandlerIO	Sets and reads the scope for the PassFail line Shared with the HWAuxIO Object
<a href="#"><u>PassFailStatus</u></a>	<b>HWMaterialHandlerIO2</b>	Returns the most recent pass/fail status value. Shared with the HWAuxIO Object
<a href="#"><u>PortLogic</u></a>	HWMaterialHandlerIO	Sets and returns the logic mode of data ports A-H
<a href="#"><u>PortMode</u></a>	HWMaterialHandlerIO	Sets and returns whether Port C or Port D is used for writing or reading data
<a href="#"><u>SweepEndMode</u></a>	HWMaterialHandlerIO	Sets and reads the event that cause the Sweep End line to go to a low state. Shared with the HWAuxIO Object

#### **HWMaterialHandlerIO History**

<b>Interface</b>	<b>Introduced with PNA Rev:</b>
HWMaterialHandlerIO	2.0
HWMaterialHandlerIO2	5.0

## IFConfiguration Object

---

### Description

These properties control the IF gain and source path settings for the following:

- E836x Opt H11 - all [IFConfiguration](#) and [IFConfiguration2](#) commands
- PNA-X - [IFConfiguration3](#) commands ONLY

### Accessing the IFConfiguration object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim cfg as IIFConfiguration
Set cfg = chan.IFConfiguration
```

### See Also:

- [IF Path Configuration](#)
- [SignalProcessingModuleFour Object](#) (PNA-X ONLY)
- [PulseGenerator Object](#) (PNA-X ONLY)
- [The PNA Object Model](#)
- [Pulsed Application](#)
- [Pulsed Measurement Example](#)

Methods	Description
---------	-------------

None

Properties	Interface <a href="#">See History</a>	Description
------------	--	-------------

<a href="#">IFFilterSampleCount</a>	IFConfiguration2	Sets or returns the number of taps in the IF filter.
<a href="#">IFFilterSamplePeriod</a>	IFConfiguration2	Sets or returns the IF filter sample period time.
<a href="#">IFFilterSamplePeriodList</a>	IFConfiguration2	Returns the list of available IF filter sample periods for the instrument.



<a href="#"><u>IFFilterSamplePeriodMode</u></a>	IFConfiguration2	Sets or returns the IF filter sample period mode.(Auto or Manual).
<a href="#"><u>IFFilterSource</u></a>	IFConfiguration2	Sets or retrieves type of IF filter to be used.
<a href="#"><u>IFFrequency</u></a>	IFConfiguration3	Sets IF frequency in manual mode.
<a href="#"><u>IFFrequencyMode</u></a>	IFConfiguration3	Sets IF frequency mode to automatic or manual.
<a href="#"><u>IFGainLevel</u></a>	IFConfiguration	Sets the gain level for the specified receiver.
<a href="#"><u>IFGainMode</u></a>	IFConfiguration	Sets the gain state for ALL receivers.
<a href="#"><u>IFGateEnable</u></a>	IFConfiguration2	Sets or retrieves the state of the IF Gate.
<a href="#"><u>IFSourcePath</u></a>	IFConfiguration	Sets the source path of the specified receiver to Internal or External.
<a href="#"><u>MaximumIFFilterSampleCount</u></a>	IFConfiguration2	Returns the maximum allowed value for the IFFilterSampleCount.
<a href="#"><u>MinimumIFFilterSampleCount</u></a>	IFConfiguration2	Returns the minimum allowed value for the IFFilterSampleCount.
<a href="#"><u>MaximumIFFrequency</u></a>	IFConfiguration3	Returns the maximum IF frequency setting
<a href="#"><u>MinimumIFFrequency</u></a>	IFConfiguration3	Returns the minimum IF frequency setting

### IFConfiguration History

Interface	Introduced with PNA Rev:
IIFConfiguration	4.0
IIFConfiguration2	4.0
IIFConfiguration3	7.2

## IMixer Interface (Option 083 )

---

### Description

Contains the methods and properties to setup FCA Mixer measurements.

The Converter Object can also be used to configure FCA measurements.

For performing calibrations, use either the SMC Type Object or the VMC Type Object .

### Accessing the IMixer Interface

Access the IMixer Interface through the Measurement Object. If the particular type of Measurement that was created supports IMixer, then the program determines this at run time and can access the functionality exposed by IMixer. Because the determination of IMixer support is not made until runtime, the program should handle the case where IMixer is not supported on the object.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", "analyzerName ")
app.Preset

' FCA Measurements can't share the channel with standard measurements
' Because preset creates a single measurement in channel 1, we first delete the
standard measurement
Dim standardMeas As IMeasurement
Set standardMeas = app.ActiveMeasurement
standardMeas.Delete

' Create a Measurement object, in this case using the IMeasurement interface
Dim meas As IMeasurement
Set meas = app.CreateCustomMeasurementEx(1, "SMC_Forward.SMC_ForwardMeas", "SC21")

' See if this measurement object supports IMixer
Dim mixer As IMixer
Set mixer = meas
```

### See Also:

PNA Automation Interfaces

The PNA Object Model

Example: How to create and calibrate a standard SMC or VMC measurement or a fixed output SMC measurement.

Methods	Interface	Description
	See History	
Apply	IMixer3	Applies mixer settings.
Calculate	IMixer	Automatically calculate Input and Output frequencies for mixer setup.

LoadFile	IMixer	Loads a previously-configured mixer attributes file (.mxr )
SaveFile	IMixer	Saves the settings for the mixer/converter test setup to a mixer attributes file
SetDutPorts Method	<b>IMixer8</b>	Sets the PNA to DUT port mapping.

## Properties

## Description

ActiveXAxisRange	<b>IMixer3</b>	Sets or returns the swept parameter to display on the X-axis.
AvoidSpurs	IMixer	Sets and returns the state of the avoid spurs feature.
<b>EmbeddedLO</b>	<b>IMixer7</b>	Provides measurements of mixers with an embedded LO.
DeviceInputPort	<b>IMixer8</b>	Reads the PNA that is mapped to the DUT Input port
DeviceOutputPort	<b>IMixer8</b>	Reads the PNA that is mapped to the DUT Output port
EnablePhase	<b>IMixer12</b>	Include phase in SMC measurements and calibrations.
IFSideband	IMixer	Sets or returns the value of the IF sideband.
IFStartFrequency	IMixer	Returns the start frequency of the mixer IF.
IFStopFrequency	IMixer	Returns the stop frequency of the mixer IF.
IncludeReverseSweep	<b>IMixer12</b>	Sets or returns whether to include SC12 sweep.
InputDenominator	IMixer	Sets or returns the denominator value of the Input Fractional Multiplier.
InputFixedFrequency	<b>IMixer6</b>	Sets or returns the mixer fixed Input frequency value.
InputNumerator	IMixer	Sets or returns the numerator value of the Input Fractional Multiplier.
InputPower	IMixer	Sets or returns the value of the Input Power.
InputRangeMode	<b>IMixer6</b>	Sets or returns the Input sweep mode.
InputStartFrequency	IMixer	Sets or returns the start frequency of the mixer input.
InputStopFrequency	IMixer	Sets or returns the stop frequency of the mixer input.
IsInputGreater ThanLO	<b>IMixer2</b>	Specifies whether to use the Input frequency that is greater than the LO or LO
LODenominator	IMixer	Sets or returns the denominator value of the LO Fractional Multiplier.
LOFixedFrequency	IMixer	Sets or returns the fixed frequency of the specified LO.
LOName	IMixer	Sets or returns the LO name.

LONumerator	IMixer	Sets or returns the numerator value of the LO Fractional Multiplier.
LOPower	IMixer	Sets or returns the value of the LO Power.
LORangeMode	<b>IMixer3</b>	Sets or returns the LO sweep mode to fixed or swept.
LOStage	IMixer	Returns the number of LOs (1 or 2).
LOStartFrequency	<b>IMixer3</b>	Sets or returns the start frequency of the specified LO.
LOStopFrequency	<b>IMixer3</b>	Sets or returns the start frequency of the specified LO.
NominalIncidentPowerState	<b>IMixer4</b>	Toggles Nominal Incident Power ON and OFF.
NormalizePoint	IMixer12	Set or return the data point used to normalize SMC phase measurements.
OutputFixedFrequency	<b>IMixer3</b>	Sets or returns the fixed frequency of the mixer output.
OutputRangeMode	<b>IMixer6</b>	Sets or returns the Output sweep mode.
OutputSideband	IMixer	Sets or returns the value of the output sideband.
OutputStartFrequency	IMixer	Sets or returns the start frequency of the mixer output.
OutputStopFrequency	IMixer	Sets or returns the stop frequency of the mixer output.

### IMixer History

Interface	Introduced with PNA Rev:
IMixer	1.0
IMixer2	3.5
IMixer3	4.0
IMixer4	4.8
IMixer5	6.04
IMixer6	6.20
IMixer7	7.21
IMixer8	7.5/8.2
IMixer12	7.5/9.2



# IMSpectrum Object

## Description

Controls the IM Spectrum settings.

## Accessing the IMSpectrum object

```
Dim app as AgilentPNA835x.Application
app.CreateCustomMeasurementEx(1, "IMSpectrum", "Output", 1)
Dim IMSpec
Set IMSpec = app.ActiveChannel.CustomChannelConfiguration
```

## See Also:

- **Example Program** Create and Cal a IM Spectrum Measurement
- [About IM Spectrum Measurements](#)
- [SweptIMD Object](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Method	Interface	Description
	<a href="#">See History</a>	

<a href="#">SetPortMap</a>	IIMSpectrum	Sets limited port mapping
----------------------------	-------------	---------------------------

Property	Interface	Description
<a href="#">CoupleTonePower</a>	IIMSpectrum	ON   OFF state of power coupling for F1 and F2.
<a href="#">DeviceInputPort</a>	IIMSpectrum	Read input port map
<a href="#">DeviceOutputPort</a>	IIMSpectrum	Read output port map
<a href="#">DeltaFrequency</a>	IIMSpectrum	Fixed tone spacing value.
<a href="#">F1Frequency</a>	IIMSpectrum	Frequency of the F1 tone.

<a href="#">F2Frequency</a>	IIMSpectrum	Frequency of the F2 tone.
<a href="#">FrequencyCenter</a>	IIMSpectrum	Center frequency of the main tones.
<a href="#">ResolutionBW</a>	IIMSpectrum	Resolution Bandwidth for the measurement.
<a href="#">SpectrumCenterFrequency</a>	IIMSpectrum	Receiver Center frequency
<a href="#">SpectrumSpanFrequency</a>	IIMSpectrum	Receiver frequency span.
<a href="#">SpectrumStartFrequency</a>	IIMSpectrum	Receiver Start frequency.
<a href="#">SpectrumStopFrequency</a>	IIMSpectrum	Receiver Stop frequency.
<a href="#">TrackingChannel</a>	IIMSpectrum	IMD channel number to which the IM Spectrum channel is coupled.
<a href="#">TrackingEnable</a>	IIMSpectrum	Enables tracking with an IMD channel.
<a href="#">TrackingManualStepEnable</a>	IIMSpectrum	Step sweep mode for the IM Spectrum channel.
<a href="#">TrackingStepIndex</a>	IIMSpectrum	IMD data point number at which the IM spectrum measurement occurs.
<a href="#">SweepOrder</a>	IIMSpectrum	IM product to view when <a href="#">SweepType</a> = NTH is specified.
<a href="#">SweepType</a>	IIMSpectrum	Type of sweep for an IIMSpectrum measurement.
<a href="#">TonePower</a>	IIMSpectrum	Power level of the Main Tones

### IIMSpectrum History

Interface	Introduced with PNA Rev:
IIMSpectrum	8.35

Last Modified:

11-Aug-2009    Added limited port mapping  
19-Aug-2008    MX New topic

# InterfaceControl Object

## Description

Contains the methods and properties that support Interface Control.

## Accessing the InterfaceControl object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim IntControl As InterfaceControl
Set IntControl = app.InterfaceControl
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Interface Control Feature](#)
- [Example Programs](#)

Methods	Interface	Description
<a href="#">ConfigurationFile</a>	InterfaceControl	Recalls an Interface Control file

Properties	Description
<a href="#">State</a>	InterfaceControl Turns Interface Control ON and OFF

## InterfaceControl History

Interface	Introduced with PNA Rev:
InterfaceControl	5.2



## Limit Test Collection

---

### Description

Child of the **Measurement** Object. A collection that provides a mechanism for iterating through the Measurement's Limit Segment objects (Limit Lines). The collection has 100 limit lines by default.

### Accessing the LimitTest collection

Get a handle to an individual limit segment by specifying an item of the LimitTest collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim limSegs As LimitTest
Set limSegs = app.ActiveMeasurement.LimitTest
limSegs.Item(1).BeginResponse = 1000000000#
```

### See Also:

- [LimitSegment Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Limit Line Testing Example](#)

Methods	Description
<a href="#">GetTestResult</a>	Retrieves the Pass/Fail results of the Limit Test (State).
<a href="#">Item</a>	Use to get a handle on a limit line in the collection.
Properties	Description
<a href="#">Count</a>	Returns the number of limit lines used in the measurement.
<a href="#">LineDisplay</a>	Displays the limit lines on the screen.
<a href="#">SoundOnFail</a>	Enables a beep on Limit Test fails.
<a href="#">State</a>	Turns ON and OFF limit testing.

### LimitTest History

Interface	Introduced with PNA Rev:
-----------	--------------------------

ILimitTest	1.0
------------	-----

## LimitSegment Object

### Description

The LimitSegment object is an individual limit line.

### Accessing the LimitSegment object

Get a handle to an individual limit line by using the LimitTest collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim limSegs As LimitTest
Set limSegs = app.ActiveMeasurement.LimitTest
limSegs(1).BeginResponse = 1000000000#
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
---------	-------------

None

Properties	Description
<a href="#">BeginResponse</a>	Specifies the Y-axis value that corresponds with Begin Stimulus (X-axis) value.
<a href="#">BeginStimulus</a>	Specifies the beginning X-axis value of the Limit Line.
<a href="#">EndResponse</a>	Specifies the Y-axis value that corresponds with End Stimulus (X-axis) value.
<a href="#">EndStimulus</a>	Specifies the End X-axis value of the Limit Line.
<a href="#">Type</a>	Specifies the Limit Line type.

### LimitSegment History

Interface	Introduced with PNA Rev:
ILimitSegment	1.0



# Marker Object

## Description

Contains the methods and properties that control Markers. There are 10 markers available per measurement:

- 1 reference marker
- 9 markers for absolute data or data relative to the reference marker (delta markers).

There are two ways to control markers through COM.

1. The [Measurement object](#) has properties that apply to ALL of the markers for that measurement. For example, **meas.MarkerFormat = naLinMag** applies formatting to all markers.
2. Marker object properties override the Measurement object properties. For example, you can then override the format setting for an individual marker by specifying **mark.Format = naLogMag** on the marker object.

Note: [SearchFilterBandwidth](#) is available through the [measurement object](#).

## Accessing the Marker object

To turn ON a marker, get a handle to the marker through the [measurement object](#). If not already activated, this command will turn ON marker 1

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

app.ActiveMeasurement.marker(1).Format = naLinMag
```

You can also set the marker object to an object variable:

```
Dim m1 As Marker
Set m1 = app.ActiveMeasurement.Marker(1)
m1.Format = naMarkerFormat_LinMag
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)
- [All about Markers](#)

Methods	Interface	Description
<a href="#">Activate</a>	IMarker	Makes an object the Active Object. Shared with the Marker Object

<a href="#">SearchCompressionPoint</a>	IMarker4	Searches the marker domain for the compression level.
<a href="#">SearchMax</a>	IMarker	Searches the marker domain for the maximum value.
<a href="#">SearchMin</a>	IMarker	Searches the marker domain for the minimum value.
<a href="#">SearchNextPeak</a>	IMarker	Searches the marker's domain for the next largest peak value.
<a href="#">SearchPeakLeft</a>	IMarker	Searches the marker's domain for the next VALID peak to the left of the marker.
<a href="#">SearchPeakRight</a>	IMarker	Searches the marker's domain for the next VALID peak to the right of the marker.
<a href="#">SearchTarget</a>	IMarker	Searches the marker's domain for the target value.
<a href="#">SearchTargetLeft</a>	IMarker	Moving to the left of the marker position, searches the marker's domain for the target value.
<a href="#">SearchTargetRight</a>	IMarker	Moving to the right of the marker position, searches the marker's domain for the target value.
<a href="#">SetCenter</a>	IMarker	Changes the analyzer's center frequency to the X-axis position of the marker.
<a href="#">SetCW</a>	IMarker	Changes the sweep type to CW mode and makes the CW frequency the marker's frequency.
<a href="#">SetCWFreq</a>	IMarker3	Changes the CW frequency to the frequency of the active marker.
<a href="#">SetElectricalDelay</a>	IMarker	Changes the measurement's electrical delay to the marker's delay value.
<a href="#">SetReferenceLevel</a>	IMarker	Changes the measurement's reference level to the marker's Y-axis value.
<a href="#">SetStart</a>	IMarker	Changes the analyzer's start frequency to the X-axis position of the marker.
<a href="#">SetStop</a>	IMarker	Changes the analyzer's stop frequency to the X-axis position of the marker.

Properties	Description
------------	-------------

<a href="#">Bucket Number</a>	IMarker	Marker data point number
<a href="#">CompressionLevel</a>	IMarker4	Set and read the marker compression level.
<a href="#">CompressionPin</a>	IMarker4	Reads the input power at the marker compression level.

<a href="#">CompressionPout</a>	IMarker4	Reads the output power at the marker compression level.
<a href="#">DeltaMarker</a>	IMarker	Makes a marker relative to the reference marker
<a href="#">Distance</a>	<b>IMarker2</b>	Sets or returns distance value for time domain trace.
<a href="#">Format</a>	IMarker	Linear, SWR, and so forth
<a href="#">Interpolated</a>	IMarker	Turn marker interpolation ON and OFF
<a href="#">Number</a>	IMarker	Read the number of the active marker
<a href="#">PeakExcursion</a>	IMarker	Sets and reads the peak excursion value for the specified marker.
<a href="#">PeakThreshold</a>	IMarker	Sets peak threshold for the specified marker.
<a href="#">SearchFunction</a>	IMarker	Emulates the Tracking function in the marker search dialog box.
<a href="#">Stimulus</a>	IMarker	Sets and reads the X-Axis value of the marker.
<a href="#">Target Value</a>	IMarker	Sets the target value for the marker when doing Target Searches.
<a href="#">Tracking</a>	IMarker	The tracking function finds the selected search function every sweep.
<a href="#">Type</a>	IMarker	Sets and reads the marker type.
<a href="#">UserRange</a>	IMarker	Assigns the marker to the specified User Range.
<a href="#">UserRangeMax</a>	IMarker	Sets the stimulus stop value for the specified User Range.
<a href="#">UserRangeMin</a>	IMarker	Sets the stimulus start value for the specified User Range.
<a href="#">Value</a>	IMarker	Reads the Y-Axis value of the marker.

### Marker History

Interface	Introduced with PNA Rev:
IMarker	1.0
IMarker2	4.2
IMarker3	8.33
IMarker4	8.50

## Measurement Object

---

See [IArrayTransfer Interface](#) for putting and getting typed data.

See [IMixer Interface](#) (used with [Option 083](#))

### Description

The Measurement object is probably the most used object in the PNA Object Model. A measurement object represents the chain of data processing algorithms that take raw data from the channel and make it ready for display, which then becomes the scope of the [Trace object](#).

A Measurement object is defined by its parameter (S11, S22, A/R1, B and so forth). The measurement object is associated with a [channel](#) which drives the hardware that produces the data that feeds the measurement. The root of a measurement is the raw data. This buffer of complex paired data then flows through a number of processing blocks: error-correction, trace math, phase correction, time domain, gating, formatting. All of these are controlled through the measurement object.

The ACTIVE measurement is the measurement that will be acted upon if you make a setting from the front panel. It is the measurement whose "button" is pressed in the window with the red "active window" frame. If you create a new measurement, that measurement becomes the active measurement.

Therefore, all automation methods with the word "Active" in them refer to the object associated with the Active measurement, whether that object is a Channel, Window, Trace or Limit line.

Learn about the [IMeasurement2 Interface](#) for reading stimulus properties.

### Accessing the Measurement object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim meas As IMeasurement
Set meas = app.ActiveMeasurement
```

or

```
Set meas = app.Measurements(n)
```

You can access four other objects through the Measurement object: markers, limit test, transform, and gating. For example, because each measurement has its own set of markers, you can set a marker by doing this:

```
Dim meas as measurement
Set meas = app.ActiveMeasurement
meas.marker(1).Stimulus = 900e6
```

### IMeasurement2 Interface

Some of the properties and methods for the IMeasurement2 Interface return stimulus values that are set using the channel object. The following is the reason these properties and methods are duplicated.

Every measurement carries with it a snapshot of the stimulus properties of the channel that were in effect when the measurement last acquired data. Therefore, it is the measurement that provides the most accurate stimulus description of its data. Any change made to the channel after the measurement was acquired renders the IChannel interface unreliable in terms of describing the measurement.

**See Also:**



- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)
- [Superseded commands](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Interface	Description
<a href="#">Activate</a>	IMeasurement	Makes an object the Active Object. Shared with the Marker Object
<a href="#">ActivateMarker</a>	IMeasurement	Makes a marker the Active Marker.
<a href="#">ChangeParameter</a>	IMeasurement	Changes the parameter of the measurement.
<a href="#">DataToDivisor</a>	IMeasurement	<b>Superseded with</b> <a href="#">DoReceiverPowerCal Method</a>
<a href="#">DataToMemory</a>	IMeasurement	Stores the active measurement into memory.
<a href="#">Delete</a>	IMeasurement	Deletes the measurement object.
<a href="#">DeleteAllMarkers</a>	IMeasurement	Deletes all of the markers from the measurement.
<a href="#">DeleteMarker</a>	IMeasurement	Deletes a marker from the active measurement.
<a href="#">getData</a>	IMeasurement	Retrieves Complex data from analyzer memory
<a href="#">getDataByString</a>	IMeasurement	Retrieves variant data from the specified location in your choice of formats.
<a href="#">GetFilterStatistics</a>	IMeasurement	Returns all four Filter Statistics
<a href="#">GetReferenceMarker</a>	IMeasurement	Returns a handle to the reference marker.
<a href="#">Get SnPData</a>	IMeasurement3	Returns SnP data.
<a href="#">GetSnpDataWithSpecifiedPorts</a>	IMeasurement7	Returns sNp data for the specified ports.
<a href="#">GetTraceStatistics</a>	IMeasurement	Returns the Trace Statistics.
<a href="#">GetXAxisValues</a>	IMeasurement2	Returns the stimulus values for the measurement.
<a href="#">InterpolateMarkers</a>	IMeasurement	Turns All Marker Interpolation ON and OFF for the measurement.
<a href="#">putDataComplex</a>	IMeasurement	Puts complex data into one of five data buffers.
<a href="#">putDataScalar</a>	IMeasurement	Puts formatted variant data into the measurement results buffer.

[SearchFilterBandwidth](#) IMeasurement Searches the domain with the current BW target.

[WriteSnPFileWithSpecifiedPorts](#) IMeasurement7 Write sNp data for specified ports to a file.

Properties	Interface	Description
<a href="#">ActiveMarker</a>	IMeasurement	Returns a handle to the Active Marker object.
<a href="#">BalancedMeasurement</a>	IMeasurement	Sets the measurement type that is used with balanced topologies.
<a href="#">BandwidthTarget</a>	IMeasurement	The insertion loss value at which the bandwidth of a filter is measured.
<a href="#">BandwidthTracking</a>	IMeasurement	Turns Bandwidth Tracking function ON and OFF.
<a href="#">CalibrationName</a>	IMeasurement2	Returns the name of the cal type.
<a href="#">CalibrationType</a>	IMeasurement	<b>Superseded with</b> <a href="#">CalibrationTypeID_property</a>
<a href="#">CalibrationTypeID</a>	IMeasurement2	Sets or returns the cal type for the current measurement.
<a href="#">Center</a>	IMeasurement2	Returns the stimulus value of the center point for the measurement.
<a href="#">channelNumber</a>	IIMeasurement	Returns the channel number. Shared with the Channel Object
<a href="#">CustomMeasurementConfiguration</a>	IIMeasurement12	Provides access to custom measurement properties and methods.
<a href="#">Domain</a>	IMeasurement2	Returns the domain (frequency, time, power) for the measurement.
<a href="#">ElectricalDelay</a>	IMeasurement	Sets electrical delay.
<a href="#">ElecDelayMedium</a>	IMeasurement2	Sets or returns the characteristic of the electrical delay medium.
<a href="#">ElecDistanceDelay</a>	IMeasurement11	Sets delay in distance
<a href="#">ElecDistanceDelayUnit</a>	IMeasurement11	Sets distance units
<a href="#">Equation</a>	IMeasurement6	Access Equation Editor
<a href="#">ErrorCorrection</a>	IMeasurement	Set or get the state of error correction for the measurement.
<a href="#">FilterBW</a>	IMeasurement	Returns the results of the SearchBandwidth method.

<a href="#">FilterCF</a>	IMeasurement	Returns the Center Frequency result of the SearchBandwidth method.
<a href="#">FilterLoss</a>	IMeasurement	Returns the Loss value of the SearchBandwidth method.
<a href="#">FilterQ</a>	IMeasurement	Returns the Q (quality factor) result of the SearchBandwidth method.
<a href="#">Format</a>	IMeasurement	Sets display format.
<a href="#">FormatUnit</a>	IMeasurement9	Sets units for unratioed measurements.
<a href="#">Gating</a>	IMeasurement	Controls Time Domain Gating.
<a href="#">GroupDelayAperture</a>	IMeasurement13	Provides access to the Group Delay Aperture settings.
<a href="#">InterpolateCorrection</a>	IMeasurement	Turns ON and OFF the calculation of new error terms when stimulus values change.
<a href="#">InterpolateNormalization</a>	IMeasurement	<b>Superseded with</b> <a href="#">DoReceiverPowerCal Method</a>
<a href="#">IsSparameter</a>	IMeasurement2	Returns true if measurement represents an S-Parameter.
<a href="#">LimitTest</a>	IMeasurement	Collection for iterating through the Limit Segment objects (Limit Lines).
<a href="#">LimitTestFailed</a>	IMeasurement	Returns the results of limit testing
<a href="#">LoadPort</a>	IMeasurement	Returns the load port number associated with an S-parameter reflection measurement.
<a href="#">LogMagnitudeOffset</a>	IMeasurement	<b>Superseded with</b> <a href="#">DoReceiverPowerCal Method</a>
<a href="#">MagnitudeOffset</a>	IMeasurment4	Offsets the magnitude of the entire data trace to a specified value.
<a href="#">MagnitudeSlopeOffset</a>	IMeasurment4	Offsets the magnitude of the data trace to a value that changes linearly with frequency.
<a href="#">Marker</a>	IMeasurement	Provides access to Marker settings.
<a href="#">MarkerFormat</a>	IMeasurement	Sets or returns the format of all the markers in the measurement.
<a href="#">Marker State</a>	IMeasurement3	Sets or returns the ON / OFF state of a marker.
<a href="#">Mean</a>	IMeasurement	Returns the mean value of the measurement.

<a href="#">Name</a>	IMeasurement	Sets or returns the name of the measurement.
<a href="#">NAWindow</a>	IMeasurement	Controls the part of the display that contains the graticule, or what is written on the display.
<a href="#">Normalization</a>	IMeasurement	<b>Superseded with</b> <a href="#">DoReceiverPowerCal Method</a>
<a href="#">Number</a>	IMeasurement	Returns the number of the measurement.
<a href="#">NumberOfPoints</a>	IMeasurement2	Returns the Number of Points of the measurement.
<a href="#">Parameter</a>	IMeasurement	Returns the measurement Parameter.
<a href="#">PeakToPeak</a>	IMeasurement	Returns the Peak to Peak value of the measurement.
<a href="#">PhaseOffset</a>	IMeasurement	Sets the Phase Offset for the active channel.
<a href="#">PSaturation</a>	IMeasurement13	Provides access to the Power Saturation marker search object.
<a href="#">ReceivePort</a>	IMeasurement2	Returns the receiver port of the measurement.
<a href="#">ReferenceMarkerState</a>	IMeasurement	Turns the reference marker ON or OFF
<a href="#">ShowStatistics</a>	IMeasurement	Displays and hides the measurement statistics (peak-to-peak, mean, standard deviation) on the screen.
<a href="#">Smoothing</a>	IMeasurement	Turns ON and OFF data smoothing.
<a href="#">SmoothingAperture</a>	IMeasurement	Specifies or returns the amount of smoothing as a ratio of the number of data points in the measurement trace.
<a href="#">SourcePort</a>	IMeasurement2	Returns the source port of the measurement.
<a href="#">Span</a>	IMeasurement2	Returns the stimulus span (stop - start) for the measurement.
<a href="#">StandardDeviation</a>	IMeasurement	Returns the standard deviation of the measurement.
<a href="#">Start</a>	IMeasurement2	Returns the stimulus value of the first point for the measurement.
<a href="#">StatisticsRange</a>	IMeasurement	Sets the User Range number for calculating measurement statistics.
<a href="#">Stop</a>	IMeasurement2	Returns the stimulus value of the last point for the measurement.
<a href="#">Trace</a>	IMeasurement	Controls scale, reference position, and reference line.

<a href="#">TraceMath</a>	IMeasurement	Performs math operations on the measurement object and the trace stored in memory.
<a href="#">TraceMax</a>	IMeasurement10	Maximizes the active trace.
<a href="#">TraceTitle</a>	IMeasurement8	Writes and reads a trace title.
<a href="#">TraceTitleState</a>	IMeasurement8	Turns trace title ON and OFF
<a href="#">Transform</a>	IMeasurement	Controls Time Domain transforms.
<a href="#">View</a>	IMeasurement	Sets (or returns) the type of trace displayed on the screen
<a href="#">WGCutoffFreq</a>	IMeasurement2	Sets or returns the value of the waveguide cut off frequency

### IMeasurement History

Interface	Introduced with PNA Rev:
IMeasurement	1.0
IMeasurement2	3.0
IMeasurement3	4.0
IMeasurement4	4.2
IMeasurement5	5.0
IMeasurement7	6.2
IMeasurement8	7.2
IMeasurement9	8.35
IMeasurement10	8.35
IMeasurement11	8.50
IMeasurement12	9.00
IMeasurement13	9.20

### IArrayTransfer Interface

## Description

Contains methods for putting data in and getting data out of the analyzer using typed data. This interface transfers data more efficiently than the IMeasurement Interface. However, this interfaces is only usable from VB6, C, & C++.

Methods	Description
<a href="#">getComplex</a>	Retrieves real and imaginary data from the specified buffer.
<a href="#">getNAComplex</a>	Retrieves typed <b>NAComplex</b> data from the specified buffer.
<a href="#">getPairedData</a>	Retrieves magnitude and phase data pairs from the specified buffer.
<a href="#">getScalar</a>	Retrieves scalar data from the specified buffer.
<a href="#">putComplex</a>	Puts real and imaginary data into the specified buffer.
<a href="#">putNAComplex</a>	Puts typed <b>NAComplex</b> data into the specified buffer.
<a href="#">putScalar</a>	Puts scalar data into the measurement result buffer.

Properties	Description
------------	-------------

None

## IArrayTransfer History

Interface	Introduced with PNA Rev:
IArrayTransfer	1.0

## Measurement Collection

---

### Description

A collection object that provides a mechanism for iterating through the Application measurements.

### Accessing the Measurements collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim measments As Measurements
Set measments = app.Measurements
```

### See Also:

- [Measurement Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Add</a>	Adds a Measurement to the collection.
<a href="#">Item</a>	Use to get a handle on a measurement in the collection.
<a href="#">Remove</a>	Removes a measurement from the measurements collection.
Properties	Description
<a href="#">Count</a>	Returns the number of measurements in the analyzer.
<a href="#">Parent</a>	Returns a handle to the current Application.

## NAWindow Object

---

### Description

The NAWindow object controls the part of the display that contains the graticule, or what is written on the display.

### Accessing the NaWindow object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim window As NAWindow
Set window = app.NAWindows(1)
window.AutoScale
```

or

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", "analyzerName")

app.NAWindows(1).AutoScale
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Description
<a href="#">Autoscale</a>	Autoscales all measurements in the window. Shared with the Trace Object
<a href="#">ShowMarkerReadout</a>	Shows and Hides the Marker readout for the active marker in the upper-right corner of the window object.
<a href="#">ShowTable</a>	Shows or Hides the specified table for the active measurement in the lower part of the window object.
Properties	Description
<a href="#">ActiveTrace</a>	Sets a trace to the Active Trace.
<a href="#">MarkerReadout</a>	Sets and reads the state of the Marker readout for the active marker in the upper-right corner of the window object.



<a href="#"><u>MarkerReadoutSize</u></a>	Specifies the size of font used when displaying Marker readout in the selected window.
<a href="#"><u>OneMarkerReadoutPerTrace</u></a>	Either show marker readout of only the active trace or all of the traces simultaneously.
<a href="#"><u>ScaleCouplingMethod</u></a>	Sets and returns the method of scale coupling.
<a href="#"><u>ScaleCouplingState</u></a>	Enables and disables scale coupling for the specified window.
<a href="#"><u>Title</u></a>	Writes or reads a custom title for the window.
<a href="#"><u>TitleState</u></a>	Turns ON and OFF the window title.
<a href="#"><b><u>Traces</u></b></a>	Collection for getting a handle to a trace or iterating through the traces in a window.
<a href="#"><u>WindowNumber</u></a>	Reads the number of the active window.
<a href="#"><u>WindowState</u></a>	Maximizes or minimizes a window. Shared with the Application Object

#### **INaWindow History**

<b>Interface</b>	<b>Introduced with PNA Rev:</b>
INaWindow	1.0
INaWindow2	9.0

## NAWindows Collection

---

### Description

A collection object that provides a mechanism for iterating through the Application windows.

### Accessing the NAWindows collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim windows As NAWindows
Set windows = app.NAWindows
```

### See Also:

- [NAWindow Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Add</a>	Adds a window to the NAWindows collection.
<a href="#">Item</a>	Use to get a handle to a window in the collection.
<a href="#">Remove</a>	Removes a window from the NAWindows collection.
Properties	Description
<a href="#">Count</a>	Returns the number of windows on the analyzer.
<a href="#">Parent</a>	Returns a handle to the current Application.

# NoiseCal Object

## Description

Controls the noise figure calibration settings for amplifiers and converters. These commands supplement the standard calibration commands on the [GuidedCalibration Object](#).

## Accessing the NoiseCal object

```
Dim app as AgilentPNA835x.Application
Set noisecal = pna.GetCalmanager.CreateCustomCalEx\(channelNum\)
Set noiseCalExtension = noisecal.CustomCalConfiguration
noiseCalExtension.NoiseSourceCold = 300
```

## See Also

- **Examples:**
  - [Create and Cal a Noise Figure Measurement](#)
  - [Create and Cal an NFX Measurement](#)
- [NoiseFigure Object](#)
- [About Noise Figure Measurements](#)
- [About NFX Measurements](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Method	Interface	Description
	<a href="#">See History</a>	

None

Property	Interface	Description
<a href="#">AutoOrientTuner</a>	INoiseCal2	Sets the state of auto orientation for a noise tuner during Noise Figure for NFX.
<a href="#">CalMethod</a>	INoiseCal	Sets and returns the method for performing calibration on a noise channel.

<a href="#">EnableLOPowerCal</a>	INoiseCal2	Enables and disables LO power calibration for NFX.
<a href="#">ENRFile</a>	INoiseCal	Sets and returns the name of the ENR file associated with the noise source.
<a href="#">ForceDeEmbedENRAdapter</a>	INoiseCal2	Sets and reads the state of ENR adapter de-embedding.
<a href="#">ForceDeEmbedSensorAdapter</a>	INoiseCal2	Sets and reads the state of noise source adapter de-embedding.
<a href="#">NoiseSourceCalKitType</a>	INoiseCal	Sets and reads the Cal Kit type used to perform a cal at the adapter which is used to connect the noise source (if required.)
<a href="#">NoiseSourceCold</a>	INoiseCal	Sets and returns the current temperature at the noise source.
<a href="#">NoiseSourceConnectorType</a>	INoiseCal	Sets and reads the connector type of the noise source used during the cal.

#### NoiseConfiguration History

Interface	Introduced with PNA Rev:
INoiseCal	8.0
INoiseCal2	9.1

Last Modified:

30-May-2007    MX New topic

## NoiseFigure Object

---

### Description

Controls the Noise Figure application settings.

### Accessing the NoiseFigure object

```
Dim app as AgilentPNA835x.Application
app.CreateCustomMeasurementEx(1, "NoiseFigure", "NF", 1)
Dim NoiseFig
Set NoiseFig = app.ActiveChannel.CustomChannelConfiguration
```

### See Also:

- Example programs
  - [Create and Cal a NoiseFigure Measurement](#)
  - [Create and Cal an NFX Measurement](#)
- [About Noise Figure Measurements](#)
- [Noise Figure Calibration Object](#)
- [app.NoiseSourceState \(ON and OFF\)](#)
- [ENRFile Object](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Method	Interface	Description
<a href="#">See History</a>		
<a href="#">SetPortMap</a>	<b>INoiseFigure6</b>	Maps DUT ports to PNA-X ports (Opt 028)

Property	Interface	Description
<a href="#">AmbientTemperature</a>	INoiseFigure	Sets the air temperature at which the measurement is being performed.
<a href="#">DeviceInputPort</a>	<b>INoiseFigure6</b>	Read DUT input port map

<a href="#"><u>DeviceOutputPort</u></a>	<b>INoiseFigure6</b>	Read DUT output port map
<a href="#"><u>ImpedanceStates</u></a>	INoiseFigure	Sets the number of impedance states to use during calibrated measurements.
<a href="#"><u>NoiseAverageFactor</u></a>	INoiseFigure	Set averaging of noise receiver.
<a href="#"><u>NoiseAverageState</u></a>	INoiseFigure	Turn noise averaging ON and OFF
<a href="#"><u>NoiseBandwidth</u></a>	INoiseFigure	Set bandwidth of noise receiver.
<a href="#"><u>NoiseGain</u></a>	INoiseFigure	Set gain state of noise receiver.
<a href="#"><u>NoiseReceiver</u></a>	<b>INoiseFigure5</b>	Sets and returns the receiver to use for noise measurements.
<a href="#"><u>NoiseReceiverSweepTime</u></a>	<b>INoiseFigure3</b>	Returns an estimate of sweep time.
<a href="#"><u>NoiseTuner</u></a>	INoiseFigure	Sets and returns the noise tuner identifier,
<a href="#"><u>NoiseTunerIn</u></a>	INoiseFigure	Sets and returns the port identifier of the ECal noise tuner Input
<a href="#"><u>NoiseTunerOut</u></a>	INoiseFigure	Sets and returns the port identifier of the ECal noise tuner Output
<a href="#"><u>SourcePullForSParameters</u></a>	<b>INoiseFigure4</b>	Set and read the use of source pull technique to compute S22 on Noise Figure on Converters.

### NoiseFigure History

Interface	Introduced with PNA Rev:
INoiseFigure	8.0
INoiseFigure3	9.0
INoiseFigure4	9.1
INoiseFigure5	9.2
INoiseFigure6	9.22

---

Last Modified:

30-Apr-2010	Added NF6
29-May-2007	MN New topic

## PathConfiguration Object

---

### Description

Provides access to the path configuration currently active on the channel object.

To load, store, or delete a configuration, see [ConfigurationManager](#) Object.

### Accessing the PathConfiguration object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim chan as Channel
Set chan = app.ActiveChannel

Dim pathConfig As PathConfiguration
Set pathConfig = chan.PathConfiguration
```

#### Note:

To learn how to make configuration (element) settings, see this [Path Configuration Example](#)

Also see this [list of configurable elements and settings](#).

### See Also:

- [PathConfigurationManager Object](#)
- [PathElement Object](#)
- [Path Configurator](#) UI
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Methods	Interface	Description
	<a href="#">See History</a>	
<a href="#">Elements</a>	IPathConfiguration	Elements are the objects that can be configured (switches and so forth). See the <a href="#">list of elements</a> and settings.
<a href="#">Store</a>	IPathConfiguration	Saves the current configuration to the specified name.
Properties	Interface	Description
	<a href="#">See History</a>	
<a href="#">DescriptiveText</a>	IPathConfiguration	Write and read descriptive text associated with the configuration.



<a href="#">Element</a>	IPathConfiguration	Returns a handle to the <a href="#">IPathElement</a> object.
<a href="#">Name</a>	IPathConfiguration	Returns the name of the current configuration.
<a href="#">Parent</a>	IPathConfiguration	Returns a pointer to the parent COM object (Channel).

#### IPathConfiguration History

Interface	Introduced with PNA Rev:
IPathConfiguration	7.2

## PathConfigurationManager Object

---

### Description

These commands allow configurations to be stored, loaded, or deleted on the PNA.

To **make** path configuration settings, see [PathConfiguration Object](#) and the [PathElement Object](#)

### Accessing the PathConfigurationManager object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim pathConfig As PathConfigurationManager
Set pathConfig = app.PathConfigurationManager
```

#### Note:

To learn how to make configuration (element) settings, see this [Path Configuration Example](#)

Also see this [list of configurable elements and settings](#).

### See Also:

- [Path Configuration Example](#)
- [Path Configurator](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">DeleteConfiguration</a>	IPathConfigurationManager	Deletes the specified configuration from the PNA.
<a href="#">LoadConfiguration</a>	IPathConfigurationManager	Loads the named configuration.
<a href="#">StoreConfiguration</a>	IPathConfigurationManager	Saves the path configuration

Properties	Interface <a href="#">See History</a>	Description
<a href="#">Configurations</a>	IPathConfigurationManager	Returns a list of configuration names stored in the PNA.

[Parent](#)

IPathConfigurationManager Returns a handle to the Application object.

#### IPathConfigurationManager History

Interface	Introduced with PNA Rev:
IPathConfigurationManager	7.2

## PathElement Object

---

### Description

Provides access to the settings for the PathElement object.

### Accessing the PathElement object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim chan as Channel
Set chan = app.ActiveChannel

Dim pathConfig As PathConfiguration
Set pathConfig = chan.PathConfiguration

Dim element as PathElement
Set element = pathConfig.PathElement("Src1")
```

#### Note:

To learn how to make configuration (element) settings, see this [Path Configuration Example](#)

Also see this [list of configurable elements and settings](#).

### See Also:

- [Path Configurator](#)
- [PathConfigurationManager Object](#)
- [PathConfiguration Object](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface	Description
	<a href="#">See History</a>	

None

Properties	Interface	Description
	<a href="#">See History</a>	

<a href="#">Name</a>	IPathElement	Returns the name of the element.
<a href="#">Parent</a>	IPathElement	Returns a pointer to the Parent Object ( <a href="#">PathConfiguration</a> )
<a href="#">Value</a>	IPathElement	Read / Write get the current setting for the element.
<a href="#">Values</a>	IPathElement	Returns all valid settings for the element.

#### IPathElement History

Interface	Introduced with PNA Rev:
IPathElement	7.2

## PortExtension Object **Superseded**

ALL methods and properties on the PortExtension Object are Superseded with the [Fixturing Object](#).

### Description

Contains the methods and properties that control Port Extensions.

### Accessing a PortExtension object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim PortExt As PortExtension
Set PortExt = app.PortExtension
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)
- [Superseded commands](#)

### Methods

None

Property	Description
<a href="#">Input A</a>	Sets the Input A extension value.
<a href="#">Input B</a>	Sets the Input B extension value.
<a href="#">Input C</a>	Sets the Input C extension value.
<a href="#">Port 1</a>	Sets the Port 1 extension value.
<a href="#">Port 2</a>	Sets the Port 2 extension value.
<a href="#">Port 3</a>	Sets the Port 3 extension value.
<a href="#">State</a>	Turns Port Extensions ON and OFF.

### IPort Extension History

Interface	Introduced with PNA Rev:
IPort Extension	1.0

## PowerLossSegment Object

---

### Description

Contains the properties describing a segment of the power loss table used in source power calibration.

You can get a handle to one of these segments through the [segments.Item](#) Method of the PowerLossSegments collection.

### Accessing the PowerLossSegment object

You can get a handle to one of these segments through `PowerLossSegments.Item(n)`

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim PwrLossSeg As PowerLossSegment
Set PwrLossSeg = app.SourcePowerCalibrator.PowerLossSegments(1)
```

### See Also:

- [About Source Power Cal](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

### Methods

None

Properties	Description
------------	-------------

<a href="#">Frequency</a>	The frequency (Hz) associated with this segment.
---------------------------	--

<a href="#">Loss</a>	The loss value (dB) associated with this segment.
----------------------	---

<a href="#">SegmentNumber</a>	Returns the number of this segment
-------------------------------	------------------------------------

### IPowerLossSegment History

Interface	Introduced with PNA Rev:
IPowerLossSegment	2.0





## PowerLossSegments Collection

---

### Description

A collection object that provides a mechanism for iterating through the segments of the power loss table used in source power calibration. The power loss table can contain up to 100 segments.

### Accessing the PowerLossSegments collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim PwrLossSegs As PowerLossSegments
Set PwrLossSegs = app.SourcePowerCalibrator.PowerLossSegments
```

### See Also:

- [PowerLossSegment Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Add</a>	Adds a PowerLossSegment object to the collection.
<a href="#">Item</a>	Use to get a handle to a PowerLossSegment object in the collection.
<a href="#">Remove</a>	Removes an object from the collection.
Properties	Description
<a href="#">Count</a>	Returns the number of objects in the collection.
<a href="#">Parent</a>	Returns a handle to the Parent object (SourcePowerCalibrator) of this collection.

---

Last Modified:

2-Jun-2008    Added segment limit

## PowerMeterInterface Object

### Description

Contains the properties used to select a power meter and sensor to be used for a source power calibration.

**Note:** This object replaces the [PowerMeterGPIBAddress Property](#).

### Accessing the PowerMeterInterface object

Get a handle to a power meter object using the [PowerMeterInterfaces](#) collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)
Dim pwrMtrInterfaces As PowerMeterInterfaces
Set pwrMtrInterfaces = app.SourcePowerCalibrator.PowerMeterInterfaces
If pwrMtrInterfaces.Count > 0 Then
    Dim pwrMtrInterface As PowerMeterInterface
    Set pwrMtrInterface = pwrMtrInterfaces(1)
    pwrMtrInterface.Path = naUSB
    pwrMtrInterface.Locator = "Agilent Technologies,U2000A,MY12345678"
End If
```

### See Also:

- [Source Power Calibration](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

### Methods

None

### Properties

### Description

#### [Path](#)

Specifies the interface to use: GPIB, USB, LAN

#### [Locator](#)

Specifies the location (address) of the power meter/sensor.

**IPowerMeterInterface History**

Interface	Introduced with PNA Rev:
-----------	--------------------------

IPowerMeterInterface	7.50
----------------------	------

Last Modified:

5-Jul-2007    MX New topic

## PowerMeterInterfaces Collection

---

### Description

A collection object that provides a mechanism for accessing the PowerMeterInterface objects.

The collection size is limited to one PowerMeterInterface object. By default, that PowerMeterInterface object refers to GPIB, and to the GPIB address that is currently set for the power meter on that PNA.

The power meter is specified by using the [Path](#) property.

### Accessing the PowerMeterInterfaces collection

Get a handle to a power meter object using the PowerMeterInterfaces collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)
Dim pwrMtrInterfaces As PowerMeterInterfaces
Set pwrMtrInterfaces = app.SourcePowerCalibrator.PowerMeterInterfaces
If pwrMtrInterfaces.Count > 0 Then
    Dim pwrMtrInterface As PowerMeterInterface
    Set pwrMtrInterface = pwrMtrInterfaces(1)
    pwrMtrInterface.Path = naUSB
    pwrMtrInterface.Locator = "Agilent Technologies,U2000A,MY12345678"
End If
```

### See Also:

- [Source Power Calibration](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

### Methods

[Item](#) Use to get a handle to a [PowerMeterInterface](#) object in the collection.

Properties	Description
<a href="#">Count</a>	Returns the number of objects in the collection.
<a href="#">Parent</a>	Returns a handle to the SourcePowerCalibrator

**IPowerMeterInterfaces History**

Interface	Introduced with PNA Rev:
-----------	-----------------------------

IPowerMeterInterfaces	7.50
-----------------------	------

Last Modified:

- 5-Mar-2009    Added Parent Property
- 9-Jul-2007    MX New topic

## PowerSensor Object

---

### Description

Each power sensor connected to the power meter associated with Source Power Calibration will have a PowerSensor object created to represent it. These PowerSensor objects reside in the [PowerSensors](#) collection within the SourcePowerCalibrator object. You cannot directly create PowerSensor objects, but can only retrieve existing ones from the PowerSensors collection.

The PowerSensorCalFactorSegment object is also accessed through the PowerSensor object. These are accessed through the CalFactorSegments collection in the PowerSensor object.

### Accessing a PowerSensor object

```
Dim pna As AgilentPNA835x.Application
Set pna = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim powerCalibrator as SourcePowerCalibrator
Dim powerSensor as PowerSensor
Dim calFactorSegment as PowerSensorCalFactorSegment

Set powerCalibrator = pna.SourcePowerCalibrator

' Specify GPIB address of the power meter.
powerCalibrator.PowerMeterGPIBAddress = 13

' Each time the PowerSensors collection is accessed, the power meter is queried to
determine which channels have sensors attached. The collection is updated
accordingly.

If powerCalibrator.PowerSensors.Count > 0 Then
' If channel B of the meter has a sensor attached but channel A does not, then
element 1 of the
' collection is sensor B. Whenever channel A has a sensor, sensor A will be element
1.
Set powerSensor = powerCalibrator.PowerSensors(1)
' Insert one new PowerSensorCalFactorSegment at the beginning of the collection
(index 1).

powerSensor.CalFactorSegments.Add(1)
' Assign our variable to refer to that object.
Set calFactorSegment = powerSensor.CalFactorSegments(1)

' Set property values for that object.
calFactorSegment.Frequency = 300000
' frequency in Hz
calFactorSegment.CalFactor = 98
' cal factor in percent

End If
```

See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

(**Bold** Methods or Properties provide access to a child object)

Methods	Description
---------	-------------

None

Properties	Description
------------	-------------

<a href="#"><b>CalFactorSegments</b></a>	Collection for iterating through the segments of a power sensor cal factor table.
<a href="#">MaximumFrequency</a>	Maximum usable frequency (Hz) specified for this power sensor.
<a href="#">MinimumFrequency</a>	Minimum usable frequency (Hz) specified for this power sensor.
<a href="#">PowerMeterChannel</a>	Identifies which power sensor this object corresponds to ( or which channel of the power meter the sensor is connected to).
<a href="#">ReferenceCalFactor</a>	Reference cal factor (%) associated with this power sensor.

#### IPowerSensor History

Interface	Introduced with PNA Rev:
IPowerSensor	2.0



# PowerSensorCalFactorSegment Object

## Description

Contains the properties describing a segment of a power sensor cal factor table.

## Accessing the PowerSensorCalFactorSegment object

You can get a handle to one of these segments through [CalFactorSegments](#).Item(n)

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim calFactSeg As CalFactorSegments
Set calFactSeg = app.SourcePowerCalibrator.PowerSensors(1).CalFactorSegments(1)
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

## Methods

None

Properties	Description
<a href="#">Frequency</a>	The frequency (Hz) associated with this segment. Shared with the PowerLossSegment Object
<a href="#">CalFactor</a>	The cal factor (%) associated with this segment.
<a href="#">SegmentNumber</a>	Returns the number of this segment. Shared with the PowerLossSegment Object

## IPowerSensorCalFactorSegment History

Interface	Introduced with PNA Rev:
IPowerSensorCalFactorSegment	2.0

## PowerSensors Collection

---

### Description

A collection object that provides a mechanism for iterating through the PowerSensor objects which are connected to the power meter. Each time this collection object is accessed, the power meter is queried to determine how many sensors are connected to it. The collection size and order of objects is then adjusted accordingly before the requested method or property operation is performed. The power meter is specified by using the [PowerMeterGPIBAddress](#) property of the [SourcePowerCalibrator](#) object.

### Accessing the PowerSensors Collection

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim PwrSensors As PowerSensors
Set PwrSensors = app.SourcePowerCalibrator.PowerSensors
```

### See Also:

- [PowerSensor Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Description
<a href="#">Item</a>	Use to get a handle to a PowerSensor object in the collection.
Properties	Description
<a href="#">Count</a>	Returns the number of objects in the collection.
<a href="#">Parent</a>	Returns a handle to the Parent object ( <a href="#">SourcePowerCalibrator</a> ) of this collection.

---

## Preferences Object

---

### Description

Sets the preferences for the behavior of several properties.

### Accessing the Preferences object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim pref As Preferences
Set pref = app.Preferences
```

### See Also:

- [PNA Preferences](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

(**Bold** Methods or Properties provide access to a child object)

### Methods

<a href="#">RestoreDefaults</a>	IPreferences9	Restores preference settings to their factory defaults.
---------------------------------	---------------	---

Properties	Interface	Description
	<a href="#">See History</a>	

<a href="#">AuxTriggerScopelsGlobal</a>	IPreferences5	Sets the External Trigger OUT behavior to have either Global or Channel scope.
---	---------------	--

<a href="#">CitiContents</a>	IPreferences	Specifies the contents of subsequent citifile saves. <b>Superseded</b> with <a href="#">SaveData</a>
------------------------------	--------------	---

<a href="#">CitiFormat</a>	IPreferences	Specifies the format of subsequent citifile saves. <b>Superseded</b> with <a href="#">SaveData</a>
----------------------------	--------------	---

<a href="#">DisplayColors</a>	IPreferences10	Provides access to the <a href="#">ComColors</a> Object.
-------------------------------	----------------	--

<a href="#">EnableSourceUnleveledEvents</a>	IPreferences6	Specifies whether or not to report Source Unleveled errors as system events.
---	---------------	--

<a href="#"><u>ExternalDeviceDeActivatePolicy</u></a>	IPreferences10	External Devices remain activated or are de-activated when the PNA is Preset or when a Instrument State is recalled.
<a href="#"><u>FrequencyOffsetRangeForCalComputations</u></a>	IPreferences10	Specifies the FOM frequency range to use when performing calibration.
<a href="#"><u>OffsetReceiverAttenuator</u></a>	IPreferences6	Mathematically offset the test port receiver.
<a href="#"><u>OffsetSourceAttenuator</u></a>	IPreferences6	Mathematically offset the reference receiver.
<a href="#"><u>Port1NoiseTunerSwitchPresetsToExternal</u></a>	IPreferences8	Sets default setting for Noise Figure switch.
<a href="#"><u>PowerOnDuringRetraceMode</u></a>	IPreferences4	Specify whether to turn RF power ON or OFF during a retrace for single-band frequency or segment sweeps ONLY.
<a href="#"><u>PowerSweepRetracePowerMode</u></a>	IPreferences3	At the end of a power sweep, specifies whether to maintain source power at the start or stop power level.
<a href="#"><u>PreferInternalTriggerOnChannelSingle</u></a>	IPreferences2	Sets the preference for chan.Single behavior.
<a href="#"><u>PreferInternalTriggerOnUnguidedCal</u></a>	IPreferences2	Set the preference for the trigger behavior when performing an Unguided calibration.
<a href="#"><u>PresetPowerState</u></a>	IPreferences11	Set and return the Preset Power ON/OFF state.
<a href="#"><u>PrintColors</u></a>	IPreferences10	Provides access to the <a href="#"><u>ComColors</u></a> Object.
<a href="#"><u>RemoteCalStoragePreference</u></a>	IPreferences7	Specifies the default manner in which calibrations performed via SCPI or COM are to be stored.
<a href="#"><u>SnPFormat</u></a>	IPreferences	Specifies the format of subsequent .S1P, .S2P, .S3P file saves.
<a href="#"><u>TwoPointGroupDelayAperture</u></a>	IPreferences11	Sets Group Delay Aperture default to 2 points.

## IPreferences History

Interface	Introduced with PNA Rev:
IPreferences	4.0
IPreferences2	6.0
IPreferences3	7.2
IPreferences4	6.04
IPreferences5	7.10
IPreferences6	7.20
IPreferences7	7.21
IPreferences8	8.0
IPreferences9	8.2
IPreferences10	9.0
IPreferences11	9.2

## PulseGenerator Object

---

### Description

Contains the properties for configuring the five internal pulse generators in the PNA-X.

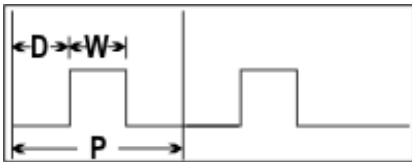
[Learn more about the PNA-X Pulse Generators.](#)

### Accessing the PulseGenerator object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim pulse as PulseGenerator
Set pulse = chan.PulseGenerator
```

Each pulse generator is specified in the Pulse Generator properties. See below.

### Pulse Definitions



- D = Delay; the time before each pulse begins
- W = Width; the time the pulse is ON
- P = Period; one complete pulse cycle
- W/P = Duty Cycle; the ratio of pulse ON/OFF

**Important:** If **D + W** is greater than **P**, then undefined PNA behavior results. There is NO error message or warning.

### See Also:

- [IF Path Block Diagram.](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About PNA-X Pulse Measurements](#)
- [Example Programs](#)

Methods	Description
---------	-------------

None

Properties	Interface <a href="#">See History</a>	Description
------------	--	-------------

<a href="#">Delay</a>	IPulsedGenerator	Sets the pulse delay.
<a href="#">DelayIncrement</a>	IPulsedGenerator	Sets the pulse delay increment.
<a href="#">Period</a>	IPulsedGenerator	Sets the pulse-period (1/PRF) for ALL PNA-X internal pulse generators.
<a href="#">State</a>	IPulsedGenerator	Turns the specified pulse generator ON and OFF.
<a href="#">SubPointTrigger</a>	IPulsedGenerator2	Enables / Disables subpoint triggering.
<a href="#">TriggerInPolarity</a>	IPulsedGenerator3	Sets the polarity of trigger to which the internal pulse generators will respond when being externally triggered.
<a href="#">TriggerInType</a>	IPulsedGenerator3	Sets the type of trigger to which the internal pulse generators will respond when being externally triggered.
<a href="#">Width</a>	IPulsedGenerator	Sets the pulse width for the specified pulse generator.

#### IPulseGenerator History

Interface	Introduced with PNA Rev:
IPulseGenerator	7.2
IPulseGenerator2	8.55.09
IPulseGenerator3	9.10

Last Modified:

9-Dec-2009	Added 9.10 commands
20-Jul-2009	Added SubPointTrig
1-Jan-2007	MX New topic



## RxLevelingConfiguration Object

---

### Description

Contains the properties for configuring Receiver Leveling in the PNA.

### Accessing the RxLevelingConfiguration object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim RxLevel as RxLevelingConfiguration
Set RxLevel = chan.RxLevelingConfiguration
```

### See Also:

- [About Receiver Leveling](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Program](#)

Methods		Description
None		

Properties	Interface	Description
	<a href="#">See History</a>	
<a href="#">FastMode</a>	IReceiverLevelingConfiguration	Select separate IFBW for leveling sweeps
<a href="#">IterationNumber</a>	IReceiverLevelingConfiguration	Max number of leveling sweeps
<a href="#">LevelingIFBW</a>	IReceiverLevelingConfiguration	IFBW for leveling sweeps
<a href="#">PowerMax</a>	IReceiverLevelingConfiguration	Max power for safe mode
<a href="#">PowerMin</a>	IReceiverLevelingConfiguration	Min power for safe mode
<a href="#">PowerOffset</a>	IReceiverLevelingConfiguration	Offset power for external components
<a href="#">PowerStep</a>	IReceiverLevelingConfiguration	Power step for safe mode

<a href="#">ReferenceReceiver</a>	IReceiverLevelingConfiguration	Select a receiver
<a href="#">SafeMode</a>	IReceiverLevelingConfiguration	Enable safe mode
<a href="#">State</a>	IReceiverLevelingConfiguration	Enable receiver leveling
<a href="#">Tolerance</a>	IReceiverLevelingConfiguration	Tolerance value for leveling sweeps

### ReceiverLevelingConfiguration History

Interface	Introduced with PNA Rev:
IReceiverLevelingConfiguration	8.5

Last Modified:

13-Feb-2009    MX New topic

## SCPIStringParser Object

---

### Description

Provides the ability to send a SCPI command from within the COM command. The two commands differ in the following ways:

**Execute** - will not return an error unless the Execute command itself fails, which is unlikely. Otherwise, you are required to read the SCPI error queue for errors that were caused by the SCPI command. The Execute command operates with minimal interference between you, the programmer, and the SCPI parser. It does not presume how you want to handle errors: handle by ignore, handle by reading the status byte, etc. This command was defined because automation engines like VB throw runtime errors when a COM method returns a failed HRESULT.

**Parse** - parses the input command, and then reads the SCPI error queue until the queue is empty. If the queue contains errors, Parse returns a failed HRESULT (E\_NA\_BAD\_SCPI\_EXECUTE). It then creates an IErrorInfo object and bundles the error numbers and descriptions into the error object. This object is available so that you can detect the failed HRESULT and interrogate the errorInfo object for more details.

#### The SCPIStringParser Methods can NOT be used with:

- [SCPI Status Reporting](#). However, the \*OPC? will work.
- Transferring Binary (block) data with commands such as: [MMEM:TRAN](#) or [CALC:DATA](#) with [Format:Data](#) set to Real32 or Real64

### Accessing the ScpiStringParser object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim SCPI As IScpiStringParser
Set SCPI = app.ScpiStringParser
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)
- [See an example](#) of how to return error information when using the [Parse method](#).

Methods	Interface	Description
---------	-----------	-------------

[Parse](#)      ISCPIStrParser      Provides the ability to send a SCPI command from within the COM command.

[Execute](#)      **ISCPIStrParser2**      Does not convert scpi errors. Use :SYST:ERR?

## Properties

None

## History

Interface	Introduced with PNA Rev:
ISCPIStrParser	1.0
ISCPIStrParser2	3.0

# Segment Object

## Description

Contains the methods and properties that affect a sweep segment.

**Note:** All of these properties are shared with at least one of the following objects: Channel, Cal Set, PowerSensorCalFactorSegment, or PowerLossSegment.

## Accessing a Segment object and setting Segment Properties

You can get a handle to a sweep segment through the segments collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim segs As ISegments
Set segs = app.ActiveChannel.Segments

segs.Add(1)
segs(1).NumberOfPoints = 30
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Segment Sweep](#)
- [Example Programs](#)

## Methods

None

Properties	Interface	Description
<a href="#">centerFrequency</a>	ISegment	Sets or returns the center frequency of the segment. Shared with the Channel Object
<a href="#">DwellTime</a>	ISegment	Dwell time value. Shared with the Channel Object
<a href="#">FrequencySpan</a>	ISegment	Sets or returns the frequency span of the segment. Shared with the Channel Object
<a href="#">IFBandwidth</a>	ISegment	Sets or returns the IF Bandwidth of the segment. Shared with the Channel Object and Cal Set object.

<a href="#">NumberOfPoints</a>	ISegment	Sets or returns the Number of Points of the segment. Shared with the Channel Object
<a href="#">SegmentNumber</a>	ISegment	Returns the number of the current segment.
<a href="#">StartFrequency</a>	ISegment	Sets or returns the start frequency of the segment. Shared with the Channel Object
<a href="#">State</a>	ISegment	Turns On or OFF a segment.
<a href="#">StopFrequency</a>	ISegment	Sets or returns the stop frequency of the segment. Shared with the Channel Object
<a href="#">SweepTime</a>	<b>ISegment2</b>	Sets or returns the sweep time of the segment. Shared with the Channel Object
<a href="#">TestPortPower</a>	ISegment	Sets or returns the RF power level of the segment. Shared with the Channel Object

#### ISegment History

Interface	Introduced with PNA Rev:
ISegment	1.0
ISegment2	7.1

Last modified:

9/29/06    MQQ Added Sweep time

## Segments Collection

---

### Description

The segment collection provides a mechanism for iterating through the sweep segments of a channel. Sweep segments are a potentially faster method of sweeping the analyzer through only the frequencies of interest. Learn more about [Segment Sweep](#).

### Accessing the Segments collection

There are two paths to the Segments Collection:

1. From the [Channel](#) object
2. From the [FOMRange](#) object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim segs As ISegments
Set segs = app.ActiveChannel.Segments
```

or

```
Set segs = app.ActiveChannel.FOM.FOMRange(1).Segments
```

### See Also:

- [Segment Object](#) to learn how to set the properties for individual segments.
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">Add</a>	ISegments	Adds an item to either the Segments collection.
<a href="#">Item</a>	ISegments	Use to get a handle to a segment in the collection..
<a href="#">GetAllSegments</a>	<b>ISegments5</b>	Downloads a segment table from the PNA
<a href="#">Remove</a>	ISegments	Removes an item from a collection of objects.
<a href="#">SetAllSegments</a>	<b>ISegments2</b>	Uploads a segment table to the PNA.

Properties		Description
<a href="#">AllowArbitrarySegments</a>	<b>ISegments3</b>	Enables the setup of arbitrary segment sweep
<a href="#">Count</a>	ISegments	Returns the number of items in a collection of objects.
<a href="#">IF Bandwidth Option</a>	ISegments	Enables the IFBandwidth to be set on individual sweep segments.
<a href="#">Parent</a>	ISegments	Returns a handle to the current naNetworkAnalyzer application..
<a href="#">Source Power Option</a>	ISegments	Enables setting the Source Power for a segment.
<a href="#">SweepTimeOption</a>	<b>ISegments4</b>	Enables the <a href="#">Sweep time</a> or <a href="#">Dwell time</a> to be set independently on sweep segments.

### ISegments History

Interface	Introduced with PNA Rev:
ISegments	1.0
ISegments2	3.5
ISegments3	4.2
ISegments4	7.1
ISegments5	8.6

Last modified:

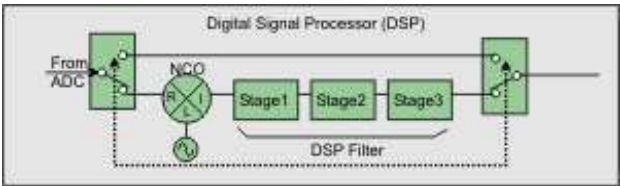
23-Jul-2009	Added more description
28-Apr-2009	Added GetAllSegments
8-Mar-2007	Modified access via fom
9/29/06	Added ISegments4



# SignalProcessingModuleFour Object

## Description

Contains the properties for configuring the DSP (digital filters) in the PNA-X.



[See the entire IF Path Block diagram.](#)

## Accessing the SignalProcessingModuleFour object

```
Dim app as AgilentPNA835x.Application
Dim chan as Channel
Set chan = app.ActiveChannel
Dim digFilter as SignalProcessingModuleFour
Set digFilter = chan.SignalProcessingModuleFour
```

## See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [About PNA-X Pulse Capabilities](#)
- [Example Programs](#)

Methods	Description
---------	-------------

None

Properties	Interface	Description
------------	-----------	-------------

<a href="#">ADCCaptureMode</a>	ISPM4	Sets ADC capture mode: auto or manual
<a href="#">FilterErrors</a>	ISPM4	Returns errors with manual digital filter settings
<a href="#">FilterMode</a>	ISPM4	Sets digital filter mode:auto or manual
<a href="#">Stage1Coefficients</a>	ISPM4	Sets Stage1Coefficients

<a href="#"><u>Stage1Frequency</u></a>	ISPM4	Sets Stage1 NCO frequency
<a href="#"><u>Stage1MaximumCoefficient</u></a>	ISPM4	Returns the maximum value of any single stage1 coefficient.
<a href="#"><u>Stage1MaximumCoefficientCount</u></a>	ISPM4	Returns the maximum number of Stage1 coefficients.
<a href="#"><u>Stage1MaximumCoefficientSum</u></a>	ISPM4	Returns the maximum sum of all Stage1 coefficients.
<a href="#"><u>Stage1MinimumCoefficientCount</u></a>	ISPM4	Returns the minimum number of Stage1 coefficients
<a href="#"><u>Stage2Coefficients</u></a>	ISPM4	Sets Stage2 Coefficients
<a href="#"><u>Stage2MaximumCoefficient</u></a>	ISPM4	Returns the maximum value of any single stage2 coefficient.
<a href="#"><u>Stage2MaximumCoefficientCount</u></a>	ISPM4	Returns the maximum number of Stage2 coefficients
<a href="#"><u>Stage2MaximumCoefficientSum</u></a>	ISPM4	Returns the maximum sum of all Stage2 coefficients.
<a href="#"><u>Stage2MinimumCoefficientCount</u></a>	ISPM4	Returns the minimum number of Stage2 coefficients
<a href="#"><u>Stage3FilterType</u></a>	ISPM4	Sets and returns stage3 filter type
<a href="#"><u>Stage3FilterTypes</u></a>	ISPM4	Returns the names of supported types of Stage3 filters.
<a href="#"><u>Stage3Parameter</u></a>	ISPM4	Sets and returns the parameter value of the current filter type.
<a href="#"><u>Stage3ParameterMaximum</u></a>	ISPM4	Returns maximum parameter value for the current filter type.
<a href="#"><u>Stage3ParameterMinimum</u></a>	ISPM4	Returns minimum parameter value for the current filter type.
<a href="#"><u>Stage3Parameters</u></a>	ISPM4	Returns the names of parameters for the current filter type.

**ISignalProcessingModuleFour History**

Interface	Introduced with PNA Rev:
-----------	--------------------------

ISignalProcessingModuleFour	7.2
-----------------------------	-----

---

Last Modified:

5-Jan-2007    MX New topic

## SMCType Object

---

### Description

Contains the methods and properties to perform an Scalar Measurement Calibration for the Frequency Converter Application (option 083).

### Accessing the SMCType object

See an example which [creates and calibrates an SMC measurement](#).

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">AcquireStep</a>	ISMCType	Acquire the measurement data for the specified step in the calibration process.
<a href="#">GenerateErrorTerms</a>	ISMCType	Generates the error terms for the calibration.
<a href="#">GenerateSteps</a>	ISMCType	Returns the number of steps required to complete the calibration.
<a href="#">GetStepDescription</a>	ISMCType	Returns the description of the specified step calibration process.
<a href="#">ImportDataSet</a>	ISMCType4	Imports separate power meter data for SMC cal.
<a href="#">Initialize</a>	ISMCType	Begins a calibration.

Properties	Description
<a href="#">AutoOrient</a>	ISMCType Sets ECAL module automatic orientation ON or OFF.
<a href="#">CalibrationPort</a>	ISMCType Sets or returns the calibration source port for the calibration.
<a href="#">CalKitType</a>	ISMCType Sets and returns a calibration kit type for calibration.

<a href="#"><u>CompatibleCalKits</u></a>	ISMCType	Returns a list of cal kits that are compatible with the connector type for the specified port.
<a href="#"><u>ConnectorType</u></a>	ISMCType	Sets or queries the connector type for the specified port.
<a href="#"><u>Do2PortEcal</u></a>	ISMCType	Specify ECAL or Mechanical calibration.
<a href="#"><u>EcalCharacterization</u></a>	ISMCType	Specifies the characterization data within an ECal module to be used for the calibration.
<a href="#"><u>EcalOrientation</u></a>	ISMCType	Specifies which port of the ECal module is connected to which port of the PNA when the AutoOrient property = False.
<a href="#"><u>FixedDelay</u></a>	ISMCType5	Set and return the known delay through the calibration mixer.
<a href="#"><u>MixerCharacterizationFile</u></a>	ISMCType5	Set the filename of the S2P file used to characterize the calibration mixer.
<a href="#"><u>NetworkFilename</u></a>	ISMCType2	Specifies the S2P filename to embed or de-embed on the input or output of your mixer measurement.
<a href="#"><u>NetworkMode</u></a>	ISMCType2	Embed (add) or de-embed (remove) circuit network effects on the input and output of your mixer measurement.
<a href="#"><u>OmitIsolation</u></a>	ISMCType	Sets and returns whether Isolation portion of the calibration will be performed or not.
<a href="#"><u>SeparatePowerCal</u></a>	ISMCType4	Use a Thru standard or to use two power sensor connections during the SMC power cal
<a href="#"><u>ThruCalMethod</u></a>	ISMCType	Sets and returns the method for performing the thru portion of the calibration.
<a href="#"><u>ValidConnectorTypes</u></a>	ISMCType	Returns a list of connector types for which there are calibration kits.

## ISMCType History

Interface	Introduced with PNA Rev:
ISMCType	3.5
ISMCType2	6.0
ISMCType4	9.0

## SourcePowerCalibrator Object

### Description

This object is a child of the Application object and is a vehicle for performing source power calibrations.

### Accessing the SourcePowerCalibrator Object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim ispc As ISourcePowerCalibrator
Set ispc = app.SourcePowerCalibrator
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

**Note:** Interface **ISourcePowerCalibrator** is abbreviated as **ISPC** in the following table.

(**Bold** Methods or Properties provide access to a child object)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">AbortPowerAcquisition</a>	ISPC	Aborts a source power cal acquisition sweep that is currently in progress.
<a href="#">AcquirePowerReadings</a>	ISPC	<b>Superseded with</b> <a href="#">AcquirePowerReadingsEx</a>
<a href="#">AcquirePowerReadingsEx</a>	ISPC4	Initiates a source power cal acquisition.
<a href="#">ApplyPowerCorrectionValues</a>	ISPC	<b>Superseded with</b> <a href="#">ApplyPowerCorrectionValuesEx</a>
<a href="#">ApplyPowerCorrectionValuesEx</a>	ISPC5	Applies correction values after completing a source power cal acquisition sweep. Optionally perform a calibration of the reference receiver used in the source power cal.
<a href="#">CheckPower</a>	ISPC2	Measures power at a specific frequency. Used to test power level before and/or after applying a source power calibration.
<a href="#">LaunchPowerMeterSettingsDialog</a>	ISPC2	Launches the Power Meter Settings dialog on the PNA.
<a href="#">SetCallInfo</a>	ISPC	<b>Superseded with</b> <a href="#">SetCallInfoEx Method</a>

<a href="#">SetCallInfo2</a>	ISPC3	<b>Superseded with</b> <a href="#">SetCallInfoEx Method</a>
<a href="#">SetCallInfoEx Method</a>	ISPC4	Specifies the channel and source port to be used for the source power calibration.
<a href="#">SetPowerAcquisitionDevice</a>	ISPC3	Sets the power sensor channel (A or B) to be used. This method is ONLY necessary when performing an SMC calibration.
Properties	Interface	Description
<a href="#">CalPower</a>	ISPC	Specifies the power level that is expected at the desired reference plane.
<a href="#">IterationsTolerance</a>	ISPC3	Sets the maximum desired deviation from the sum of the test port power and the offset value.
<a href="#">LastCalPassedTolerance</a>	ISPC7	Returns pass / fail status of the user-specified tolerance limits on the target cal power.
<a href="#">MaximumIterationsPerPoint</a>	ISPC3	Specifies maximum number of readings to take at each data point for iterating the source power.
<a href="#">PowerAcquisitionDevice</a>	ISPC2	Specifies the power sensor channel (A or B) that is currently selected for use at a specific frequency.
<a href="#">PowerLossSegments (collection)</a>	ISPC2	Collection for iterating through the segments of the power loss table used in source power calibration.
<a href="#">PowerMeterGPIBAddress</a>	ISPC	Specifies the GPIB address of the power meter.
<a href="#">PowerMeterInterfaces</a>	ISPC6	Collection for getting a handle to the available power meters.
<a href="#">PowerSensors (collection)</a>	ISPC2	Collection for iterating through the PowerSensor objects which are connected to the power meter for a source power cal.
<a href="#">ReadingsPerPoint</a>	ISPC	Specifies the maximum power readings for power meter settling.
<a href="#">ReadingsTolerance</a>	ISPC3	Power meter settling tolerance value.
<a href="#">USBPowerMeterCatalog</a>	<b>ISPC6</b>	Returns a list of USB power meters that are connected to the PNA.
<a href="#">UsePowerLossSegments</a>	ISPC	Specifies if subsequent calls to the AcquirePowerReadings method will make use of the loss table (PowerLossSegments).



### [UsePowerSensorFrequencyLimits](#)

ISPC

Specifies if subsequent calls to the AcquirePowerReadings method will make use of power sensor frequency checking capability.

### **ISourcePowerCalibrator History**

<b>Interface</b>	<b>Introduced with PNA Rev:</b>
ISourcePowerCalibrator	2.0
ISourcePowerCalibrator2	3.5
ISourcePowerCalibrator3	4.0
ISourcePowerCalibrator4	6.2
ISourcePowerCalibrator5	7.2
ISourcePowerCalibrator6	7.5
ISourcePowerCalibrator7	9.2

## SweptIMD Object

---

### Description

Controls the Swept IMD Application settings.

See [Properties to set for each sweep type](#).

### Accessing the SweptIMD object

```
Dim app as AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application")
app.CreateCustomMeasurementEx(1, "SweptIMD", "PwrMain", 1)
Dim IMD
Set IMD = app.ActiveChannel.CustomChannelConfiguration
```

### See Also:

- **Example Program** [Create and Cal a Swept IMD Measurement](#)
- [SweptIMDCal Object](#)
- [About the Swept IMD Application](#)
- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)

Method	Interface	Description
<a href="#">See History</a>		
<a href="#">SetPortMap</a>	ISweptIMD	Sets port mapping

Property	Interface	Description
<a href="#">CompositeNormalizationMode</a>	ISweptIMD	Sets and returns the method by which CTB and CSO calculations are performed.
<a href="#">CompositeNormalizedCSOPower</a>	ISweptIMD	Sets and returns the CSO Power.
<a href="#">CompositeNormalizedCTBPower</a>	ISweptIMD	Sets and returns the CTB Power.
<a href="#">CoupleTonePower</a>	ISweptIMD	ON   OFF state of power coupling for F1 and F2.

<a href="#"><u>CSONumDistortionProducts</u></a>	ISweptIMD	Sets the “N = number of distortion products” value for the calculation of the CSO parameter.
<a href="#"><u>CSOffset Property</u></a>	ISweptIMD	Sets and returns the offset that is applied to CSO measurements.
<a href="#"><u>CTBOffset Property</u></a>	ISweptIMD	Sets and returns the offset that is applied to CTB measurements.
<a href="#"><u>CTBXMODNumCarriers</u></a>	ISweptIMD	Sets the “N = Total number of carriers” value used in the calculation of the XMOD and CTB parameter.
<a href="#"><u>DeltaFrequency</u></a>	ISweptIMD	Fixed tone spacing value.
<a href="#"><u>DeltaFrequencyStart</u></a>	ISweptIMD	Start spacing of the main tones.
<a href="#"><u>DeltaFrequencyStop</u></a>	ISweptIMD	Stop spacing of the main tones.
<a href="#"><u>DeviceInputPort</u></a>	ISweptIMD	Reads input port map
<a href="#"><u>DeviceOutputPort</u></a>	ISweptIMD	Reads output port map
<a href="#"><u>F1Frequency</u></a>	ISweptIMD	Frequency of the F1 tone.
<a href="#"><u>F2Frequency</u></a>	ISweptIMD	Frequency of the F1 tone.
<a href="#"><u>FrequencyCenter</u></a>	ISweptIMD	Center frequency of the main tones.
<a href="#"><u>FrequencyCenterCenter</u></a>	ISweptIMD	Sweep center frequency when sweeping the main tones.
<a href="#"><u>FrequencyCenterSpan</u></a>	ISweptIMD	Frequency span when sweeping the main tones.
<a href="#"><u>FrequencyCenterStart</u></a>	ISweptIMD	Start frequency when sweeping the main tones.
<a href="#"><u>FrequencyCenterStop</u></a>	ISweptIMD	Stop frequency when sweeping the main tones.
<a href="#"><u>HighestOrderProduct</u></a>	ISweptIMD	Reads the highest product that can be measured by SweptIMD.
<a href="#"><u>IMToneIFBandwidth</u></a>	ISweptIMD	IF Bandwidth for measurement of the intermodulation products.
<a href="#"><u>MainToneIFBandwidth</u></a>	ISweptIMD	IF Bandwidth for measurement of the Main tones.

<a href="#">SweepType</a>	ISweptIMD	Type of sweep for a Swept IMD measurement.
<a href="#">TonePower</a>	ISweptIMD	Power level of the Main Tones.
<a href="#">TonePowerStart</a>	ISweptIMD	Start power level of the Main tones.
<a href="#">TonePowerStop</a>	ISweptIMD	Stop power level of the Main tones.

The following commands are relevant for each [IMD sweep type](#):

#### 0 - naIMDToneCWSweep

- [F1Frequency Property](#)
- [F2Frequency Property](#)
- [FrequencyCenter Property](#)
- [DeltaFrequency Property](#)
- [TonePower \(1,2\) Property](#)

#### 1 - naIMDTonePowerSweep

- [F1Frequency Property](#)
- [F2Frequency Property](#)
- [FrequencyCenter Property](#)
- [DeltaFrequency Property](#)
- [TonePowerStart Property](#)
- [TonePowerStop Property](#)

#### 2 - naIMDToneCenterFreqSweep

- [FrequencyCenterStart Property](#)
- [FrequencyCenterStop Property](#)
- [FrequencyCenterCenter Property](#)
- [FrequencyCenterSpan Property](#)
- [DeltaFrequency Property](#)
- [TonePower \(1,2\) Property](#)

### 3 - naIMDDeltaFrequencySweep

- [DeltaFrequencyStart Property](#)
- [DeltaFrequencyStop Property](#)
- [FrequencyCenter Property](#)
- [TonePower \(1,2\) Property](#)

### 4 - naIMDToneSegmentSweep (Not available for IMDx)

- [DeltaFrequency Property](#)
- [TonePower \(1,2\) Property](#)
- Also use standard [segment sweep commands](#)

### 5 - naLOPowerSweep (IMDx ONLY)

- converter.[LOStartPower Property](#)
- converter.[LOStopPower Property](#)

### ISweptIMD History

Interface	Introduced with PNA Rev:
ISweptIMD	8.33

---

#### Last Modified:

- |             |                        |
|-------------|------------------------|
| 11-Aug-2009 | Added limited port map |
| 25-Mar-2009 | Fixed Access example   |
| 19-Aug-2008 | MX New topic           |

# SweptIMDCal Object

## Description

Sets properties that are unique to a Swept IMD Cal (opt 087).  
Use the [Guided Calibration commands](#) for the remaining commands to perform a Swept IMD Cal

## Accessing the SweptIMDCal object

```
Dim app as AgilentPNA835x.Application
Set IMDcal = pna.GetCalmanager.CreateCustomCalEx\(channelNum\)
Set IMDCalExtension = IMDcal.CustomCalConfiguration
IMDCalExtension.PowerLevel = 5
```

## See Also:

- **Example Program** [Create and Cal a Swept IMD Measurement](#)
- [SweptIMD Object](#)
- [About Swept IMD measurements](#)
- [The PNA Object Model](#)
- [PNA Automation Interfaces](#)

Method	Interface	Description
<a href="#">See History</a>		

None

Property	Interface	Description
<a href="#">CalibrationFrequencies</a>	ISweptIMDCal	Perform the source power cal at the center frequencies or at all main tone frequencies.
<a href="#">CalMethod</a>	ISweptIMDCal	Sets the method by which the match-correction portion of an IMD cal is performed
<a href="#">EnableLOPowerCal</a>	ISweptIMDCal2	Enable or disable an LO power cal with an IMDx calibration.
<a href="#">Include2ndOrderProduct</a>	ISweptIMDCal	Include the second order products in the calibration.

<a href="#">MaxProduct</a>	ISweptIMDCal	Sets and returns the maximum intermod product frequencies to be calibrated.
<a href="#">PowerLevel</a>	ISweptIMDCal	Set and read the power level of the source power cal.
<a href="#">PowerSensorCalKitType</a>	ISweptIMDCal	Set and read the cal kit to be used for port 1 adapter compensation.
<a href="#">PowerSensorConnectorType</a>	ISweptIMDCal	<b>Superseded</b> Use GuidedCal. <a href="#">PowerSensorConnectorType</a>

### ISweptIMD History

Interface	Introduced with PNA Rev:
ISweptIMDCal	8.33
ISweptIMDCal2	8.55

Last Modified:

9-Sep-2008    MX New topic

## TestsetControl Object

---

### Description

A TestsetControl object is used to control one of the [supported test sets](#). Only one external test set can be controlled by the PNA at any time. The Testset Control object appears as an item in the ExternalTestsets collection, which in turn is a property of the main application object.

If the specified test set is not connected to the PNA or is not ON, then setting [Enabled](#) = True will return an error. All other properties can be set even if the test set is not connected.

**Note:** The ONLY way to load a test set configuration file is by sending the [testsets.Add](#) method. There is no method to query the test set type. See an [example program](#).

### Accessing a TestsetControl object

The [ExternalTestsets collection](#) is a property of the main **Application** Object. You can obtain a handle to a testset object by specifying an item in the collection.

### Visual Basic Example

```
Dim pna
Dim testsets As ExternalTestsets
Dim tset1 As TestsetControl
Set pna = CreateObject("AgilentPNA835x.Application")
Set testsets = pna.ExternalTestsets
Set tset1 = testsets(1)
' make COM calls on tset1 object
End Sub
```

### See Also:

- [E5091A Testset Object](#)
- [About External Testset Control](#)
- [ExternalTestset Control Example](#)
- [ExternalTestsets Collection](#)
- [The PNA Object Model](#)



Methods	Interface ( <a href="#">See history</a> )	Description
---------	--	-------------

None

Properties	Description
------------	-------------

<a href="#">ControlLines</a>	IExternalTestset	Sets the control lines of the specified Test set.
<a href="#">Enabled</a>	IExternalTestset	Enables and disables (ON/OFF) the port mapping and control line output of the specified test set.
<a href="#">ID</a>	IExternalTestset	Returns the test set ID number.
<a href="#">Label</a>	IExternalTestset	Returns the label on a given channel for the specified test set.
<a href="#">NumberOfPorts</a>	IExternalTestset	Reads the number of ports that are on the specified test set.
<a href="#">OutputPorts</a>	IExternalTestset	Sets or returns the port mappings for ALL ports.
<a href="#">PortCatalog</a>	IExternalTestset	Returns the selections available for a given logical port.
<a href="#">SelectPort</a>	IExternalTestset	Sets and returns the logical port value.
<a href="#">ShowProperties</a>	IExternalTestset	Turns status bar display of test set properties on or off.
<a href="#">Type</a>	IExternalTestset	Returns the test set model.

#### ExternalTestset History

	Interface	Introduced with PNA Rev:
	IExternalTestset	6.0
	IExternalTestset	6.0

## Trace Object

---

### Description

The Trace object controls how the measurement data is displayed. You can control scale, reference position, and value from the Trace Object.

### Accessing a Trace object

There are several ways to get a handle to a trace.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim trace As Trace
```

Then you can do any of the following:

```
Set trace = app.NAWindows(1).traces(1)
```

```
set trace = app.NAWindows.item(1).ActiveTrace
```

```
set trace = app.ActiveNAWindow.traces.item(1)
```

```
set trace = app.ActiveNAWindow.ActiveTrace
```

```
Set trace = app.Measurements(1).trace
```

```
Set trace = app.ActiveMeasurement.trace
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Traces, Channels, and Windows on the PNA](#)
- [Example Programs](#)

Methods	Description
---------	-------------

[Autoscale](#) Autoscales the trace or all of the traces in the selected window.  
Shared with the NAWindow Object

Property	Description
----------	-------------

[Name](#) Sets or returns the trace name

[ReferencePosition](#) Sets or returns the Reference Position of the active trace.

[ReferenceValue](#) Sets or returns the value of the Y-axis Reference Level of the active trace.

[YScale](#) Sets or returns the Y-axis Per-Division value of the active trace.

## ITrace History

Interface	Introduced with PNA Rev:
-----------	-----------------------------

ITrace 1.0

## Traces Collection

---

### Description

Child of the **Application** Object. A collection that provides a mechanism for getting a handle to a trace or iterating through the traces in a window.

### Accessing the Traces collection

Get a handle to the traces collection through the NaWindows collection. The following example sets the variable **trcs** to the collection of traces in window 1 of the NaWindows collection.

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim trcs As traces
Set trcs = app.NaWindows(1).traces
```

### See Also:

- [Trace Object](#)
- [Collections in the Analyzer](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	
<a href="#">Item</a>	Use to get a handle to a trace
Properties	
<a href="#">Count</a>	Returns the number of traces in the collection.
<a href="#">Parent</a>	Returns a handle to the current Application.

## Transform Object

---

### Description

Contains the methods and properties that control Time Domain transforms.

### Accessing the Transform Object

```
Dim app As AgilentPNA835x.Application
Set app = CreateObject("AgilentPNA835x.Application", <analyzerName>)

Dim trans As Transform
Set trans = app.ActiveMeasurement.Transform
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Time Domain Topics](#)
- [Example Programs](#)

**Note:** [Sweep Type](#) must be set to Linear before setting Time Domain Transform (state) ON.

Methods	Interface	Description
<a href="#">SetFrequencyLowPass</a>	ITransform	Sets low frequencies for low pass.
Properties	Description	
<a href="#">Center</a>	ITransform	Sets or returns the Center time. Shared with the Gating Object
<a href="#">CoupledParameters</a>	ITransform2	Select Transform parameters to couple
<a href="#">DistanceMarkerMode</a>	ITransform2	Sets the measurement type in order to determine the correct marker distance.
<a href="#">DistanceMarkerUnit</a>	ITransform2	Sets the unit of measure for the display of marker distance values.
<a href="#">ImpulseWidth</a>	ITransform	Sets or returns the Impulse Width of Time Domain transform windows.
<a href="#">KaiserBeta</a>	ITransform	Sets or returns the Kaiser Beta of Time Domain transform windows.
<a href="#">Mode</a>	ITransform	Sets the type of transform.

<a href="#">Span</a>	ITransform	Sets or returns the Span time. Shared with the Gating Object
<a href="#">Start</a>	ITransform	Sets or returns the Start time. Shared with the Gating Object
<a href="#">State</a>	ITransform	Turns an Object ON and OFF.
<a href="#">StepRiseTime</a>	ITransform	Sets or returns the Rise time of the stimulus in Low Pass Step Mode.
<a href="#">Stop</a>	ITransform	Sets or returns the Stop time. Shared with the Gating Object

### ITransform History

Interface	Introduced with PNA Rev:
ITransform	1.0
ITransform2	4.2

## TriggerSetup Object

---

### Description

These properties setup Global triggering that effects the entire PNA application.

### Accessing the TriggerSetup object

```
Dim app as AgilentPNA835x.Application
Dim trigSetup as ITriggerSetup
Set trigSetup = app.TriggerSetup
```

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Triggering in the PNA](#)
- [Example Programs](#)

Methods	Interface	Description
	<a href="#">See History (below)</a>	

None

Properties		Description
<a href="#">AcceptTriggerBeforeArmed</a>	ITriggerSetup2	Allows a trigger signal to be remembered and then used when the PNA becomes armed (ready to be triggered).
<a href="#">ExternalTriggerConnectionBehavior</a>	ITriggerSetup	Configures the external triggering signal for the PNA
<a href="#">ReadyForTriggerPolarity</a>	ITriggerSetup3	Specifies the polarity of Ready for Trigger output.
<a href="#">Scope</a>	ITriggerSetup	Determines whether a trigger signal affects a single channel or all channels in the PNA.
<a href="#">Source</a>	ITriggerSetup	Sets or returns the source of triggering in the PNA.

[TriggerOutputEnabled](#)

ITriggerSetup2 Enables the PNA to send trigger signals out the rear-panel TRIGGER OUT connector.

### ITriggerSetup History

Interface	Introduced with PNA Rev:
ITriggerSetup	4.0
ITriggerSetup2	4.2
ITriggerSetup3	7.50.2 and 8.2



## VMC Type Object

---

### Description

Contains the methods and properties to perform a Vector Measurement Calibration for the Frequency Converter Application (option 083).

### Accessing the VMCType object

See an example which [creates and calibrates a VMC measurement](#).

### See Also:

- [PNA Automation Interfaces](#)
- [The PNA Object Model](#)
- [Example Programs](#)

Methods	Interface <a href="#">See History</a>	Description
<a href="#">AcquireStep</a>	IVMCType	Acquire the measurement data for the specified step in the calibration process.
<a href="#">GenerateErrorTerms</a>	IVMCType	Generates the error terms for the calibration.
<a href="#">GenerateSteps</a>	IVMCType	Returns the number of steps required to complete the calibration.
<a href="#">GetStepDescription</a>	IVMCType	Returns the description of the specified step in the calibration process.
<a href="#">Initialize</a>	IVMCType	Begins a calibration.

Properties	Description
<a href="#">AutoOrient</a>	IVMCType Sets ECAL module automatic orientation ON or OFF.
<a href="#">CalKitType</a>	IVMCType Sets and returns a calibration kit type for calibration.
<a href="#">CharacterizeMixerOnly</a>	IVMCType Sets and returns whether to perform a mixer characterization ONLY or full 2-port calibration.
<a href="#">CharFileName</a>	IVMCType Specifies the .S2P mixer characterization file name.

<a href="#"><u>CharMixerReverse</u></a>	IVMCType2	Specifies the direction in which to characterize the calibration mixer.
<a href="#"><u>CompatibleCalKits</u></a>	IVMCType	Returns a list of cal kits that are compatible with the connector type for the specified port.
<a href="#"><u>ConnectorType</u></a>	IVMCType	Sets or queries the connector type for the specified port.
<a href="#"><u>Do1PortEcal</u></a>	IVMCType	Specify ECAL or Mechanical calibration for the mixer characterization portion of a VMC calibration.
<a href="#"><u>Do2PortEcal</u></a>	IVMCType	Specify ECAL or Mechanical calibration for the 2-port calibration portion of a VMC calibration.
<a href="#"><u>EcalCharacterization</u></a>	IVMCType	Specifies the characterization data within an ECal module to be used for the calibration.
<a href="#"><u>EcalOrientation1Port</u></a>	IVMCType	For Mixer Characterization ONLY - Specifies which port of the ECal module is connected to which port of the PNA
<a href="#"><u>EcalOrientation2Port</u></a>	IVMCType	For full 2-port VMC cal - Specifies which port of the ECal module is connected to which port of the PNA
<a href="#"><u>LoadCharFromFile</u></a>	IVMCType	Specifies and loads a mixer characterization (S2P) file.
<a href="#"><u>NetworkFilename</u></a>	IVMCType3	Specifies the S2P filename to embed or de-embed on the input or output of your mixer measurement.
<a href="#"><u>NetworkMode</u></a>	IVMCType3	Embed (add) or de-embed (remove) circuit network effects on the input and output of your mixer measurement.
<a href="#"><u>OmitIsolation</u></a>	IVMCType	Sets and returns whether Isolation portion of the calibration will be performed or not.
<a href="#"><u>ThruCalMethod</u></a>	IVMCType	Sets and returns the method for performing the thru portion of the calibration.
<a href="#"><u>ValidConnectorTypes</u></a>	IVMCType	Returns a list of connector types for which there are calibration kits.

#### **IVMCType History**

Interface	Introduced with PNA Rev:
IVMCType	3.5
IVMCType2	3.53
IVMCType3	6.0

## AcceptTriggerBeforeArmed Property

---

**Description** Determines what happens to an EDGE trigger signal if it occurs before the PNA is ready to be triggered. (LEVEL trigger signals are always ignored.) For more information, see [External triggering](#).

**VB Syntax** *trigsetup.AcceptTriggerBeforeArmed = boolean*

**Variable** [\(Type\)](#) - Description

*trigsetup* A [TriggerSetup2](#) (object)

*boolean* Choose from:

**False** - A trigger signal is ignored if it occurs before the PNA is ready to be triggered.

**True** - A trigger signal is remembered and then used when the PNA becomes armed (ready to be triggered). The PNA remembers only one trigger signal.

**Return Type** Boolean

**Default** False

**Examples** `trigsetup.AcceptTriggerBeforeArmed = True 'Write`

`atba = trigsetup.AcceptTriggerBeforeArmed 'Read`

**C++ Syntax** HRESULT get\_AcceptTriggerBeforeArmed( BOOL \*pVal);  
HRESULT put\_AcceptTriggerBeforeArmed( BOOL newVal);

**Interface** ITriggerSetup2

## AcquisitionDirection Property

---

**Description** Specifies the direction of each part of a 2-port calibration.

**VB Syntax** *cal.AcquisitionDirection* = *value*

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

*value* (**enum NADirection**) - Choose from:

**0 - naForward** - measures the forward direction

**1 - naReverse** - measures the reverse direction

**Return Type** Long Integer

**Default** naForward

**Examples** `cal.AcquisitionDirection = naForward`

**C++ Syntax** HRESULT AcquisitionDirection(tagNADirection dir);

**Interface** ICalibrator

## AcquisitionMode Property

---

**Description** Set and read the method by which gain compression data is acquired.

**VB Syntax** `gca.AcquisitionMode = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (**NAGCAAcquisitionMode**) Choose from:

- **naSmartSweep (0)** Iterate quickly to find compression point
- **naSweepPowerAtEachFreq2D (1)** Sweep power at each frequency
- **naSweepFreqAtEachPower2D (2)** Sweep frequency at each power level

**Return Type** Enum

**Default** **naSmartSweep**

**Examples** `gca.AcquisitionMode = naSmartSweep 'Write`

```
acqMode = gca.AcquisitionMode 'Read
```

**C++ Syntax** HRESULT get\_AcquisitionMode(tagNAGCAAcquisitionMode\* mode)  
HRESULT put\_AcquisitionMode(tagNAGCAAcquisitionMode mode)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## ActiveCalKit Property

---

**Description** Returns a handle to the Active CalKit object. The active cal kit is the kit selected for use in **Unguided** calibrations.

You can either **(1)** use the handle directly to access CalKit properties and methods, or **(2)** set a variable to the CalKit object. The variable retains a handle to the original object if another CalKit becomes active.

**VB Syntax** 1) app.**ActiveCalKit**.<setting>  
**or**  
2) Set *cKit* = app.**ActiveCalKit**

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (**object**)

<setting> A CalKit property (or method) and arguments

*cKit* (**object**) - A CalKit object

**Return Type** CalKit object

**Default** None

**Examples**

```
Public cKit as calKit
Set cKit = app.ActiveCalKit 'read
```

**C++ Syntax** HRESULT get\_ActiveCalKit (ICalKit \* kit)

**Interface** IApplication

---

Last Modified:

3-Jun-2008    Added Unguided

## ActiveChannel Property

---

**Description** Returns a handle to the Active Channel object. You can either **(1)** use the handle directly to access channel properties and methods, or **(2)** set a variable to the channel object. The variable retains a handle to the original channel if another channel becomes active.

**VB Syntax** (1) `app.ActiveChannel.<setting>`  
or  
(2) Set `chan = app.ActiveChannel`

**Variable** [\(Type\)](#) - Description

`chan` A Channel **(object)**

`app` An [Application](#) **(object)**

`<setting>` A channel property (or method) and arguments

**Return Type** Channel object

**Default** Not applicable

**Examples**

```
1) app.ActiveChannel.Averaging = 1
2) Public chan as Channel
   Set chan = app.ActiveChannel
```

**C++ Syntax** HRESULT get\_ActiveChannel( IChannel\* \*pVal)

**Interface** IApplication



## ActiveMarker Property

---

**Description** Returns a handle to the Active Marker object. You can either **(1)** use the handle directly to access Marker properties and methods, or **(2)** set a variable to the Marker object. The variable retains a handle to the original object if another Marker becomes active.

**VB Syntax** 1) *meas.ActiveMarker.<setting>*  
**or**  
2) Set *mark* = *meas.ActiveMarker*

**Variable** [\(Type\)](#) - Description

*meas* **(object)** - An Measurement object

*<setting>* A marker property (or method) and arguments

*mark* **(object)** - A marker object

**Return Type** marker object

**Default** None

**Examples**

```
Public mark as marker
Set mark = meas.ActiveMarker
```

**C++ Syntax** HRESULT get\_ActiveMarker(IMarker\*\* marker)

**Interface** IMeasurement

Read-only

## ActiveMeasurement Property

---

**Description** Returns a handle to the Active Measurement object. You can either **(1)** use the handle directly to access measurement properties and methods, or **(2)** set a variable to the measurement object. The variable retains a handle to the original measurement.

**VB Syntax** 1) `app.ActiveMeasurement.<setting>`  
**or**  
2) Set `meas = app.ActiveMeasurement`

**Variable** [\(Type\)](#) - Description

`meas` A Measurement **(object)**

`app` An [Application](#) **(object)**

`<setting>` A measurement property (or method) and arguments

**Return Type** Measurement object

**Default** None

**Examples**

```
1) app.ActiveMeasurement.Averaging = 1
2) Public meas as Measurement
   Set meas = app.ActiveMeasurement
```

**C++ Syntax** HRESULT get\_ActiveMeasurement(IMeasurement \*\*ppMeas)

**Interface** IApplication

## ActiveNAWindow Property

---

**Description** Returns a handle to the Active Window object. You can either **(1)** use the handle directly to access window properties and methods, or **(2)** set a variable to the window object. The variable retains a handle to the original window if another window becomes active.

**VB Syntax** 1) `app.ActiveNAWindow.<setting>`  
or  
2) Set `win = app.ActiveNAWindow`

**Variable** [\(Type\)](#) - Description

`win` A NAWindow (**object**)

`app` An [Application](#) (**object**)

`<setting>` A NAWindow property (or method) and arguments

**Return Type** A NAWindow object

**Default** Not applicable

**Examples**

```
Public win as NAWindow
Set win = app.ActiveWindow
```

**C++ Syntax** HRESULT get\_ActiveNAWindow(INAWindow \*\*ppWindow)

**Interface** IApplication

## ActiveXAxisRange Property

---

**Description** Sets or returns the swept parameter to display on the X-axis for the selected FCA measurement.

**VB Syntax** *mixer.ActiveXAxisRange = value*

**Variable** [\(Type\)](#) - Description

*mixer* A Mixer **(object)**

*value* **(Enum as MixerStimulusRange)** - Parameter to display on the X-axis. Choose from:

**0 - mixINPUT** - Input frequency span

**1 - mixLO\_1** - First LO frequency span

**2 - mixLO\_2** - Second LO frequency span

**3 - mixOUTPUT** - Output frequency span

**Return Type** Enum

**Default** OUTPUT

**Examples**

```
mixer.ActiveXAxisRange = 1 'Write
```

```
variable = mixer.ActiveXAxisRange 'Read
```

**C++ Syntax** HRESULT get\_ActiveXAxisRange(tagMixerStimulusRange \*Val)  
HRESULT put\_ActiveXAxisRange(tagMixerStimulusRange newVal)

**Interface** IMixer3

Last Modified:

26-Jan-2009    Corrected enums

## ActiveXAxisRange Property

---

**Description** Sets or returns the swept frequency range to display on the X-axis for the IMDx channel.

**VB Syntax** *obj.ActiveXAxisRange = value*

**Variable** [\(Type\) - Description](#)

*obj* A [Converter Object](#)

*value* **(Enum as ConverterStimulusRange)**

Swept stimulus range to display on the X-axis. Choose from:

**0 - naInputRange** - Input frequency range

**1 - naLO1Range** - LO frequency range

**3 - naOutputRange** - Output frequency range

If the specified frequency range is not swept, the default swept range is used.

**Return Type** Enum

**Default** Search is performed in the following order until a swept range is found:

1. OUTPUT
2. INPUT (If the OUTPUT is fixed)
3. Number of Points (If ALL ranges are fixed)

**Examples** `conv.ActiveXAxisRange = naInputRange 'Write`

`variable = conv.ActiveXAxisRange 'Read`

**C++ Syntax** HRESULT get\_ActiveXAxisRange(tagConverterStimulusRange range \*Val)  
 HRESULT put\_ActiveXAxisRange(tagConverterStimulusRange range newVal)

**Interface** IConverter

Last Modified:

2-Feb-2009 Added converter

26-Jan-2009 Corrected enums

## ADCCaptureMode Property

---

**Description** Sets and returns the ADC capture mode modeled as a 2-pole switch in the diagram on the [SignalProcessingModuleFour](#) page. The switch either bypasses or routes the IF through the 3-stage digital filter.

**VB Syntax** `spm4.ADCCaptureMode = value`

**Variable** [\(Type\)](#) - Description

*spm4* A [SignalProcessingModuleFour](#) (object)

*value* (Enum as NAStates) Capture mode.

**naOFF (0)** - The digital filters are used to process IF information. The filters can be configured automatically or manually using [FilterMode Property](#).

**naON (1)** - The digital filters are bypassed and the raw ADC readings are taken directly. A maximum of 4096 data points per sweep can be acquired.

**Return Type** Enum

**Default** OFF

**Examples** `spm4.ADCCaptureMode = 0 'Write`

`mode = spm4.ADCCaptureMode 'Read`

**C++ Syntax** HRESULT get\_ADCCaptureMode(tagNAStates\* pCaptureMode);  
HRESULT put\_ADCCaptureMode(tagNAStates pCaptureMode);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

18-Jun-2007 18-Jun-2007 MX New topic

## ALCLevelingMode Property

---

**Description** Sets and returns the ALC mode for the specified channel and port. Use [GetSupportedALCModes](#) to return a list of valid ALC modes for the PNA.

[Learn more about ALC mode.](#)

**VB Syntax** `chan.ALCLevelingMode (sourcePort) = value`

**Variable** [\(Type\)](#) - **Description**

*chan* **(object)** - A [Channel](#) object

*sourcePort* **(long integer)** - The source port for which to make this setting. If ports are [remapped](#), specify the logical port number.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* **(enum as naALCLevelingMode)** - Choose from:

- **0** naALCInternal
- **1** naALCExternal (E835x Only)
- **2** naALCOpenLoop (PNA-X only)

**Return Type** Enum

**Default** naALCInternal

**Examples**

```
chan.ALCLevelingMode(1) = 'Write
state = chan.ALCLevelingMode(4) 'Read
```

**C++ Syntax** HRESULT get\_ALCLevelingMode(long port, tagNAALCLevelingMode\* pVal);  
 HRESULT put\_ALCLevelingMode(long port,tagNAALCLevelingMode newVal);

**Interface** IChannel9

---

Last modified:

3-Mar-2009	Removed 3
24-Apr-2008	Added note for string names
30-Apr-2007	Edited for src strings
10/18/06	MX New topic



## ActiveTrace Property

---

**Description** Returns a handle to the Active Trace object. You can either **(1)** use the handle directly to access trace properties and methods, or **(2)** set a variable to the trace object. The variable retains a handle to the original trace if another trace becomes active.

**VB Syntax** 1) *win*.**ActiveTrace**.<setting>  
or  
2) Set *trce* = *win*.**ActiveTrace**

**Variable** [\(Type\)](#) - Description

*trce* A Trace **(object)**

*win* An NAWindow **(object)**

<setting> A trace property (or method) and arguments

**Return Type** An NAWindow object

**Default** None

**Examples**

```
1) win.ActiveTrace.Autoscale
2) Public trce as Trace
   Set trce = Application.ActiveNAWindow.ActiveTrace
```

**C++ Syntax** HRESULT get\_ActiveTrace(ITrace\* \*pVal)

**Interface** INAWindow

## AllowArbitrarySegments Property

---

**Description** Enables you to setup a segment sweep with arbitrary frequencies. The start and stop frequencies of each segment can overlap other segments. Also, each segment can have a start frequency that is greater than its stop frequency which causes a reverse sweep over that segment. Learn more about [Arbitrary Segment Sweep](#).

**VB Syntax** `segs.AllowArbitrarySegments = value`

**Variable** [\(Type\)](#) - **Description**

*segs* A [Segments](#) collection (**object**)

*value* (**boolean**)

**True** - Allows the setup of arbitrary segment sweep.

**False** - Prevents the setup of arbitrary segment sweep.

**Return  
Type** Boolean

**Default** False

**Examples** `segs.AllowArbitrarySegments = True 'Write`

`AllowArbSegs = AllowArbitrarySegments 'Read`

**C++ Syntax** HRESULT get\_AllowArbitrarySegments(VARIANT\_BOOL \*pVal)  
HRESULT put\_AllowArbitrarySegments(VARIANT\_BOOL newVal)

**Interface** ISegments3

## AlternateSweep Property

---

**Description** Sets sweeps to either alternate or chopped.

**VB Syntax** *object*.AlternateSweep = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)  
**or**  
CalSet (**object**) - Read-only property

*value* (**boolean**) - Choose either:  
**False** - Sweep mode set to **Chopped** - reflection and transmission are measured on the same sweep.  
**True** - Sweep mode set to **Alternate** - reflection and transmission measured on separate sweeps. Improves Mixer bounce and Isolation measurements. Increases cycle time.

**Return Type** boolean

**Default** False (0)

**Examples** `chan.AlternateSweep = True 'Write`

`altSwp = chan.AlternateSweep 'Read`

**C++ Syntax** HRESULT AlternateSweep(VARIANT\_BOOL \*pVal)  
HRESULT AlternateSweep(VARIANT\_BOOL newVal)

**Interface** IChannel  
ICalSet3

## AmbientTemperature Property

---

**Description** Sets and returns the temperature at which the current noise measurement is occurring. [Learn more.](#)

**VB Syntax** `noiseCal.AmbientTemperature = value`

**Variable** [\(Type\)](#) - [Description](#)

*noiseCal* A [NoiseCal](#) (object)

*value* (**double**) Ambient temperature in Kelvin.

**Return Type** Double

**Default** 295

**Examples** `noise.AmbientTemperature = 289` **Write**

`temp = noise.AmbientTemperature` **Read**

**C++ Syntax** HRESULT get\_AmbientTemperature(Double\* pValue)  
HRESULT put\_AmbientTemperature(Double pNewValue)

**Interface** INoiseCal

---

Last Modified:

6-Sep-2007    MX New topic

Read-only

## Application Property

---

**Description** Returns the name of the Analyzer making measurements on the channel.

**VB Syntax** *chan*.**Application**

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (**object**)

**Return Type** object

**Default** None

**Examples** `rfna = chan.Application 'returns the Analyzer name`

**C++ Syntax** HRESULT get\_Application(IApplication\*\* Application)

**Interface** IChannel

## ArrangeWindows Property

---

**Description** Sets the arrangement of all the windows. Overlay, Stack2, Split3 and Quad4 will create windows.

To control the state of one window, use [app.WindowState](#).

**VB Syntax** `app.ArrangeWindows = value`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (enum **NAWindowModes**) - Choose from:

0 - naTile  
1 - naCascade  
2 - naOverlay  
3 - naStack2  
4 - naSplit3  
5 - naQuad4

**Return Type** Not Applicable

**Default** naTile

**Examples** `app.ArrangeWindow = naTile 'Write`

**[C++ Syntax](#)** HRESULT put\_ArrangeWindows(tagNAWindowModes newVal)

**Interface** IApplication

## AttenuatorMode Property

---

**Description** Sets or returns the mode of operation of the attenuator control for the specified port number. This command is automatically set to Manual when an Attenuator value is set.

**VB Syntax** *object*.**AttenuatorMode**(*portNum*) = *value*

**Variable** [\(Type\)](#) - **Description**

*object* [Channel](#) (object)

or

[CalSet](#) (object) - Read-only property

*portNum* **(long)** - Port number of attenuator control to be changed.

*value* **(enum NAModes)** - Choose from:

**0 - naAuto** - Attenuator control set to automatic. The analyzer will set the attenuator control appropriately to deliver the specified power at the source.

**1 - naManual** - Specify the attenuator setting using chan.[Attenuator](#) (which automatically sets AttenuatorMode = naManual).

**Return Type** NAModes

**Default** 0 - Auto

**Examples** `chan.AttenuatorMode(1) = naAuto 'Write`

`attn = chan.AttenuatorMode(1) 'Read`

**C++ Syntax** HRESULT get\_AttenuatorMode(long port, tagNAModes\* pVal)  
HRESULT put\_AttenuatorMode(long port, tagNAModes newVal)

**Interface** IChannel  
ICalSet3

## Attenuator Property

---

**Description** Sets or returns the value of the source attenuator for the specified port number. Sending this command automatically sets [AttenuatorMode](#) to Manual.

**VB Syntax** *object*.Attenuator(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*object* [Channel](#) (**object**)

or

[CalSet](#) (**object**) - Read-only property

*portNum* (**long integer**) - Port number of source attenuator to be changed.

*value* (**double**) - Attenuation value. The range of settable values depends on the PNA model. To determine the valid settings, do one of the following:

- See [PNA models and options](#) to see the range and step size for each model / option.
- To determine the maximum attenuator value use [MaximumSourceStepAttenuator](#) and . However, this will not tell you the attenuation step size.

If an invalid attenuation setting is entered, the PNA will select the next lower valid value. For example, if 19 is entered, then for an E8361A, 10 dB attenuation will be selected.

**Return Type** Double

**Default** 20 dB

**Examples** `chan.Attenuator(1) = 20 'Write`

`attn = chan.Attenuator(cnum) 'Read`

**C++ Syntax** HRESULT get\_Attenuator(long port, double \*pVal)  
HRESULT put\_Attenuator(long port, double newVal)

**Interface** IChannel  
ICalSet3

Last Modified:

25-Oct-2007    Edit value text

30-Apr-2007    Minor edits





## AutoIFBWAdjustment Property

---

**Description** Set and read auto IFBW adjustment ON | OFF state for Gain Compression measurements.

**VB Syntax** `gca.AutoIFBWAdjustment = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (**Boolean**) - Auto IFBW adjustment state. Choose from:

**False** - Sets auto IFBW adjustment OFF

**True** - Sets auto IFBW adjustment ON

**Return Type** Boolean

**Default** ON

**Examples** `gca.AutoIFBWAdjustment = True 'Write`

`aifbw = gca.AutoIFBWAdjustment 'Read`

**C++ Syntax** HRESULT get\_AutoIFBWAdjustment(VARIANT\_BOOL\* bState)

HRESULT put\_AutoIFBWAdjustment(VARIANT\_BOOL bState)

**Interface** IGainCompression

---

Last Modified:

8-Nov-2007    MX New topic

## AutoOrient Property

---

**Description** Sets ECAL module automatic orientation ON or OFF.

**VB Syntax** *obj.AutoOrient = bool*

**Variable** [\(Type\)](#) - Description

*obj* [SMCType](#) (object)

or

[VMCType](#) (object)

*bool* (Boolean)

**True** - Set AutoOrientation ON

**False** - Set AutoOrientation OFF

**Return Type** Boolean

**Default** True

**Examples** `Smc.AutoOrient = True`

**C++ Syntax** HRESULT put\_AutoOrient(VARIANT\_BOOL bAutoOrient);  
HRESULT get\_AutoOrient(VARIANT\_BOOL \*bAutoOrient);

**Interface** SMCType  
VMCType

## AutoPortExtConfig Property

---

**Description** Sets the frequency span that is used to calculate Automatic Port Extension. [Learn more about calculating Automatic Port Extension.](#)

**VB Syntax** *fixture*.AutoPortExtConfig = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (ENUM as NAAutoPortExtConfig)

**0 naAPEC\_CSPN** - Use current span.

**1 naAPEC\_AMKR** - Use active marker frequency.

**2 naAPEC\_USPN** - Use custom user span. Use [AutoPortExtSearchStart Property](#) and [AutoPortExtSearchStop Property](#) to specify start and stop frequency.

**Return Type** ENUM

**Default** **0 naAPEC\_CSPN**

**Examples** `fixture.AutoPortExtConfig = naAPEC_AMKR`

`value = fixture.AutoPortExtConfig 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtConfig(tagNAAutoPortExtConfig \*pVal );  
 HRESULT put\_AutoPortExtConfig(tagNAAutoPortExtConfig Val );

**Interface** IFixturing2

## AutoPortExtDCOffset Property

---

**Description** Specifies whether or not to include DC Offset as part of automatic port extension. Learn more about [Automatic DC Offset](#). Only allowed when [AutoPortExtLoss Property](#) is set to ON.

**VB Syntax** *fixture*.AutoPortExtDCOffset = *bool*

**Variable** [\(Type\)](#) - **Description**

*fixture* A [Fixturing](#) (object)

*bool* **True** - Includes DC Offset correction.

**False** - Does NOT include DC Offset correction.

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.AutoPortExtDCOffset = True`

`value = fixture.AutoPortExtDCOffset 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtDCOffset(VARIANT\_BOOL \*pState);  
HRESULT put\_AutoPortExtDCOffset(VARIANT\_BOOL bState);

**Interface** IFixturing2

## AutoPortExtLoss Property

---

**Description** Specifies whether or not to include loss correction as part of automatic port extension. [Learn more about Loss Compensation in port extension.](#)

**VB Syntax** *fixture*.AutoPortExtLoss = *bool*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*bool* **True** - Includes Loss correction.  
**False** - Does NOT include Loss correction.

**Return Type** Boolean

**Default** False

**Examples** `fixture.AutoPortExtLoss = True`

`value = fixture.AutoPortExtLoss 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtLoss(VARIANT\_BOOL \*pState);  
HRESULT put\_AutoPortExtLoss(VARIANT\_BOOL bState);

**Interface** IFixturing2

## AutoPortExtSearchStart Property

---

**Description** Set the start frequency for custom user span. Only applies when [fixture.AutoPortExtConfig](#) = 0 naAPEC\_CSPN.

[Learn more about User Span.](#)

**VB Syntax** *fixture.AutoPortExtSearchStart = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Double ) User span start value. Must be within the frequency range of the active channel and less than the value set by [AutoPortExtSearchStop Property](#)

**Return  
Type** Double

**Default** Start frequency of the current active channel.

**Examples** `fixture.AutoPortExtSearchStart = 1E9`

`value = fixture.AutoPortExtSearchStart 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtSearchStart(double \*pdVal );  
HRESULT put\_AutoPortExtSearchStart(double dVal);

**Interface** IFixturing2

## AutoPortExtSearchStop Property

---

**Description** Set the stop frequency for custom user span. Only applies when [fixture.AutoPortExtConfig](#) = 0 naAPEC\_CSPN.

[Learn more about User Span.](#)

Only applies when [fixture.AutoPortExtConfig](#) = 0 naAPEC\_CSPN

**VB Syntax** *fixture.AutoPortExtSearchStop* = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Double ) User span stop value. Must be within the frequency range of the active channel and greater than the value set by [AutoPortExtSearchStart Property](#)

**Return Type** Double

**Default** Stop frequency of the current active channel.

**Examples** `fixture.AutoPortExtSearchStop = 1E9`

`value = fixture.AutoPortExtSearchStop 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtSearchStop(double \*pdVal );  
HRESULT put\_AutoPortExtSearchStop(double dVal);

**Interface** IFixturing2



## AutoPortExtState Property

---

**Description** Enables and disables automatic port extensions on the specified port. All enabled ports will have their reference plane automatically adjusted after performing Automatic Port Extension.

**VB Syntax** *fixture.AutoPortExtState (port) = bool*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (**Integer**) Port number to enable or disable.

*bool* (Boolean)

**True** - Enables Auto Port Extensions

**False** - Disables Auto Port Extensions

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.AutoPortExtState(1) = True`

`value = fixture.AutoPortExtState(2) 'Read`

**C++ Syntax** HRESULT get\_AutoPortExtState(short port, VARIANT\_BOOL \*pState );  
HRESULT put\_AutoPortExtState(short port, VARIANT\_BOOL bVal);

**Interface** IFixturing2

## AuxiliaryTriggerCount Property

---

**Description** Returns the number of aux trigger input / output connector pairs in the PNA

**VB Syntax** *value* = *app*.AuxiliaryTriggerCount

**Variable** [\(Type\)](#) - Description

*value* **(Long Integer)** Variable to store the returned value.

2 = PNA-X models

1 = All other PNA models

*app* An [Application](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `ioConns = app.AuxiliaryTriggerCount`

**C++ Syntax** HRESULT AuxiliaryTriggerCount(long \*count);

**Interface** IApplication11

---

Last Modified:

6-Apr-2009 Replaced N5242A with PNA-X

14-Dec-2006 MX New topic

## AuxTriggerScopelsGlobal Property

---

**Description** Sets the Trigger OUT behavior to either Global or Channel. [Learn more about this setting.](#)

This command will cause the PNA to [Preset](#).

This setting remains until changed again using this command, or until the hard drive is changed or reformatted.

See the [AuxTrigger Object](#).

**VB Syntax** *pref.AuxTriggerScopelsGlobal = value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* (**Boolean**) - Choose from:

- **True** - Trigger properties apply to ALL channels (Global).
  - Default setting for **E836x** and **PNA-L** models.
  - Allows use of `command` to configure the external trigger properties.
  - "Per Point" trigger property is not settable. Use the channel's [Point trigger](#) setting.
- **False** - External Trigger properties apply to each channel independently.
  - Default setting for **PNA-X** models.
  - Must use [AuxTrigger](#) commands to configure the external trigger properties. [ExternalTriggerConnectionBehavior Property](#) will NOT work.
  - "Per Point" trigger output property is set using the channel's [Point trigger](#) setting **AND** [TriggerOutInterval Property](#).

**Return Type** Boolean

**Default** **True** - E836xB and PNA-L models  
**False** - PNA-X models

**Examples** `pref.AuxTriggerScopeIsGlobal = 1 'Write`

`auxTrigPref = pref.AuxTriggerScopeIsGlobal 'Read`

**C++ Syntax** `HRESULT get_AuxTriggerScopelsGlobal(VARIANT_BOOL * pref);`  
`HRESULT put_AuxTriggerScopelsGlobal(VARIANT_BOOL pref);`

Last modified:

25-Feb-2008 Clarification

Jan 3, 2007 MX New command

## AverageMode Property

---

**Description** Specifies the type of averaging to perform: Point or Sweep.

**VB Syntax** *chan.AverageMode = value*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*value* **(Enum as naAverageMode)** - Average Type. Choose from:

**0 - naPoint** Averaging measurements are made on each data point before stepping to the next data point. (Not available on 'C' models).

**1 - naSweep** Averaging measurements are made on subsequent sweeps until the required number of averaging sweeps are performed.

**Return Type** Enum

**Default** 1- naSweep

**Examples** `chan.AverageMode = naSweep 'Write`

`avgType = chan.AverageMode ' Read`

**C++ Syntax** HRESULT get\_AverageMode(NAAverageMode \* mode);  
HRESULT put\_AverageMode(NAAverageMode mode);

**Interface** IChannel16

---

Last Modified:

8-Jun-2009 Not available on 'C' models.

13-Oct-2008 MX New topic

## AveragingCount Property

---

**Description** Returns the number of sweeps that have been acquired and averaged into the measurements on this channel. [AveragingFactor](#) specifies the number of sweeps to average. AveragingCount indicates the progress toward that goal.

**VB Syntax** *value* = *chan*.**AveragingCount**

**Variable** [\(Type\)](#) - **Description**

*chan* A Channel (**object**)

*value* (**Long Integer**) - Variable to store the returned count

**Return Type** Long Integer

**Default** Not Applicable

**Example** `avgcount = chan.AveragingCount`

**C++ Syntax** HRESULT get\_AveragingCount(long\* count)

**Interface** IChannel

## AveragingFactor Property

---

**Description** Specifies the number of measurements to combine for an average. Must also turn averaging ON by setting *chan.Averaging* = 1.

**VB Syntax** *chan.AveragingFactor* = *value*

**Variable** [\(Type\)](#) - Description

*chan* A Channel (**object**)

*value* (**Long Integer**) - Number of measurement sweeps to average. Choose any number between **1** and **65536** (2<sup>16</sup>).

**Return Type** Long Integer

**Default** 1

**Examples**

```
chan.AveragingFactor = 5 'Write
```

```
avgfact = chan.AveragingFactor ' Read
```

**C++ Syntax** HRESULT get\_AveragingFactor(long \*pVal)  
HRESULT put\_AveragingFactor(long newVal)

**Interface** IChannel

---

Last Modified:

16-Apr-2009     Updated for Point Averaging

## Averaging Property

---

**Description** Turns trace averaging ON or OFF for all measurements on the channel. Averaging is only allowed on ratioed measurements; not on single input measurements.

**VB Syntax** *chan.Averaging = state*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*state* (boolean)

**False** - Turns averaging OFF

**True** - Turns averaging ON

**Return  
Type** Boolean

**Default** False

**Examples** `chan.Average = True 'Write`

`averg = chan.Averaging 'Read`

**C++ Syntax** HRESULT get\_Averaging(BOOL \*pVal)  
HRESULT put\_Averaging(BOOL newVal)

**Interface** IChannel



## AvoidSpurs Property

---

**Description** Sets and returns the state of the avoid spurs feature.

**VB Syntax** *mixer.AvoidSpurs = boolean*

**Variable** [\(Type\)](#) - Description

*mixer* A [Mixer](#) (object)

*value* **(Boolean)** - State of avoid spurs feature. Choose from

**False** Avoid spurs OFF

**True** Avoid spurs ON

**Return Type** Boolean

**Default** False

**Examples**

```
mixer.AvoidSpurs = True 'Write
```

```
variable = mixer.AvoidSpurs 'Read
```

**[C++ Syntax](#)** HRESULT get\_AvoidSpurs(Bool \*bVal)  
HRESULT put\_AvoidSpurs(Bool newVal)

**Interface** IMixer3

## BalancedMode Property

---

**Description** Sets and returns whether the balanced transform is ON or OFF

**VB Syntax** *balMeas.BalancedMode = value*

**Variable** [\(Type\)](#) - **Description**

*balMeas* A [BalancedMeasurement](#) (**object**)

*value* (**Boolean**) - State of balanced transform. Choose from

**False** Balanced Transform OFF

**True** Balanced Transform ON

**Return Type** Boolean

**Default** **False**

**Examples** `balMeas.BalancedMode = True 'Write`

`variable = balMeas.BalancedMode 'Read`

**C++ Syntax** HRESULT get\_BalancedMode(VARIANT\_BOOL \*bVal)  
HRESULT put\_BalancedMode(VARIANT\_BOOL newVal)

**Interface** IBalancedMeasurement

## BalPort1PhaseOffset Property

---

**Description** Sets and returns the phase offset between the two ports that comprise Balanced port 1. [balStim.Mode](#) must be set to a True Stimulus mode. Applicable only with [Opt 460 - iTMSA](#).

**VB Syntax** *balStim.BalPort1PhaseOffset = value*

**Variable** [\(Type\)](#) - **Description**

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Phase Offset in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples** `balStim.BalPort1PhaseOffset = 10 'Write`

`variable = balStim.BalPort1PhaseOffset 'Read`

**C++ Syntax** HRESULT get\_BalPort1PhaseOffset (double \*pVal)  
HRESULT put\_BalPort1PhaseOffset (double newVal)

**Interface** IBalancedStimulus

---

Last Modified:

15-May-2008    MX New topic

## BalPort1PowerOffset Property

---

Last Modified:

15-May-2008    MX New topic

## BalPort1StartPhase Property

---

**Description** Sets and returns the start phase of a phase sweep.

**VB Syntax** *balStim*.**BalPort1StartPhase** = *value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Start phase in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples** `balStim.BalPort1StartPhase = 10 'Write`

`variable = balStim.BalPort1StartPhase 'Read`

**C++ Syntax** HRESULT get\_BalPort1StartPhase (double \*pVal)  
HRESULT put\_BalPort1StartPhase (double newVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## BalPort1StopPhase Property

---

**Description** Sets and returns the stop phase of a phase sweep.

**VB Syntax** *balStim.BalPort1StopPhase = value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Stop phase in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples**

```
balStim.BalPort1StopPhase = 10 'Write
```

```
variable = balStim.BalPort1StopPhase 'Read
```

**C++ Syntax** HRESULT get\_BalPort1StopPhase (double \*pVal)  
HRESULT put\_BalPort1StopPhase (double newVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## BalPort2PhaseOffset Property

---

**Description** Sets and returns the phase offset between the two ports that comprise Balanced port 2. [balStim.Mode](#) must be set to a True Stimulus mode. Applicable only with [Opt 460 - iTMSA](#).

**VB Syntax** *balStim.BalPort2PhaseOffset = value*

**Variable** [\(Type\) - Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Phase Offset in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples** `balStim.BalPort2PhaseOffset = 10 'Write`

`variable = balStim.BalPort2PhaseOffset 'Read`

**C++ Syntax** HRESULT get\_BalPort2PhaseOffset (double \*pVal)  
HRESULT put\_BalPort2PhaseOffset (double newVal)

**Interface** IBalancedStimulus

---

Last Modified:

15-May-2008    MX New topic

## BalPort2PowerOffset Property

---

**Description** Sets and returns the power offset between the two ports that comprise Balanced port 2. [balStim.Mode](#) must be set to a True Stimulus mode. Applicable only with [Opt 460 - iTMSA](#).

**VB Syntax** *balStim.BalPort2PowerOffset = value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Power Offset in dB. Choose a value between .

**Return Type** Double

**Default** 0

**Examples** `balStim.BalPort2PowerOffset = 2 'Write`

`variable = balStim.BalPort2PowerOffset 'Read`

**C++ Syntax** HRESULT get\_BalPort2PowerOffset (double \*pVal)  
HRESULT put\_BalPort2PowerOffset (double newVal)

**Interface** IBalancedStimulus

---

Last Modified:

15-May-2008    MX New topic



## BalPort2StartPhase Property

---

**Description** Sets and returns the start phase of a phase sweep.

**VB Syntax** *balStim*.**BalPort2StartPhase** = *value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Start phase in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples** `balStim.BalPort2StartPhase = 10 'Write`

`variable = balStim.BalPort2StartPhase 'Read`

**C++ Syntax** HRESULT get\_BalPort2StartPhase (double \*pVal)  
HRESULT put\_BalPort2StartPhase (double newVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## BalPort2StopPhase Property

---

**Description** Sets and returns the stop phase of a phase sweep.

**VB Syntax** *balStim*.**BalPort2StopPhase** = *value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Double)** - Stop phase in degrees. Choose a value between -360 and 360.

**Return Type** Double

**Default** 0

**Examples**

```
balStim.BalPort2StopPhase = 10 'Write
```

```
variable = balStim.BalPort2StopPhase 'Read
```

**C++ Syntax** HRESULT get\_BalPort2StopPhase (double \*pVal)  
HRESULT put\_BalPort2StopPhase (double newVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## BandwidthTarget Property

---

**Description** Sets the insertion loss value at which the bandwidth of a filter is measured (using [BandwidthTracking](#) or [SearchFilterBandwidth](#)). For example, if you want to determine the filter bandwidth 3 db below the bandpass peak value, set BandwidthTarget to **-3**.

**VB Syntax** *meas.BandwidthTarget = value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**single**) - Target value. Choose any number between **-500** and **500**

**Return Type** Single

**Default** -3

**Examples** `meas.BandwidthTarget = -3 'Write`

`fbw = meas.BandwidthTarget 'Read`

**C++ Syntax** HRESULT put\_BandwidthTarget(float target)  
HRESULT get\_BandwidthTarget(float\* target)

**Interface** IMeasurement

## BandwidthTracking Property

---

**Description** Searches continually (every sweep) for the current [BandwidthTarget](#) (default is -3). To search the filter bandwidth for ONE SWEEP only (not continually), use `meas.SearchFilterBandwidth`.

This feature uses markers 1-4. To turn off these markers, either turn them off individually or [DeleteAllMarkers](#).

The bandwidth statistics are displayed on the analyzer screen. To get the bandwidth statistics, use either [GetFilterStatistics](#) or [FilterBW](#), [FilterCF](#), [FilterLoss](#), or [FilterQ](#).

The analyzer screen will show either Bandwidth statistics OR Trace statistics; not both.

To restrict the search to a [UserRange](#) with the bandwidth search, first activate marker 1 and set the desired UserRange. Then send the SearchFilterBandwidth command. The user range used with bandwidth search only applies to marker 1 searching for the max value. The other markers may fall outside the user range.

**VB Syntax** `meas.BandwidthTracking = value`

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*value* (boolean)  
**True** - Turns bandwidth tracking ON  
**False** - Turns bandwidth tracking OFF

**Return Type** Boolean

**Default** **False**

**Examples** `meas.BandwidthTracking = False 'Write`

`bwtrack = meas.BandwidthTracking 'Read`

**C++ Syntax** HRESULT put\_BandwidthTracking(VARIANT\_BOOL state)  
 HRESULT get\_BandwidthTracking(VARIANT\_BOOL\* state)

**Interface** IMeasurement

## BB\_BalPort1Negative Property

---

**Description** With a Balanced - Balanced topology, returns the PNA port number that is connected to the Negative side of the DUT's logical Port 1.

Use [SetBBPorts Method](#) to set the port mapping for a Balanced - Balanced topology.

**VB Syntax** `var = balTopology.BB_BalPort1Negative`

**Variable** [\(Type\)](#) - **Description**

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.BB_BalPort1Negative` 'Read

**C++ Syntax** HRESULT get\_BB\_BalPort1Negative(long \*bVal)

**Interface** IBalancedTopology

## BB\_BalPort1Positive Property

---

**Description** With a Balanced - Balanced topology, returns the PNA port number that is connected to the Positive side of the DUT's logical Port 1.

Use [SetBBPorts Method](#) to set the port mapping for a Balanced - Balanced topology.

**VB Syntax** `var = balTopology.BB_BalPort1Positive`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (object)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.BB_BalPort1Positive` 'Read

**C++ Syntax** HRESULT get\_BB\_BalPort1Positive(long \*bVal)

**Interface** IBalancedTopology

## BB\_BalPort2Negative Property

---

**Description** With a Balanced - Balanced topology, returns the PNA port number that is connected to the Negative side of the DUT's logical Port 2.

Use [SetBBPorts Method](#) to set the port mapping for a Balanced - Balanced topology.

**VB Syntax** `var = balTopology.BB_BalPort2Negative`

**Variable** [\(Type\)](#) - **Description**

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.BB_BalPort2Negative` 'Read

**C++ Syntax** HRESULT get\_BB\_BalPort2Negative(long \*bVal)

**Interface** IBalancedTopology

## BB\_BalPort2Positive Property

---

**Description** With a Balanced - Balanced topology, returns the PNA port number that is connected to the Positive side of the DUT's logical Port 2.

Use [SetBBPorts Method](#) to set the port mapping for a Balanced - Balanced topology.

**VB Syntax** `var = balTopology.BB_BalPort2Positive`

**Variable** [\(Type\)](#) - **Description**

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.BB_BalPort2Positive` 'Read

**C++ Syntax** HRESULT get\_BB\_BalPort2Positive(long \*bVal)

**Interface** IBalancedTopology



**BBalMeasurement Property**

---

**Description** Sets and returns the measurement for the Balanced - Balanced topology.

**VB Syntax** *balMeas.BBalMeasurement = value*

**Variable** [\(Type\)](#) - **Description**

*balMeas* A [BalancedMeasurement](#) (**object**)

*value* **(String)** - Balanced - Balanced Measurement parameter. Not case sensitive. Choose from:

Sdd11	Sdd12	Sdc11	Sdc12
Sdd21	Sdd22	Sdc21	Sdc22
Scd11	Scd12	Sc11	Sc12
Scd21	Scd22	Sc21	Sc22
lmb1	lmb2	CMRR -(Sdd21/Sc11)	

**Return Type** String

**Default** Sdd11

**Examples**

```
balMeas.BBalMeasurement = "Sdd11" 'Write
```

```
variable = balMeas.BBalMeasurement 'Read
```

**C++ Syntax** HRESULT get\_BBalMeasurement(BSTR \*pVal)  
HRESULT put\_BBalMeasurement(BSTR newVal)

**Interface** IBalancedMeasurement

## BeginResponse Property

---

**Description** When constructing a limit line, specifies the amplitude value of the start of a limit segment.

**VB Syntax** *limtseg.BeginResponse* = *value*

**Variable** [\(Type\)](#) - **Description**

*limtseg* A LimitSegment (**object**)

*value* (**double**) - Amplitude value. No units

**Return Type** Double

**Default** 0

**Examples**

```
Set limtseg = meas.LimitTest(1)
limtseg.BeginResponse = 10 'Write
```

```
BegResp = limtseg.BeginResponse 'Read
```

**C++ Syntax** HRESULT get\_BeginResponse(double \*pVal)  
HRESULT put\_BeginResponse(double newVal)

**Interface** ILimitSegment

## BeginStimulus Property

---

**Description** When constructing a limit line, specifies the beginning X-axis value.

**VB Syntax** *limtseg*.**BeginStimulus** = *value*

**Variable** [\(Type\)](#) - Description

*limtseg* A LimitSegment (**object**)

*value* (**double**) - Stimulus value. No units

**Return Type** Double

**Default** 0

**Examples**

```
Set limtseg = meas.LimitTest(1)
limtseg.Type = naLimitSegmentType_Maximum
limtseg.BeginStimulus = 3e9
limtseg.EndStimulus = 4e9
limtseg.BeginResponse = 10
limtseg.EndResponse = 10
```

```
BegStim = limtseg.BeginStimulus 'Read
```

**C++ Syntax** HRESULT get\_BeginStimulus(double \*pVal)  
HRESULT put\_BeginStimulus(double newVal)

**Interface** ILimitSegment

## BroadbandTuningSpan Property

---

**Description** Sets and returns the frequency span for the broadband tuning sweep.

**VB Syntax** *obj.BroadbandTuningSpan = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* **(Double)** Broadband frequency span in Hz.

**Return Type** **(Double)**

**Default** 3 MHz

**Examples** `embedLO.BroadbandTuningSpan = 1E6 'write`

`value = embedLO.BroadbandTuningSpan 'read`

**C++ Syntax** HRESULT get\_BroadbandTuningSpan(double\* span);  
HRESULT put\_BroadbandTuningSpan(double span);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## BucketNumber Property

---

**Description** Sets or returns the bucket number (data point) for the active marker. When the markers are [interpolated \(non-discrete\)](#), the returned value is the nearest marker bucket position.

**VB Syntax** *mark*.**BucketNumber** = *value*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**long integer**) - Data point. Choose any number between 0 and the measurement's number of data points - 1. For example, with Number of points = 201, choose between 0 and 200

**Return Type** Long Integer

**Default** The first marker is set to the middle of the span. Subsequent markers are set to the bucket number of the previously active marker.

**Examples** `mark.BucketNumber = 100 'moves the active marker to data point 100 -Write`  
`pointNumber = mark.BucketNumber 'returns the data point number of the`  
`marker object. When the markers are interpolated (non-discrete), the`  
`returned value is the nearest marker bucket position.`

**C++ Syntax** HRESULT get\_BucketNumber(long \*pVal)  
HRESULT put\_BucketNumber(long newVal)

**Interface** IMarker

## C0 Property

---

**Description** Sets and Returns the C0 (C-zero) value (the first capacitance value) for the calibration standard.  
To set the other capacitance values, use [C1](#), [C2](#), [C3](#)

**VB Syntax** *calstd.C0 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for C0 in femtofarads (1E-15)

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.C0 = 15 'Write the value of C0 to 15femtofarads`

`cap0 = calstd.C0 'Read the value of C0`

**C++ Syntax** HRESULT get\_C0(float \*pVal)  
HRESULT put\_C0(float newVal)

**Interface** ICalStandard

## C1 Property

---

**Description** Sets and Returns the C1 value (the second capacitance value) for the calibration standard.  
To set the other capacitance values, use [C0](#), [C2](#), [C3](#).

**VB Syntax** *calstd.C1 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for C1.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.C1 = 15 'Write the value of C1.`

`cap1 = calstd.C1 'Read the value of C1.`

**C++ Syntax** HRESULT get\_C1(float \*pVal)  
HRESULT put\_C1(float newVal)

**Interface** ICalStandard

## C2 Property

---

**Description** Sets and Returns the C2 value (the third capacitance value) for the calibration standard.  
To set the other capacitance values, use [C0](#), [C1](#), [C3](#).

**VB Syntax** *calstd.C2 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for C2.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.C2 = 15 'Write the value of C2.`

`cap2 = calstd.C2 'Read the value of C2`

**C++ Syntax** HRESULT get\_C2(float \*pVal)  
HRESULT put\_C2(float newVal)

**Interface** ICalStandard



## C3 Property

---

**Description** Sets and Returns the C3 value (the fourth capacitance value) for the calibration standard.  
To set the other capacitance values, use [C0](#), [C1](#), [C2](#)

**VB Syntax** *calstd.C3 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for C3.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.C3 = 15 'Write the value of C3.`

`cap3 = calstd.C3 'Read the value of C3`

**C++ Syntax** HRESULT get\_C3(float \*pVal)  
HRESULT put\_C3(float newVal)

**Interface** ICalStandard

## CalFactor Property

---

**Description** Sets or returns the cal factor value associated with a power sensor cal factor segment.

**VB Syntax** *calFactSeg.CalFactor = value*

**Variable** [\(Type\)](#) - **Description**

*calFactSeg* A [PowerSensorCalFactorSegment](#) (Object) or  
A [PowerSensorCalFactorSegmentPMAR](#) (Object)

*value* **(double)** – Cal factor in percent. Choose any value between 1 and 150

**Return Type** Double

**Default** 0

**Examples** `calFactSeg.CalFactor = 98 'Write`  
`factor = calFactSeg.CalFactor 'Read`

**C++ Syntax** HRESULT put\_CalFactor(Double newVal);  
HRESULT get\_CalFactor(Double \*pVal);

**Interface** IPowerSensorCalFactorSegment  
IPowerSensorCalFactorSegmentPMAR

---

Last Modified:

27-Aug-2009    MX New topic

**CalibrationType Property** **Superseded**

**Description** **Note:** This command has been replaced by [CalibrationTypeID\\_property](#), which provides selection of Calibration Type by string.

Specifies the type of calibration to perform or apply to the active S-Parameter measurement. This command determine the ports involved in the CalType by the ports being used by the active measurement.

For example:

- If the measurement is an S23, it uses ports 2 and 3.
- If the measurement is an S22 it will use the legacy load port to figure out which two ports form the caltype. The legacy load port is set using [CreateMeasurement](#).
- If **naCalType\_ThreePort\_SOLT** is specified on a 4-port PNA, an E\_NA\_DEPRECATED\_COMMAND error is returned. There is no way to determine the intended three ports.
- If **naCalType\_FourPort\_SOLT** is specified on a 4-port PNA, it is obvious that the ports involved are ports 1,2,3, and 4.

**Note:** For FCA measurements, use [CalibrationName](#) and [CalibrationTypeID](#).

**VB Syntax** *meas*.**CalibrationType** = *type*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*type* (**enum NACalType**) - Calibration type. Choose from:

- 0 - naCalType\_Response\_Open
- 1 - naCalType\_Response\_Short
- 2 - naCalType\_Response\_Thru
- 3 - naCalType\_Response\_Thru\_And\_Isol
- 4 - naCalType\_OnePort
- 5 - naCalType\_TwoPort\_SOLT
- 6 - naCalType\_TwoPort\_TRL
- 7 - naCalType\_None
- 8 - naCalType\_ThreePort\_SOLT
- 9 - Custom
- 10 - naCalType\_FourPort\_SOLT

**Return Type** **NACalType**

**Default** naCalType\_None

**Examples** `meas.CalibrationType = naCalType_Response_Open 'Write`

`meascal = meas.CalibrationType 'Read`

**C++ Syntax** HRESULT put\_CalibrationType (tagNACalType CalType)  
HRESULT get\_CalibrationType (tagNACalType\* pCalType)

**Interface** IMeasurement

Read only

## CalibrationName Property

---

**Description** Returns the name of the current Cal Type.

**VB Syntax** *value* = *meas*.**CalibrationName**

**Variable** [\(Type\)](#) - **Description**

*value* **(string)** - Variable to store the returned value.

*meas* A [Measurement](#) **(object)**

**Return Type** String

**Default** Not Applicable

**Examples** `ct = meas.CalibrationName`

**C++ Syntax** HRESULT get\_CalibrationName( BSTR\* CalibrationName);

**Interface** IMeasurement2

## CalibrationPort Property - **Obsolete**

---

**Description** **Note:** Beginning with Rev 6.0, this command is no longer necessary. [Learn more.](#) Because of improved calibration techniques, **Both** is always selected although a power meter measurement is performed only on port 1.

Specifies which SMC port to calibrate.

**VB Syntax** `SMC.CalibrationPort = value`

**Variable** [\(Type\)](#) - Description

`SMC` [SMCType](#) (object)

`value` (String) Port number to be calibrated. Choose from:

- **1** - SMC forward
- **2** - SMC reverse
- **Both**

**Return Type** String

**Default** 1

**Examples** `value = SMC.CalibrationPort = "Both"`

**C++ Syntax** `HRESULT put_CalibrationPort(BSTR port);`  
`HRESULT get_CalibrationPort(BSTR *port);`

**Interface** `SMCType`  
`VMCType`

## CalibrationTypeID Property

---

**Description** **Note:** This command replaces [Calibration Type Property](#).

Sets or returns the current cal type for the measurement using a Cal Type Name.

This command is used to set the Cal Type after recalling a Cal Set. [Learn more](#)

You can also use the CLSID or GUID associated with the Cal Type.

**VB Syntax** *meas*.**CalibrationTypeID** = *id*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*id* **(String)** Cal type. Case sensitive. Use one of the following:

### For Full Calibrations:

This command does not distinguish between TRL and SOLT. The same number of error terms is applied for both Cal Types.

"Full n Port(x,y,z...)"

where

n = the number of ports to calibrate

x,y,z = the port numbers to calibrate

For example:

```
"Full 7 Port (2,3,4,5,6,7,8) "
```

### For Response Calibrations:

"Response(param)" OR

"ResponseAndIsolation(param)"

Where param =

- S-parameter. For example"
  - `"Response (S21) "`
  - `"ResponseAndIsolation (A/R) "`
- Single or ratioed receivers using either [logical receiver notation](#) or physical receiver notation. For example:
  - `"Response (A) "`
  - `"ResponseAndIsolation (a3/b4) "`

### For FCA Calibrations:

- **"SMC\_2P"** (Response + Input + Output) All four sweeps required. Most accurate.
- **"SMCRsp+IN"** No Output match. All four sweeps required.
- **"SMCRsp+OUT"** No Output match. All four sweeps required.
- **"SMCRsp"** No Input or Output match. Saves two sweeps.

For VMC, multiple Cal types are not available.

### For Gain Compression Cal

where r = receive port; s = source port

- "GCA 2P (r,s)" - full 2-port cal
- "GCA Enh Resp (r,s)" - Enhanced Response Cal

**Return Type** String

**Default** Not Applicable

#### Examples

```
Dim pna
Dim m

Set pna = CreateObject("AgilentPNA835x.Application")
Set m = pna.ActiveMeasurement
m.CalibrationTypeID = "Scalar Mixer Cal"
m.ErrorCorrection = True
MsgBox m.CalibrationName
```

**C++ Syntax** HRESULT get\_CalibrationTypeID( BSTR\* CalibrationTypeID );  
HRESULT put\_CalibrationTypeID( BSTR CalibrationTypeID );

**Interface** IMeasurement2

---

Last modified:

22-Sep-2009	Removed VMC strings
27-May-2008	Edit channel vs meas
11-Feb-2008	Fixed typo
9/12/06	MQ Added for multiport.



## CalibrationFrequencies Property

---

**Description** Sets and returns the whether to perform the source power cal at the center frequencies midway between the main tones, or at all main tone frequencies.

**VB Syntax** *imd.CalibrationFrequencies* = *value*

**Variable** [\(Type\)](#) - **Description**

*imd* A [SweptIMDCal](#) (object)

*value* (Enum as NAIMDCalibrationFrequencies) Choose from:

**0 - naIMDCenterFrequencies** -Perform source power calibration at only the center frequencies midway between the main tones.

**1 - naIMDALLFrequencies** - Perform source power calibration at all main tone frequencies.

**Return Type** Enum

**Default** 0 - naIMDCenterFrequencies

**Examples** `imd.CalibrationFrequencies = naIMDALLFrequencies 'Write`

`calFreq = imd.CalibrationFrequencies 'Read`

**C++ Syntax** HRESULT get\_CalibrationFrequencies(tagNAIMDCalibrationFrequencies \* Val)  
HRESULT put\_CalibrationFrequencies(tagNAIMDCalibrationFrequencies newVal)

**Interface** ISweptIMD

---

Last Modified:

9-Sep-2008    MX New topic

## CalKitType Property

---

**Description** Sets and returns a calibration kit type for calibration or to be used for kit modification. To get a ha kit, use `app.ActiveCalKit` .

There is also a `CalKitType` property for use during a Guided, SMC, and VMC Calibration.

**VB Syntax** `object .CalKitType = value`

**Variable (Type) - Description**

*object* calkit (object) or  
Application (object)

**Note:** `app.CalKitType` and `calkit.calKitType` perform exactly the same function.

*value* (**enum naCalKit** ) - Calibration Kit type. Choose from:

1 - naCalKit\_User1  
2 - naCalKit\_User2  
3 - naCalKit\_User3  
4 - naCalKit\_User4  
..  
..  
..  
49 - naCalKit\_User49  
50 - naCalKit\_User50

These enumerated values correspond with the calibration kit ID on the Advanced Cal Kit Modify ( To change the cal kit name, use Name property .

**Return Type** NACalKit

**Default** Not Applicable

**Examples** `calkit.CalKitType = naCalKit_User27`

`kitype = app.CalKitType`

**C++ Syntax** `HRESULT get_CalKitType(tagNACalKit *pVal);`  
`HRESULT put_CalKitType(tagNACalKit newVal);`

**Interface** IApplication  
ICalKit

## CalKitType Property

---

**Description** Sets and returns a calibration kit type for the specified port number to be used during the calibration.

There is also a [CalKitType Property](#) for use during an Unguided Cal.

**Note:** Sliding loads are not fully supported from the GuidedCalibration object. The **Measure** button must be pressed manually on the PNA.

**VB Syntax** *object.CalKitType (port) = value*

**Variable** [\(Type\)](#) - Description

*object* Any of the following:

[GuidedCalibration](#) (object)

[SMCType](#) (object)

[VMCType](#) (object)

*port* (Long) Port number to which the cal kit will be assigned.

For All Guided Cals, select port number.

For SMC and VMC calibrations:

**1** - Input port of the mixer under test.

**2** - Output port of the mixer under test.

For VMC cal only:

**Output port of MUT +1** - Output port of the calibration mixer. Generally this is port 3.

[See the user interface \(UI\) equivalent of this command.](#)

*value* **(string)** - Calibration Kit type.

Use [CompatibleCalKits](#) property for a list of valid Cal Kits.

**Return Type** String

**Default** Not Applicable

**Examples** `SMC.CalKitType(1) = naCalKit_User27`

`value = smc.CalKitType(1)`

**C++ Syntax** HRESULT get\_CalKitType( long port, BSTR \*calkit)  
HRESULT put\_CalKitType( long port, BSTR calkit)

**Interface** IGuidedCalibration  
SMCType  
VMCType

---

Last Modified:

13-Aug-2007    Added detail for port argument

## CalMethod Property

---

**Description** Sets and returns the method for performing calibration on a noise channel.

**VB Syntax** *noise*.CalMethod = *value*

**Variable**    [\(Type\)](#) - Description

*noise*    A [NoiseCal](#) (object)

*value*    **(string)** Cal Method. Choose from:

- "VectorFull" or "Vector"
- "SParameter" (Not available for NFX measurements)
- "ScalarFull" or "Scalar"

**Return Type** String

**Default** "VectorFull"

**Examples** `noise.CalMethod = "VectorFull" 'Write`

`calMethod = noise.CalMethod 'Read`

**C++ Syntax** HRESULT get\_CalMethod(BSTR\* pValue)  
HRESULT put\_CalMethod(BSTR pNewValue)

**Interface** INoiseCal

---

Last Modified:

27-Oct-2009    Updated for NFX

29-May-2007    MN New topic

## CalMethod Property

---

**Description** Sets and returns the method by which the match-correction portion of an IMD calibration is performed. [Learn more.](#)

**VB Syntax** *imd.CalMethod* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMDCal](#) (object)

*value* (Enum as NAIMDCalMethod) Choose from:

**0 - naIMDMatchCorrectedResponse** -Performs a full 2-port cal for full match-correction.

**1 - naIMDResponseOnly** - Performs only a response cal instead of a full 2-port cal.

**Return  
Type** Enum

**Default** 0 - naIMDMatchCorrectedResponse

**Examples** `imd.CalMethod = naIMDMatchCorrectedResponse 'Write`

`calMeth = imd.CalMethod 'Read`

**C++ Syntax** HRESULT get\_CalMethod(tagNAIMDCalMethod \* Val)  
HRESULT put\_CalMethod(tagNAIMDCalMethod newVal)

**Interface** ISweptIMD

---

Last Modified:

18-Sep-2008    MX New topic

**CalPower Property** **Superseded**

---

**Description** This command is replaced by [SetCalInfoEx Method](#).  
Specifies the power level that is expected at the desired reference plane (DUT input or output). This is not used for [segment sweep with independent power levels](#) or [power sweeps](#).

**VB Syntax** *value* = *powerCalibrator*.**CalPower** (*chan*, *sourcePort*)

**Variable** [\(Type\)](#) - Description

*value* **(double)** - Variable to store the returned Cal power value in dBm.

*powerCalibrator* **(object)** - A SourcePowerCalibrator object

*chan* **(long integer)** - Channel number of the PNA.

*sourcePort* **(long integer)** - Source port number.

Use [GetPortNumber](#) to return the port number of a source that only has a string name, such as an [External Source](#).

**Return Type** None

**Default** 0

**Examples**

```
Set powerCalibrator = pna.SourcePowerCalibrator
powerCalibrator.CalPower = -10 'Write

power = powerCalibrator.CalPower 'Read
```

**C++ Syntax** HRESULT get\_CalPower(long channel, long sourcePort, double \*pVal);

**Interface** ISourcePowerCalibrator

---

Last Modified:

30-Apr-2007    Superseded

## Center Property

---

**Description** Sets or returns the Center time of either Gating or Time Domain transform windows

**VB Syntax** *object.Center* = *value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** As Gating  
or  
**(object)** As Transform

*value* **(double)** - Center time in seconds. Choose any number between:  
 $\pm (\text{points}-1) / \text{frequency span}$

**Return Type** Double

**Default** 0

**Examples**

```
trans.Center = 4.5e-9 'sets the Center time of a transform window
-Write
gate.Center = 4.5e-9 'sets the Center time of a gating window -
Write

cnt = trans.Center 'Read
```

**C++ Syntax** HRESULT get\_Center(double \*pVal)  
HRESULT put\_Center(double newVal)

**Interface** ITransform  
IGating



Read-only

## Center Property

---

**Description** Returns the stimulus value of the center data point for the measurement. This function does NOT work for segment sweep measurements. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**Center**

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** - Variable to store the returned value.

*meas* A Measurement **(object)**

**Return Type** Double

**Default** Not Applicable

**Examples** `Print meas.Center 'prints the center data point`

**C++ Syntax** HRESULT get\_Center(double \* Val);

**Interface** IMeasurement2

## CenterFrequency Property

---

**Description** Sets or returns the center frequency of the channel  
**or**  
Sets or returns the center frequency of the segment.  
See the [Measurement2 Interface](#) to learn how this method differs from [meas.Center](#).

**VB Syntax** *object.CenterFrequency = value*

**Variable** [\(Type\)](#) - Description

*object* A [Channel](#) (object)

**or**

A [Segment](#) (object)

*value* **(double)** - Center frequency in Hertz. Choose any number between the **minimum** and **maximum** frequencies of the analyzer.

**Return  
Type** Double

**Default** Center of the frequency range

**Examples** `chan.CenterFrequency = 4.5e9 'sets the center frequency of a linear sweep for the channel object -Write`

`centfreq = chan.CenterFrequency 'Read`

**C++ Syntax** HRESULT get\_CenterFrequency(double \*pVal)  
HRESULT put\_CenterFrequency(double newVal)

**Interface** IChannel  
ISegment

## ChannelNumber Property

---

**Description** Returns the Channel number of the Channel or Measurement object.

**VB Syntax** *object*.ChannelNumber

**Variable** [\(Type\)](#) - Description

*object* A Channel (**object**)  
**or**  
A Measurement (**object**)

**Return Type** Long Integer

**Default** Not applicable

**Examples**

```
chanNum = chan.ChannelNumber 'returns the channel number
chanNum = meas.ChannelNumber 'returns the channel number of the
measurement
```

**[C++ Syntax](#)** HRESULT get\_ChannelNumber(long \*pVal)

**Interface** IChannel  
IMeasurement

## CharacterizationNumber Property

---

**Description** Sets and reads the number to which the user characterization will be stored in the ECal module. The number must be set before sending [Initialize](#) or the default value (1) will be used.

**VB Syntax** *userChar.CharacterizationNumber = value*

**Variable** [\(Type\)](#) - Description

*userChar* An [ECalUserCharacterizer](#) Object

*value* (Long) User Characterization number. Choose a value between 1 and 12.

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `userChar.CharacterizationNumber = 5`

**C++ Syntax** HRESULT put\_CharacterizationNumber(long \*Number);

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## CharacterizeMixerOnly Property

---

**Description** Sets and returns whether to perform ONLY a mixer characterization.

**VB Syntax** `VMC.CharacterizeMixerOnly = bool`

**Variable** [\(Type\)](#) - Description

*VMC* [VMCType](#) (object)

*bool* (Boolean)

**True** - Perform ONLY mixer characterization.

**False** - Perform both mixer characterization and calibration.

**Return Type** Boolean

**Default** **False**

**Examples** `value = VMC.CharacterizeMixerOnly`

**C++ Syntax** `HRESULT put_CharacterizeMixerOnly(VARIANT_BOOL  
bCharMixerOnly);  
HRESULT get_CharacterizeMixerOnly(VARIANT_BOOL  
*bCharMixerOnly);`

**Interface** VMCType

## CharFileName Property

---

**Description** Specifies the mixer characterization (.S2P) file and immediately loads the file. Also specify the use of a characterization file with [LoadCharFromFile Property](#)

**VB Syntax** `VMC.CharFileName = value`

**Variable** [\(Type\)](#) - **Description**

*VMC* [VMCType](#) (object)

*value* (String) Full path, file name, and extension of the mixer characterization file.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `VMC.CharFileName = "C:/Program Files/Agilent/Network Analyzer/Documents/default.S2P"`

**C++ Syntax** `HRESULT put_CharFileName(BSTR filename);`  
`HRESULT get_CharFileName(BSTR *filename);`

**Interface** `VMCType`

## CharMixerReverse Property

---

<b>Description</b>	Specifies the direction in which to characterize the calibration mixer. <a href="#">Learn more about the calibration mixer.</a>
<b>VB Syntax</b>	<i>VMC.CharMixerReverse</i> = <i>bool</i>
<b>Variable</b>	<a href="#">(Type) - Description</a>
<i>VMC</i>	<a href="#">VMCType</a> (object)
<i>bool</i>	(Boolean)  <b>0</b> - Characterize the calibration mixer in the SAME direction as that specified in the mixer setup.  <b>1</b> - Characterize the calibration mixer in the REVERSE direction as that specified in the mixer setup.
<b>Return Type</b>	Boolean
<b>Default</b>	0
<b>Examples</b>	<code>VMC.CharMixerReverse = 0</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT put_CharMixerReverse(VARIANT_BOOL bcharReverse); HRESULT get_CharMixerReverse(VARIANT_BOOL *bcharReverse);
<b>Interface</b>	VMCType2

## CitiContents Property - Superseded

---

<b>Description</b>	This command is replaced with <a href="#">SaveData Method</a> Specifies the contents of subsequent citifile saves using app. <a href="#">SaveCitiDataData</a> or app. <a href="#">SaveCitiFormattedData</a>
<b>VB Syntax</b>	<i>pref.CitiContents</i> = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>pref</i>	A <a href="#">Preferences</a> (object)
<i>value</i>	(string) - Contents that will be saved with subsequent save commands. Choose from: "Single" - Single trace "Displayed" - All displayed traces "Auto" - All displayed traces
<b>Return Type</b>	String
<b>Default</b>	"Auto"
<b>Examples</b>	<pre>pref.CitiContents = "Single" 'Write  content = pref.CitiContents 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_CitiContents(BSTR *Contents) HRESULT put_CitiContents(BSTR Contents)
<b>Interface</b>	IPreferences



**CitiFormat Property - Superseded**

---

**Description** This command is replaced with [SaveData Method](#)Specifies the format of subsequent citifile saves using app.[SaveCitiFormattedData](#)**VB Syntax** *pref.CitiFormat = value***Variable** [\(Type\)](#) - Description*pref* A [Preferences](#) (object)*value* **(string)** - Format in which the citifile will be saved with subsequent save commands. Choose from:**"MA"** - Linear Magnitude / degrees**"DB"** - Log Mag / degrees**"RI"** - Real / Imaginary**"Auto"** - Format in which the trace is already displayed. If other than Log Mag, Linear Magnitude, or Real/Imag, then the format will be in Real/Imag.**Return  
Type** String**Default** "Auto"**Examples** `pref.CitiFormat = "MA" 'Write``format = pref.CitiFormat 'Read`**[C++ Syntax](#)** HRESULT get\_CitiFormat(BSTR \*Format)  
HRESULT put\_CitiFormat(BSTR Format)**Interface** IPreferences

## CmnModeZConvPortImag Property

---

**Description** Sets the imaginary part of the impedance value for the common port impedance conversion function.

**VB Syntax** *fixture.CmnModeZConvPortImag (portNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Imaginary part of the Impedance value. Choose a value between 0 and 1E18.

**Return  
Type** Double

**Default** 0

**Examples** `fixture.CmnModeZConvPortImag(2) = 75 'Write`

`value = fixture.CmnModeZConvPortImag(1) 'Read`

**C++ Syntax** HRESULT get\_CmnModeZConvPortImag( short portNum, double \*pVal)  
HRESULT put\_CmnModeZConvPortImag( short portNum, double newVal)

**Interface** IFixturing2

## CmnModeZConvPortReal Property

---

**Description** Sets the real part of the impedance value for the common port impedance conversion function.

**VB Syntax** *fixture.CmnModeZConvPortReal(portNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Real part of the Impedance value. Choose a value between 0 and 1E18.

**Return  
Type** Double

**Default** See [Common Mode Port Z Conversion Default](#)

**Examples** `fixture.CmnModeZConvPortReal(2) = 75 'Write`

`value = fixture.CmnModeZConvPortReal(1) 'Read`

**C++ Syntax** HRESULT get\_CmnModeZConvPortImag( short portNum, double \*pVal)  
HRESULT put\_CmnModeZConvPortImag( short portNum, double newVal)

**Interface** IFixturing2

## CmnModeZConvPortZ0 Property

---

**Description** Sets the impedance value for the common port impedance conversion function. Set either this single value or set the [real](#) and [imaginary](#) parts separately. The imaginary part is set to 0.0 using this command.

**VB Syntax** *fixture.CmnModeZConvPortZ0(portNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Impedance value. Choose a value between 0 and 1E7.

**Return Type** Double

**Default** See [Common Mode Port Z Conversion Default](#)

**Examples** `fixture.CmnModeZConvPortZ0(2) = 75 'Write`

`value = fixture.CmnModeZConvPortZ0(1) 'Read`

**C++ Syntax** HRESULT get\_CmnModeZConvPortImag( short portNum, double \*pVal)  
HRESULT put\_CmnModeZConvPortImag( short portNum, double newVal)

**Interface** IFixturing2

## CmnModeZConvState Property

---

**Description** Turns ON or OFF 4-port common port impedance conversion function. Must also set the fixture simulator function to ON using [FixturingState Property](#).

**VB Syntax** *fixture.CmnModeZConvState* = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns common port impedance conversion OFF

**True** - Turns common port impedance conversion ON

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.CmnModeZConvState = False 'Write`

`value = fixture.CmnModeZConvState 'Read`

**C++ Syntax** HRESULT get\_CmnModeZConvState( VARIANT\_BOOL \*pVal)  
HRESULT put\_CmnModeZConvState( VARIANT\_BOOL newVal)

**Interface** IFixturing2

## CompatibleCalKits Property

---

**Description** Returns a comma-separated list of valid kits that use the specified connector type. This includes mechanical cal kits, applicable characterizations found within ECal modules currently connected to the PNA, and all user characterizations stored in PNA disk memory.

**Note:** Beginning with PNA Rev 9.1, the serial number is returned for ALL ECal modules that are connected with the connector type of the specified port. Previously, the returned list would include the serial numbers to distinguish the ECal modules only when two or more identical ECal models were connected to the PNA.

**VB Syntax** *value* = *obj*.CompatibleCalKits (*port*)

**Variable** [\(Type\)](#) - **Description**

*value* (Variant) Variable to store the returned list of Cal Kits.

*obj* Any of the following:

[GuidedCalibration](#) (object) **Superseded** Replaced with [GetCompatibleCalKits Method](#)

[SMCType](#) (object)

[VMCType](#) (object)

*port* (Long) Port number for which you want compatible kits.

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
Dim kits As Variant
kits = MySMC.CompatibleCalKits(1)
```

**C++ Syntax** HRESULT get\_CompatibleCalKits(long port, VARIANT\* Kits);

**Interface** IGuidedCalibration  
SMCType  
VMCType

---

Last Modified:

16-Nov-2009 Added Note (9.1)

16-Nov-2009 Superseded

## CompositeNormalizationMode Property

---

**Description** Sets and returns the method by which CTB and CSO calculations are performed.

**VB Syntax** *imd.CompositeNormalizationMode = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Enum)

**0 - naNone** - the normalized power is not used in calculation

**1 - naNumberOfCarriers** - CTB and CSO is corrected by subtracting  $10 \cdot \log(N/2)$ , where

- N = # of carriers for CTB
- N = # of distortion products for CSO

**2 - naPdBm** - the composited normalized power for CTB or CSO is treated as a dBm value

**3 - naPdBmV** - the composited normalized power for CTB or CSO is treated as a dBmV value.

**Note:** Power values are stored using the currently-set units. Therefore, first set units with this command, then set power values using:

[CompositeNormalizedCSOPower](#)

[CompositeNormalizedCTBPower](#)

**Return Type** Enum

**Default** **naNumberOfCarriers**

**Examples** `imd.CompositeNormalizationMode = naNone 'Write`

`value = imd.CompositeNormalizationMode 'Read`

**C++ Syntax** HRESULT get\_CompositeNormalizationMode(tagNAIMDCompositeNormalizationMode \*pVal)  
HRESULT put\_CompositeNormalizationMode(tagNAIMDCompositeNormalizationMode pVal)

**Interface** ISweptIMD

---

Last Modified:

16-Sep-2008    MX New topic

## CompositeNormalizedCSOPower Property

---

**Description** Sets and returns the CSO Power for POWER [normalization mode](#). Valid only with measurement parameters: CSO2Lo and CSO2Hi and for Normalization Modes **dBm** and **dBmV**.

**VB Syntax** *imd.CompositeNormalizedCSOPower = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) Power level. The units are determined by [CompositeNormalizationMode Property](#), which must be set first.

**Return Type** Double

**Default** 0

**Examples** `imd.CompositeNormalizedCSOPower = -5 'Write`

`value = imd.CompositeNormalizedCSOPower 'Read`

**C++ Syntax** HRESULT get\_CompositeNormalizedCSOPower(double \*pVal)  
HRESULT put\_CompositeNormalizedCSOPower(double pVal)

**Interface** ISweptIMD

---

Last Modified:

16-Sep-2008    MX New topic



## CompositeNormalizedCTBPower Property

---

**Description** Sets and returns the CSO Power. Valid only with measurement parameters: CTBLo and CTBHi and for Normalization Modes **dBm** and **dBmV**.

**VB Syntax** *imd.CompositeNormalizedCTBPower* = *value*

**Variable**    [\(Type\)](#) - Description

*imd*    A [SweptIMD](#) Object

*value*    (Double) Power level. The units are determined by [CompositeNormalizationMode Property](#), which must be set first.

**Return  
Type**    Double

**Default**    0

**Examples**    `imd.CompositeNormalizedCTBPower = -5 'Write`

`value = imd.CompositeNormalizedCTBPower 'Read`

**C++ Syntax**    HRESULT get\_CompositeNormalizedCTBPower(double \*pVal)  
                  HRESULT put\_CompositeNormalizedCTBPower(double pVal)

**Interface**    ISweptIMD

---

Last Modified:

7-Apr-2009    Corrected parameters

16-Sep-2008    MX New topic

## CompressionAlgorithm Property

---

**Description** Set and read the algorithm method used to compute gain compression.

**VB Syntax** `gca.CompressionAlgorithm = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (tagNAGCACompressionAlgorithm) - Algorithm method. Choose from:

- naCompressionFromLinearGain (0)
- naCompressionFromMaximumGain (1)
- naBackoffCompression (2)
- naXYCompression (3)
- naSaturation (4)

**Return Type** Enum

**Default** naCompressionFromLinearGain (0)

**Examples** `gca.CompressionAlgorithm = naXYCompression 'Write`

```
compAlg = gca.CompressionAlgorithm 'Read
```

**C++ Syntax** HRESULT get\_CompressionAlgorithm(tagNAGCACompressionAlgorithm\* pVal)  
HRESULT put\_CompressionAlgorithm(tagNAGCACompressionAlgorithm newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## CompressionBackoff Property

---

**Description** Set and read value for the BackOff compression algorithm.

**VB Syntax** `gca.CompressionBackoff = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(Double)** Backoff value in dB. Choose from 30 to (-30)

**Return Type** Double

**Default** 10

**Examples** `gca.CompressionBackoff = 7 'Write`

`acqMode = gca.CompressionBackoff 'Read`

**C++ Syntax** HRESULT get\_CompressionBackoff(double\* pValue)  
HRESULT put\_CompressionBackoff(double newValue)

**Interface** IGainCompression

---

Last Modified:

21-Nov-2007    MX New topic

## CompressionDeltaX Property

---

**Description** Set and read the 'X' value in the delta X/Y compression algorithm.

**VB Syntax** `gca.CompressionDeltaX = value`

**Variable** [\(Type\)](#) - Description

`gca` A [GainCompression](#) (object)

`value` (**double**) X value in dB. Choose from 30 to (-30)

**Return Type** Double

**Default** 10

**Examples** `gca.CompressionDeltaX = 'Write`

```
xDelta = gca.CompressionDeltaX 'Read
```

**C++ Syntax** HRESULT get\_CompressionDeltaX(double\* pVal)  
HRESULT put\_CompressionDeltaX(double newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## CompressionDeltaY

---

**Description** Set and read the 'Y' value in the delta X/Y compression algorithm.

**VB Syntax** `gca.CompressionDeltaY = value`

**Variable** [\(Type\)](#) - Description

`gca` A [GainCompression](#) (object)

`value` (double) -

**Return Type** Double

**Default** 9

**Examples** `gca.CompressionDeltaY = 7 'Write`

`xDelta = gca.CompressionDeltaY 'Read`

**C++ Syntax** HRESULT get\_CompressionDeltaY(double\* pVal)  
HRESULT put\_CompressionDeltaY(double newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## CompressionInterpolation Property

---

**Description** Sets whether or not interpolation should be performed on 2D measured compression data. Applies ONLY to [2D acquisition modes](#).

**VB Syntax** `gca.CompressionInterpolation = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (**boolean**) - Choose from:

**True** Interpolate the results

**False** Do NOT interpolate the results but return the value closest to compression.

**Return  
Type** Boolean

**Default** False

**Examples** `gca.CompressionInterpolation = True 'Write`

`compInt = gca.CompressionInterpolation 'Read`

**C++ Syntax** HRESULT get\_CompressionInterpolation(VARIANT\_BOOL\* pVal)  
HRESULT put\_CompressionInterpolation(VARIANT\_BOOL newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## CompressionLevel Property

---

**Description** Set and read the desired gain reduction (from reference gain).  
This value is used for [Compression Methods](#): Compression from Linear Gain and Compression from Maximum Gain.

**VB Syntax** `gca.CompressionLevel = value`

**Variable** [\(Type\)](#) - Description

`gca` A [GainCompression](#) (object)

`value` **(Double)** - Compression level in dB. Choose a value greater than 0.1 dB.

**Return Type** Double

**Default** 1

**Examples**

```
gca.CompressionLevel = 1.5 'Write
```

```
compLevel = gca.CompressionLevel 'Read
```

**C++ Syntax** HRESULT get\_CompressionLevel(double\* pVal)  
HRESULT put\_CompressionLevel(double newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## CompressionLevel (Marker) Property

---

<b>Description</b>	Set and read the marker compression level. First use <a href="#">SearchCompressionPoint</a> to create the compression marker.
<b>VB Syntax</b>	<i>mkr</i> . <b>CompressionLevel</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>mkr</i>	A <a href="#">Marker</a> (object)
<i>value</i>	<b>(Double)</b> - Compression level in dB. Choose any number between: -500 dB to 500 dB Standard gain compression values are positive.
<b>Return Type</b>	Double
<b>Default</b>	1
<b>Examples</b>	<a href="#">See example program</a>
<b>C++ Syntax</b>	HRESULT get_CompressionLevel(double* pVal) HRESULT put_CompressionLevel(double newVal)
<b>Interface</b>	IMarker4

---

Last Modified:

8-Feb-2009    MX New topic



## CompressionPin Property

---

**Description** Reads the input power at the marker compression level. First issue [SearchCompressionPoint Method](#) or [Tracking Property](#).

**VB Syntax** *value* = *mkr*.**CompressionPin**

**Variable** [\(Type\)](#) - **Description**

*mkr* A [Marker](#) (object)

*value* **(Double)** - Variable to store the returned input power value.

**Return Type** Double

**Default** Not Applicable

**Examples** `compLevel=mkr.CompressionPin 'Read`

[See example program](#)

**C++ Syntax** HRESULT get\_CompressionPin(double\* pVal)

**Interface** IMarker4

---

Last Modified:

8-Feb-2009    MX New topic

## CompressionPout Property

---

**Description** Reads the output power at the marker compression level. First issue [SearchCompressionPoint Method](#) or [Tracking Property](#).

**VB Syntax** *value* = *mkr*.**CompressionPout**

**Variable** [\(Type\)](#) - **Description**

*mkr* A [Marker](#) (object)

*value* **(Double)** - Variable to store the returned output power value.

**Return Type** Double

**Default** Not Applicable

**Examples** `compLevel=mkr.CompressionPout 'Read`

[See example program](#)

**C++ Syntax** HRESULT get\_CompressionPout(double\* pVal)

**Interface** IMarker4

---

Last Modified:

8-Feb-2009    MX New topic

## ConnectorType Property

---

**Description** Sets or queries the connector type for the specified port.

**VB Syntax** *obj*.**ConnectorType** (*port*) = *value*

**Variable** [\(Type\) - Description](#)

*obj* Any of the following:

[GuidedCalibration](#) (object)

[SMCType](#) (object)

[VMCType](#) (object)

*port* (Long) Port number of the connector type.

*value* (String) - Connector type.

Use [ValidConnectorType Property](#) to list connector types.

**Return Type** String

**Default** None

**Examples**

```
Dim value As String
Value = MySMC.ConnectorType
```

**C++ Syntax** HRESULT get\_ConnectorType(long port, BSTR  
\*connector)  
  
HRESULT put\_ConnectorType(long port, BSTR  
connector)

**Interface** IGuidedCalibration  
  
SMCType  
  
VMCType

---

Last Modified:

13-Aug-2007    Added detail to port argument

## ConnectorType Property

---

**Description** Sets or queries the connector type for the specified port.

**VB Syntax** `ecalUser.ConnectorType (port) = value`

**Variable** [\(Type\) - Description](#)

*ecalUser* An [ECalUserCharacterizer](#) (object)

*port* (Enum) ECal port for which connector type is to be set. Choose from:

1 or **naECalPort\_A**

2 or **naECalPort\_B**

3 or **naECalPort\_C**

4 or **naECalPort\_D**

*value* (String) - Connector type.

When the User Characterization is to be stored in the ECal module, then the connector type is limited to a Factory-defined connector type. [See the list](#).

When the User Characterization is to be stored in PNA disk memory, then the connector type can also be a User-defined connector type.

**Return Type** String

**Default** "" (Empty String)

**Examples** `ecalUser.ConnectorType(naECalPort_B) = "APC 3.5 male" ' Write`

`Value = ecalUser.ConnectorType(naECalPort_B)`

**C++ Syntax** HRESULT get\_ConnectorType(NAECalPort port, BSTR \*connector);  
HRESULT put\_ConnectorType(NAECalPort port, BSTR connector);

**Interface** IECalUserCharacterizer

---

Last Modified:

18-Sep-2009 Modified for factory connectors (A.09.00)

5-Nov-2008 New topic (8.33)

## ControlLines Property

---

**Description** Sets the control lines of the specified test set. Control lines, provided through the front panel connector of a test set, are used to control external equipment such as a part handler. See your test set documentation to learn more about control lines.

**VB Syntax** *tset*.**ControlLines** (*chNum*) = *value*

**Variable** [\(Type\)](#) - Description

*tset* A [TestsetControl](#) object.

OR

An [E5091Testset](#) object.

*chNum* **(Integer)** Channel number of the measurement.

*value* **(Double)** Data value used to set control lines. Values are obtained by adding weights from the following table that correspond to individual lines.

Line	Weight
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128

- The E5091A interprets SENS:MULT1:OUTP 0 as all lines LOW.
- All "Z" and "H" series test sets interpret SENS:MULT1:OUTP 0 as all lines HIGH.

Refer to your test set documentation for setting control line values.

**Return Type** Variant

**Default** 0

**Examples** 'For a Z5623A K64 test set, the following sets line 3 and 4 OFF;  
all other lines ON.

`testset1.ControlLines(2) = 12`

[See E5091A Example Program](#)

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_ControlLines(long channelNum, VARIANT \*stateByte);  
HRESULT put\_ControlLines(long channelNum, VARIANT stateByte);

**Interface** ITestsetControl  
IE5091Testset

---

Last Modified:

17-Aug-2007    Added different test sets active High and low

## Count Property

---

**Description** Returns the number of items in a collection of objects.

**VB Syntax** *object*.Count

**Variable** [\(Type\)](#) - Description

*object* Any of the following **(objects)**:

- [CalFactorSegments Collection](#)
- [CalFactorSegmentsPMAR Collection](#)
- [Cal Sets Collection](#)
- [Channels Collection](#)
- [E5091Testset Collection](#)
- [ExternalDevices Collection](#)
- [ExternalTestsets Collection](#)
- [LimitTest Collection](#)
- [Measurements Collection](#)
- [NaWindows Collection](#)
- [PowerLossSegments Collection](#)
- [PowerLossSegmentsPMAR\\_Collection.htm](#)
- [PowerSensors Collection](#)
- [Segments Collection](#)
- [Traces Collection](#)
- [PowerMeterInterfaces Collection](#)

**Return Type** Long Integer

**Default** Not applicable

**Examples**

```
numofchans = chans.Count 'return the number of channels  
-Read
```

**C++ Syntax** HRESULT get\_Count(long \*pInterface)

**Interface** All listed above

---

Last Modified:

31-Jul-2009 Added ExternalDevices (9.0)

6-Mar-2009 PowerMeterInterfaces



## CouplePorts Property

---

**Description** Turns ON and OFF port power coupling. ON means the power level is the same for both ports. OFF means the power level may be set independently for each port.

**VB Syntax** *object.CouplePorts* = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)  
**or**  
CalSet (**object**) - Read-only property

*value* (**enum NASTates**) Choose from:  
**0 - NaOff** - Turns coupling OFF  
**1 - NaOn** - Turns coupling ON

**Return Type** Long Integer  
**1** - ON  
**0** - OFF

**Default** NaON (1)

**Examples** `chan.CouplePorts = NaOff 'Write`

`coulport = chan.CouplePorts 'Read`

**C++ Syntax** HRESULT get\_CouplePorts(tagNASTates \*pState)  
HRESULT put\_CouplePorts(tagNASTates newState)

**Interface** IChannel  
|CalSet3

## CoupleChannelParams Property

---

**Description** Turns ON and OFF Time Domain Trace Coupling. All of the measurements in the specified channel are coupled.

- To select Transform parameters to couple, use [Trans.CoupledParameters Property](#)
- To select Gating parameters to couple, use [Gate.CoupledParameters Property](#)

**VB Syntax** *chan.CoupleChannelParams = state*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*state* (boolean)  
**False** - Turns Trace Coupling OFF  
**True** - Turns Trace Coupling ON

**Return Type** Boolean

**Default** True

**Examples** `chan.CoupleChannelParams = False 'Write`

`couple = chan.CoupleChannelParams 'Read`

**C++ Syntax** HRESULT get\_CoupleChannelParams(VARIANT\_BOOL \*isCoupled);  
 HRESULT put\_CoupleChannelParams(VARIANT\_BOOL isCoupled);

**Interface** IChannel5

## Coupled Property

---

**Description** Sets and returns the state of coupling (ON or OFF) of this range to the primary range.

**VB Syntax** *FOMRange.Coupled* = *value*

**Variable** [\(Type\)](#) - Description

*object* An [FOMRange](#) (**object**)

*value* (**boolean**) - State of coupling.

**True** - Couple range to primary range.

**False** - Do NOT couple to primary range.

**Return Type** Boolean

**Default** True

**Examples** `fomRange.Coupled = False 'this range is NOT coupled to the primary range.`

`coup1 = fomRange.Coupled 'Read`

**[C++ Syntax](#)** HRESULT get\_Coupled(VARIANT\_BOOL \*pVal)  
HRESULT put\_Coupled(VARIANT\_BOOL pVal)

**Interface** IFOMRange

## CoupledMarkers Property

---

**Description** Sets and Reads the state of Coupled Markers (ON and OFF)

**VB Syntax** *app.CoupledMarkers = state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (boolean)  
**False (0)** - Turns Coupled Markers OFF  
**True (1)** - Turns Coupled Markers ON

**Return Type** Boolean  
**False** - OFF  
**True** - ON

**Default** **False**

**Examples** `app.CoupledMarkers = True 'Write`

`coupl = app.CoupledMarkers 'Read`

**C++ Syntax** HRESULT put\_CoupledMarkers(VARIANT\_BOOL bState)  
HRESULT get\_CoupledMarkers(VARIANT\_BOOL \*bState)

**Interface** IApplication

## CoupledParameters Property (Gating)

---

**Description** Specifies the time domain gating parameters to be coupled. The settings for those parameters will be copied from the active measurement to all other measurements on the channel.

To turn coupling ON and OFF, use [CoupleChannelParams Property](#)

To specify Transform parameters to couple, use [Transform.CoupledParameters Property](#)

**VB Syntax** *gate.CoupledParameters* = *value*

**Variable** [\(Type\)](#) - **Description**

*trans* A [Gating](#) (object)

*value* **(Enum As NAGatingCoupledParams)** - Parameters to couple. To specify more than one parameter, add the numbers. Choose from:

1 - naGatingStimulusCoupled (Start, Stop, Center, and Span TIME settings.)

2 - naGateStateCoupled (ON / OFF)

4 - naGatingShapeCoupled (Minimum, Normal, Wide, and Maximum)

8 - naGatingTypeCoupled (Bandpass and Notch)

**Return Type** Enum

**Default** 29

**Examples** `gate.CoupledParameters = 15 'Couple all parameters`

`CP = gate.CoupledParameters 'Read`

**C++ Syntax** HRESULT get\_CoupledParameters(long \*IParams);  
HRESULT put\_CoupledParameters(long IParams);

**Interface** IGating2

## CoupledParameters Property (Transform)

---

**Description** Specifies the time domain transform parameters to be coupled. The settings for those parameters will be copied from the active measurement to all other measurements on the channel.

To turn coupling ON and OFF, use [CoupleChannelParams Property](#)

To specify Gating parameters to couple, use [Gate.CoupledParameters Property](#)

**VB Syntax** *trans.CoupledParameters* = *value*

**Variable** [\(Type\)](#) - **Description**

*trans* A [Transform](#) (object)

*value* **(Enum As NATransformCoupledParams)** - Parameters to couple. To specify more than one parameter, add the numbers. Choose from:

1 - naTransformStimulusCoupled (Start, Stop, Center, and Span TIME settings.)

2 - naTransformStateCoupled (ON / OFF)

4 - naTransformWindowCoupled (Kaiser Beta / Impulse Width)

8 - naTransformModeCoupled (Low Pass Impulse, Low Pass Step, Band Pass)

16 - naTransformDistMkrUnitCoupled (Distance maker Units)

**Return Type** Enum

**Default** 29

**Examples** `trans.CoupledParameters = 31 'Couple all parameters`

`CP = trans.CoupledParameters 'Read`

**C++ Syntax** HRESULT get\_CoupledParameters(long \*IParams);  
HRESULT put\_CoupledParameters(long IParams);

**Interface** ITransform2

## CoupleTonePower Property

---

**Description** Sets and returns the ON | OFF state of power coupling for F1 and F2.

**VB Syntax** *object.CoupleTonePower = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*value* (Boolean) - Choose from:

**True** - F1 and F2 power is coupled.

**False** - F1 and F2 power is NOT coupled. Set power levels individually.

**Return Type** Boolean

**Default** True

**Examples** `ims.CoupleTonePower = true 'Write`

`value = ims.CoupleTonePower 'Read`

**C++ Syntax** HRESULT get\_CoupleTonePower(VARIANT\_BOOL\* val)  
HRESULT put\_CoupleTonePower(VARIANT\_BOOL val)

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## CSONumDistortionProducts Property

---

**Description** Sets and returns the “N = number of distortion products” value for the calculation of the CSO parameter. [Learn more.](#)

**VB Syntax** *imd.CSONumDistortionProducts* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) Number of distortion products

**Return  
Type** Double

**Default** 40

**Examples** `imd.CSONumDistortionProducts = True 'Write`

`value = imd.CSONumDistortionProducts 'Read`

**C++ Syntax** HRESULT get\_CSONumDistortionProducts(double \*pVal)  
HRESULT put\_CSONumDistortionProducts(double \*pVal)

**Interface** ISweptIMD

---

Last Modified:

25-Aug-2008    MX New topic



## CSOffset Property

---

**Description** Sets and returns the offset that is applied to CSO measurements. Valid only with measurement parameters: CSO2Lo and CSO2Hi.

**VB Syntax** *imd.CSOffset* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) Offset value in dBm.

**Return  
Type** Double

**Default** 0

**Examples** `imd.CSOffset = 2 'Write`

`value = imd.CSOffset 'Read`

**C++ Syntax** HRESULT get\_CSOffset(double \*pVal)  
HRESULT put\_CSOffset(double pVal)

**Interface** ISweptIMD

---

Last Modified:

16-Sep-2008    MX New topic

## CTBOffset Property

---

**Description** Sets and returns the offset that is applied to CTB measurements. Valid only with measurement parameters: CTB, CTBLo, CTBHi, CTBE, CTBELo, and CTBEHi.

**VB Syntax** *imd.CTBOffset* = *value*

**Variable**    [\(Type\)](#) - Description

*imd*    A [SweptIMD](#) Object

*value*    (Double) Offset value in dBm.

**Return  
Type**    Double

**Default**    0

**Examples**    `imd.CTBOffset = 2 'Write`

`value = imd.CTBOffset 'Read`

**[C++ Syntax](#)**    HRESULT get\_CTBOffset(double \*pVal)  
                  HRESULT put\_CTBOffset(double pVal)

**Interface**    ISweptIMD

---

Last Modified:

7-Apr-2009    Added valid parameters

16-Sep-2008    MX New topic

## CTBXMODNumCarriers Property

---

**Description** Sets the “N = Total number of carriers” value used in the calculation of the XMOD and CTB parameter. [Learn more.](#)

**VB Syntax** *imd.CTBXMODNumCarriers* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Long Integer) Number of carriers.

**Return Type** Long Integer

**Default** 40

**Examples** `imd.CTBXMODNumCarriers = 15 'Write`

`value = imd.CTBXMODNumCarriers 'Read`

**C++ Syntax** HRESULT get\_CTBXMODNumCarriers(double \*pVal)  
HRESULT put\_CTBXMODNumCarriers(double \*pVal)

**Interface** ISweptIMD

---

Last Modified:

25-Aug-2008    MX New topic

## CustomChannelConfiguration Property

---

**Description** Returns a handle to the custom application object on the active channel. You can either **(1)** use the handle directly to access measurement properties and methods, or **(2)** set a variable to the measurement object. The variable retains a handle to the original measurement.

Currently, the custom application objects to which this property provides access are:

- [NoiseFigure Object](#)
- [GainCompression Object](#)
- [SweptIMD Object](#)

**VB Syntax** 1) *set custChan = chan.CustomChannelConfiguration.* <setting>  
**or**  
 2) *set custChan = app.ActiveChannel.CustomChannelConfiguration*

**Variable** [\(Type\)](#) - **Description**

*custChan* A variable in which the handle to a custom application is returned. **(object)**

*chan* A [Channel](#) **(object)**

<setting> A property or method setting on the custom application object.

**Return Type** Custom application object

**Default** None

**Examples** See examples: [NoiseFigure](#) [GainCompression](#) [SweptIMD](#)

**C++ Syntax** HRESULT CustomChannelConfiguration(IDispatch\*\* value);

**Interface** IChannel12

---

Last Modified:

27-Aug-2008 Added Swept IMD

16-Oct-2007 MX New topic

## CWFrequency Property

---

**Description** Set the Continuous Wave (CW) frequency. Must first send `chan.SweepType = naCWTimeSweep`

**VB Syntax** `object.CWFrequency = value`

**Variable** [\(Type\)](#) - Description

*object* One of the following:

- [Channel](#) (object)
- [FOMRange](#) (object) Range must be [UNCOUPLED](#).
- [CalSet](#) (object) - Read-only property

See also [Measurement2](#) interface.

*value* **(double)** CW frequency. Choose any number between:  
the **minimum** and **maximum** frequency limits of the analyzer  
Units are Hz

**Return Type** Double

**Default** 1e9

**Examples** `chan.CWFrequency = 5e9 'Write`

```
cwfreq = chan.CWFrequency 'Read
```

**C++ Syntax** HRESULT put\_CWFrequency(double newVal)  
HRESULT get\_CWFrequency(double \*pVal)

**Interface** IChannel  
|CalSet3  
IFOMRange

## Data Property

---

**Description** Reads the next specified number of data points from the FIFO buffer. The data is returned in 32 bit real/imaginary pair. Data is cleared as it is read.

**Note:** This method is the slowest way to transfer data using COM. However, it is supported by all COM client programming languages. For better performance, try using [DataInCompactForm](#).

**VB Syntax** `array = fifo.Data (count)`

**Variable** [\(Type\)](#) - **Description**

`array` (Variant) Variable to store the returned data.

`fifo` A [FIFO](#) Object

`count` (Long Integer) Number of data points to read.

**Return Type** Variant array. Each VARIANT is typed as a 4-byte floating point number.

**Default** Not Applicable

**Examples** `value = fifo.Data(500) 'Read`

**C++ Syntax** HRESULT get\_Data(long count,VARIANT \* data);

**Interface** IFIFO

---

Last Modified:

7-Oct-2008    MX New topic

## DataCount Property

---

**Description** Returns the total number of data points in the FIFO buffer.

**VB Syntax** *value* = *fifo*.**DataCount**

**Variable** [\(Type\)](#) - **Description**

*value* (Long Integer) Variable to store the returned number of data points.

*fifo* A [FIFO](#) Object

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `value = fifo.DataCount 'Read`

**[C++ Syntax](#)** HRESULT get\_DataCount(long \*value)

**Interface** IFIFO

---

Last Modified:

7-Oct-2008    MX New topic

## DataInCompactForm Property

---

**Description** Reads FIFO data the same as [Data Property](#) but returns the data in a more compact form of SAFEARRAY. This is significantly faster but it is not supported in all client environments.

**VB Syntax** `array = fifo.DataInCompactForm (count)`

**Variable** [\(Type\)](#) - [Description](#)

*array* (Variant) Variable to store the returned data

*fifo* A [FIFO](#) Object

*count* (Long Integer) Number of data points to read.

**Return Type** Returns an array of 4 byte floating point numbers.

**Default** Not Applicable

**Examples** `value = fifo.DataInCompactForm(500) 'Read`

**C++ Syntax** HRESULT get\_DataInCompactForm(long count,VARIANT \* data);

**Interface** IFIFO

---

Last Modified:

3-Jun-2010    MX New topic



## Delay Property

---

**Description** Sets and Returns the electrical delay value for the calibration standard.

**VB Syntax** *calstd.Delay (n) = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*n* Pulse Generator number.

*value* (**single**) - Electrical delay in picoseconds

**Return Type** Single

**Default** Not Applicable

**Exaamples** `calstd.Delay = 12 'Write 12ps Delay`

`stdDelay = calstd.Delay 'Read the value of Delay`

**C++ Syntax** HRESULT get\_Delay(float \*pVal)  
HRESULT put\_Delay(float newVal)

**Interface** ICalStandard

---

Last Modified:

5-Jan-2007    MX New topic

## Delay (pulse) Property

---

**Description** Sets the pulse delay - the amount of time before a new pulse begins.

**VB Syntax** *pulse.Delay(n) = value*

**Variable** [\(Type\)](#) - Description

*pulse* A [PulseGenerator](#) (object)

*n* **(Integer)** Pulse generator number. Choose from 0 to 4.  
0 is the generator that pulses the ADC.

*value* **(Double)** Delay value in seconds. Choose a value from about 33ns to about 70 seconds.

**Return Type** Double

**Default** 0

**Examples**

```
pulse.Delay(1) = 1ms 'Write
```

```
value = pulse.Delay(4) 'Read
```

**C++ Syntax** HRESULT get\_Delay(integer pulse, double\* delay);  
HRESULT put\_Delay(integer pulse, double delay);

**Interface** IPulseGenerator

---

Last Modified:

1-Jan-2007    MX New topic

## Delay Property

---

**Description** Specifies the delay that should be applied by the PNA after the aux trigger input is received and before the acquisition is made.

**Note:** Use on PNA-X ONLY. Other models do NOT have an Aux Input.

**VB Syntax** *auxTrig.Delay = value*

**Variable** [\(Type\)](#) - Description

*auxTrig* An [AuxTrigger](#) (object)

*value* (**double**) - Delay value in seconds. Choose a value between 0 and 3.0 seconds.

**Return  
Type** Double

**Default** Not Applicable

**Examples** `auxTrig.Delay = 1.2 'Write 1.2s Delay`

`value = auxTrig.Delay 'Read the value`

**C++ Syntax** HRESULT get\_Delay(double \*val);  
HRESULT put\_Delay(double val);

**Interface** IAuxTrigger

---

Last Modified:

5-Sep-2008 Added Note

14-Dec-2006 MX New topic

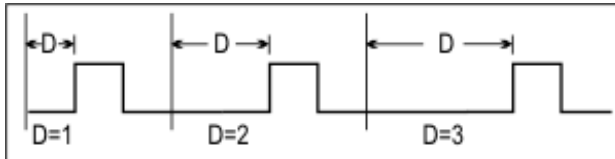
## DelayIncrement Property

---

**Description** Sets the pulse delay increment. The delay increments with each pulse by the <value> amount. For example, in this diagram the delay starts as 1. On the second pulse, delay=2. On the third pulse, delay=3.

**Important:** If **D + W** is greater than **P**, then undefined PNA behavior results. There is NO error message or warning. Delay includes the incremented value.

This is useful for pulse profiling.



**VB Syntax** `pulse.DelayIncrement(n) = value`

**Variable** [\(Type\)](#) - Description

*pulse* A [PulseGenerator](#) (object)

*n* **(Integer)** Pulse generator number. Choose from 0 to 4.  
0 is the generator that pulses the ADC.

*value* **(Double)** Delay increment value in seconds.

**Return Type** Double

**Default** 0

**Examples** `pulse.DelayIncrement(1) = 1ms 'Write`

`value = pulse.DelayIncrement(4) 'Read`

**C++ Syntax** `HRESULT get_DelayIncrement(integer pulse, double* dIncr);`  
`HRESULT put_DelayIncrement(integer pulse, double dIncr);`

**Interface** IPulseGenerator

Last Modified:

18-Jun-2007 MX New topic

## DeltaFrequency Property

---

**Description** Sets and returns the fixed tone spacing value. Use with IMD sweep types:

- `naIMDToneCWSweep`
- `naIMDTonePowerSweep`
- `naIMDToneCenterFreqSweep`

**VB Syntax** `object.DeltaFrequency = value`

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IISpectrum](#) Object

*value* (Double) - Tone spacing frequency in Hz. Both the F1 and F2 tones MUST be within the frequency range of the PNA.

**Return Type** Double

**Default** 1 MHz

**Examples**

```
imd.DeltaFrequency = 1e6 'Write
```

```
value = imd.DeltaFrequency 'Read
```

**C++ Syntax** HRESULT get\_DeltaFrequency(double \*pVal)  
HRESULT put\_DeltaFrequency(double newVal)

**Interface** ISweptIMD  
IISpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## DeltaFrequencyStart Property

---

**Description** Sets and returns the starting main tone separation for [sweep type](#)= naIMDDeltaFrequencySweep

**VB Syntax** *imd.DeltaFrequencyStart* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) - Starting tone separation between F1 and F2 in Hz. Both F1 and F2 tones MUST be within the frequency range of the PNA where:

F1 (start) =  $\text{imd.FrequencyCenter} - \text{imd.DeltaFrequencyStart} / 2$

F2 (start) =  $\text{imd.FrequencyCenter} + \text{imd.DeltaFrequencyStart} / 2$

**Return Type** Double

**Default** 1 MHz

**Examples** `imd.DeltaFrequencyStart = 5e6 'Write`

`value = imd.DeltaFrequencyStart 'Read`

**C++ Syntax** HRESULT get\_DeltaFrequencyStart(double \*pVal)  
HRESULT put\_DeltaFrequencyStart(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## DeltaFrequencyStop Property

---

**Description** Sets and returns the stop spacing of the main tones. Use with [sweep type](#)=  
naIMDDeltaFrequencySweep

**VB Syntax** *imd.DeltaFrequencyStop = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) - Stopping tone separation between F1 and F2 in Hz. Both F1 and F2 tones MUST be within the frequency range of the PNA where:

$F1 \text{ (stop)} = imd.FrequencyCenter - imd.DeltaFrequencyStop / 2$

$F2 \text{ (stop)} = imd.FrequencyCenter + imd.DeltaFrequencyStop / 2$

**Return Type** Double

**Default** 10 MHz

**Examples** `imd.DeltaFrequencyStop = 20e6 'Write`

`value = imd.DeltaFrequencyStop 'Read`

**C++ Syntax** HRESULT get\_DeltaFrequencyStop(double \*pVal)  
HRESULT put\_DeltaFrequencyStop(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## DeltaMarker Property

---

**Description** Sets a marker as a delta marker. The reference marker must already be turned ON. See [meas.ReferenceMarkerState](#)

**VB Syntax** *mark*.**DeltaMarker** = *state*

**Variable** [\(Type\)](#) - **Description**

*app* A Marker (**object**)

*state* (**boolean**) -

**True** - marker is a delta marker

**False** - marker is NOT a delta marker

**Return  
Type** Boolean

**Default** **False**

**Examples** `mark.DeltaMarker = True 'Write`

`delta = mark.DeltaMarker 'Read`

**C++ Syntax** HRESULT get\_DeltaMarker(VARIANT\_BOOL bState)  
HRESULT put\_DeltaMarker(VARIANT\_BOOL \*bState)

**Interface** IMarker



## Description Property

---

**Description** Sets or returns the descriptive string assigned to the Cal Set. Change this string so that you can easily identify each Cal Set constructed.

**VB Syntax** *CalSet*.**Description** = *value*

**Variable** [\(Type\)](#) - Description

*CalSet* **(object)** - A [Cal Set object](#)

*value* **(string)** – Description of the Cal Set

**Return Type** String

**Default** "CalSet\_n" where n is an integer number.

**Examples**

```
CalSet.Description = "My Cal Set" 'Write  
desc = CalSet.Description 'Read
```

**C++ Syntax** HRESULT get\_Description(BSTR \*pVal)  
HRESULT put\_Description(BSTR newVal);

**Interface** ICalSet

---

Last modified:

Dec.12, 2006    MX New topic

## DescriptiveText Property

---

**Description** Write and read descriptive text associated with the configuration. This text is displayed in the path configuration dialog. Text is generally used to describe external connections that must be made manually to complete the configuration setup.

**VB Syntax** *pathConfig*.**DescriptiveText** = *text*

**Variable** [\(Type\)](#) - Description

*name* **(String)** Variable to store the returned configuration name.

*pathConfig* A [PathConfiguration](#) **(object)**

*text* **(String)** Descriptive text enclosed in quotes.

**Return Type** String

**Default** Not Applicable

**Examples** `pathConf.DescriptiveText "here are the instructions for connecting the device for this configuration"`

**C++ Syntax** `HRESULT get_DescriptionText(BSTR* pConnectionText );`  
`HRESULT put_DescriptionText(BSTR connectionText );`

**Interface** IPathConfiguration

---

Last Modified:

14-Dec-2006    MX New topic

## DeviceInputPort Property

---

**Description** Returns the PNA port number to be used for the DUT input.

**VB Syntax** *port* = *obj*.**DeviceInputPort**

**Variable** [\(Type\)](#) - **Description**

*port* **(long)** Variable to store the PNA port number of the DUT input.

*mixer* A [Mixer](#) **(object)**

**Return Type** Long

**Default** 1

**Examples** `value = mixer.DeviceInputPort`

**C++ Syntax** HRESULT get\_DeviceInputPort(long \*pVal)

**Interface** IMixer8

---

Last Modified:

23-Apr-2008    New topic.

## DeviceInputPort Property

---

**Description** Read the PNA port number which is connected to the DUT input.

Use [SetPortMap Method](#) to change the port mapping.

**VB Syntax** *obj*.DeviceInputPort

**Variable** [\(Type\)](#) - Description

*obj* A [GainCompression](#) (object) or

A [SweptIMD](#) (object) or

An [IMSpectrum](#) (object)

A [NoiseFigure](#) (object)

**Return Type** Integer

**Default** 1

**Examples**

```
inPort = gca.DeviceInputPort 'Read
```

**C++ Syntax** HRESULT get\_DeviceInputPort(int\* pVal)

**Interface** IGainCompression

ISweptIMD

IMSpectrum

INoiseFigure6

---

Last Modified:

30-Apr-2010 Added NF

11-Aug-2009 Added IMD and IMS (9.0)

11-Sep-2007 MX New topic

## DeviceOutputPort Property

---

**Description** Returns the PNA port number to be used for the DUT output.

**VB Syntax** *port* = *mixer*.DeviceOutputPort

**Variable** [\(Type\)](#) - Description

*port* **(long)** Variable to store the PNA port number of the DUT output.

*mixer* A [Mixer](#) (object)

**Return Type** Long

**Default** 2

**Examples** `value = mixer.DeviceOutputPort`

**C++ Syntax** HRESULT get\_DeviceOutputPort(long \*pVal)

**Interface** IMixer8

---

Last Modified:

23-Apr-2008    New topic.

## InputLinearPowerLevel Property

---

**Description** Set and read the input power at which Linear Gain and all S-parameters are measured.

**VB Syntax** `gca.InputLinearPowerLevel = value`

**Variable** [\(Type\)](#) - **Description**

*gca* A [GainCompression](#) (object)

*value* (**double**) Linear input power level in dBm. Choose a value from +30 to (-30).

**Return Type** Double

**Default** - 25 dBm

**Examples**

```
gca.InputLinearPowerLevel = -10 'Write
```

```
LinPwr = gca.InputLinearPowerLevel 'Read
```

**C++ Syntax** HRESULT get\_InputLinearPowerLevel(double\* pVal)  
HRESULT put\_InputLinearPowerLevel(double newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## DeviceOutputPort Property

---

**Description** Read the PNA port number which is connected to the DUT Output.

Use [SetPortMap Method](#) to change the port mapping.

**VB Syntax** *obj*.DeviceOutputPort

**Variable** [\(Type\)](#) - Description

*obj* A [GainCompression](#) (object) or

A [SweptIMD](#) (object) or

An [IMSpectrum](#) (object)

A [NoiseFigure](#) (object)

**Return Type** Integer

**Default** 2

**Examples** `outPort = gca.DeviceOutputPort 'Read`

**C++ Syntax** HRESULT get\_DeviceOutputPort(int\* pVal)

**Interface** IGainCompression

ISweptIMD

IMSpectrum

INoiseFigure6

---

Last Modified:

30-Apr-2010 Added NF

11-Aug-2009 Added IMD and IMS

11-Sep-2007 MX New topic

## DiffPortMatch\_C Property

---

**Description** Sets the Capacitance value of the differential matching circuit.

**VB Syntax** *fixture.DiffPortMatch\_C (portNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1,2 ,3. [Learn more about logical ports.](#)

*value* **(Double)** Capacitance value in farads. Choose a value between **-1E18** to **1E18**.

**Return Type** Double

**Default** 0

**Examples** `fixture.DiffPortMatch_C(2) = 1e-6 'Write`

`value = fixture.DiffPortMatch_C(1) 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatch\_C( short portNum, double \*pVal)  
HRESULT put\_DiffPortMatch\_C( short portNum, double newVal)

**Interface** IFixturing2



## DiffPortMatch\_G Property

---

**Description** Sets the Conductance value of the differential matching circuit.

**VB Syntax** *fixture*.DiffPortMatch\_G(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Conductance value in siemens. Choose a value between **-1E18** to **1E18**.

**Return  
Type** Double

**Default** 0

**Examples** `fixture.DiffPortMatch_G(2) = 1e-3 'Write`

`value = fixture.DiffPortMatch_G(1) 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatch\_G( short portNum, double \*pVal)  
HRESULT put\_DiffPortMatch\_G( short portNum, double newVal)

**Interface** IFixturing2

## DiffPortMatch\_L Property

---

**Description** Sets the Inductance value of the differential matching circuit.

**VB Syntax** *fixture.DiffPortMatch\_L(portNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Inductance value in henries. Choose a value between **-1E18** to **1E18**.

**Return  
Type** Double

**Default** 0

**Examples** `fixture.DiffPortMatch_L(2) = 1e-3 'Write`

`value = fixture.DiffPortMatch_L(1) 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatch\_L( short portNum, double \*pVal)  
HRESULT put\_DiffPortMatch\_L( short portNum, double newVal)

**Interface** IFixturing2

## DiffPortMatch\_R Property

---

**Description** Sets the Resistance value of the differential matching circuit.

**VB Syntax** *fixture*.DiffPortMatch\_R(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Resistance value in ohms. Choose a value between **-1E18** to **1E18**.

**Return  
Type** Double

**Default** 0

**Examples**

```
fixture.DiffPortMatch_R(2) = 1e3 'Write
```

```
value = fixture.DiffPortMatch_R(1) 'Read
```

**C++ Syntax** HRESULT get\_DiffPortMatch\_R(short portNum, double \*pVal)  
HRESULT put\_DiffPortMatch\_R(short portNum, double newVal)

**Interface** IFixturing2

## DiffPortMatchMode Property

---

**Description** Sets the differential matching circuit type. To select a user-defined circuit, specify IN ADVANCE the 2-port touchstone filename with [DiffPortMatch\\_UserFilename Property](#). If you do not specify the appropriate file and you select USER, an error occurs and naNO\_CIRCUIT is automatically selected.

**VB Syntax** *fixture.DiffPortMatchMode(pNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*pNum* **(Integer)** Balanced (logical) port number. Choose from logical ports **1, 2, or 3**. [Learn more about logical ports.](#)

*value* **(Enum as NADiffPortMatchCircuitMode)** Choose from:  
**0** or **naSHUNT\_L\_SHUNT\_C\_CIRCUIT** - Specifies the circuit that consists of shunt L and shunt C.  
**1** or **naUSER\_FILE\_CIRCUIT** - Specifies the user-defined circuit.  
**2** or **naNO\_CIRCUIT** - Specifies no-circuit.

**Return Type** Enum

**Default** naSHUNT\_L\_SHUNT\_C\_CIRCUIT

**Examples** `fixture.DiffPortMatchMode(2) = naNO_CIRCUIT 'Write`

`value = fixture.DiffPortMatchMode(1) 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatchMode( short port, tagNADiffPortMatchCircuitMode \*eVal)  
 HRESULT put\_DiffPortMatchMode( short port, tagNADiffPortMatchCircuitMode eVal)

**Interface** IFixturing2

## DiffPortMatchState Property

---

**Description** Turns ON or OFF 4-port differential port matching function. Must also set the fixture simulator function to ON using [FixturingState Property](#).

**VB Syntax** *fixture*.DiffPortMatchState = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns differential port matching OFF

**True** - Turns differential port matching ON

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.DiffPortMatchState = False 'Write`

`value = fixture.DiffPortMatchState 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatchState( VARIANT\_BOOL \*pVal)  
HRESULT put\_DiffPortMatchState( VARIANT\_BOOL newVal)

**Interface** IFixturing2

## DiffPortMatchUserFilename Property

---

**Description** Specifies the 2-port touchstone file in which the information on the user-defined differential matching circuit is saved. Following this command, send [DiffPortMatchCircuit Property](#). If the specified file does not exist, an error occurs when you set the type of differential matching circuit to USER.

**VB Syntax** *fixture.DiffPortMatchUserFilename(pNum) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*pNum* **(Integer)** Balanced (logical) port number. Choose from logical ports **1, 2** or **3**. [Learn more about logical ports.](#)

*value* **(String)** Full path, file name, and extension (.s2P) of the de-embedding circuit. Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents

**Return Type** String

**Default** Not Applicable

**Examples** `fixture.DiffPortMatchUserFilename(2) = "C:/Program Files/Agilent/Network Analyzer/Documents/myFile.s4p" 'Write`

`value = fixture.DiffPortMatchUserFilename(1) 'Read`

**C++ Syntax** HRESULT get\_DiffPortMatchUserFilename( short port, BSTR \*bstrFile)  
HRESULT put\_DiffPortMatchUserFilename( short port, BSTR bstrFile)

**Interface** IFixturing2

## DiffZConvPortImag Property

---

**Description** Sets the imaginary part of the impedance value for the differential port impedance conversion function.

**VB Syntax** *fixture*.DiffZConvPortImag(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Imaginary part of the Impedance value. Choose a value between 0 and 1E18

**Return  
Type** Double

**Default** 0

**Examples** `fixture.DiffZConvPortImag(2) = 75 'Write`

`value = fixture.DiffZConvPortImag(1) 'Read`

**C++ Syntax** HRESULT get\_DiffZConvPortImag( short portNum, double \*pVal)  
HRESULT put\_DiffZConvPortImag( short portNum, double newVal)

**Interface** IFixturing2

## DiffZConvPortReal Property

---

**Description** Sets the imaginary part of the impedance value for the differential port impedance conversion function.

**VB Syntax** *fixture*.DiffZConvPortReal(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Real part of the Impedance value. Choose a value between 0 and 1E18

**Return  
Type** Double

**Default** See [Differential Port Z Conversion Default](#)

**Examples** `fixture.DiffZConvPortReal(2) = 75 'Write`

`value = fixture.DiffZConvPortReal(1) 'Read`

**C++ Syntax** HRESULT get\_DiffZConvPortReal( short portNum, double \*pVal)  
HRESULT put\_DiffZConvPortReal( short portNum, double newVal)

**Interface** IFixturing2



## DiffZConvPortZ0 Property

---

**Description** Sets the impedance value for the differential port impedance conversion function. Set either this single value or set the [real](#) and [imaginary](#) parts separately. The imaginary part is set to 0.0 using this command.

**VB Syntax** `fixture.DiffZConvPortZ0(portNum) = value`

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Balanced (logical) port number. Choose from logical ports 1, 2, or 3. [Learn more about logical ports.](#)

*value* **(Double)** Impedance value. Choose a value between 0 and 1E18

**Return Type** Double

**Default** See [Differential Port Z Conversion Default](#)

**Examples** `fixture.DiffZConvPortZ0(2) = 75 'Write`

`value = fixture.DiffZConvPortZ0(1) 'Read`

**C++ Syntax** HRESULT get\_DiffZConvPortZ0( short portNum, double \*pVal)  
HRESULT put\_DiffZConvPortZ0( short portNum, double newVal)

**Interface** IFixturing2

## DiffZConvState Property

---

**Description** Turns ON or OFF 4-port differential impedance conversion function. Must also set the fixture simulator function to ON using [FixturingState Property](#).

**VB Syntax** *fixture*.DiffZConvState = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns differential impedance conversion OFF.

**True** - Turns differential impedance conversion ON.

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.DiffZConvState = False 'Write`

`value = fixture.DiffZConvState 'Read`

**C++ Syntax** HRESULT get\_DiffZConvState( VARIANT\_BOOL \*pVal)  
HRESULT put\_DiffZConvState( VARIANT\_BOOL newVal)

**Interface** IFixturing2

## Format Property

---

**Description** Sets or returns the display format of the measurement.

**VB Syntax** *meas.Format* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**enum NADataFormat**) - Choose from:

- 0 - naDataFormat\_LinMag
- 1 - naDataFormat\_LogMag
- 2 - naDataFormat\_Phase
- 3 - naDataFormat\_Polar
- 4 - naDataFormat\_Smith
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Celsius

**Return Type** Long Integer

**Default** 1 - naDataFormat\_LogMag

**Examples** `meas.Format = naDataFormat_Real 'Write`

`fmt = meas.Format 'Read`

**C++ Syntax** HRESULT get\_Format(tagDataFormat \*pVal)  
HRESULT put\_Format(tagDataFormat newVal)

**Interface** IMeasurement

---

Last Modified:

1-Oct-2007    Added temperature formats



## DisplayAutomationErrors Property

---

**Description** Enables or disables automation error messages from being displayed on the screen.

**VB Syntax** `app.DisplayAutomationErrors = value`

**Variable** [\(Type\)](#) - [Description](#)

*app* An [Application](#) (object)

*value* (Boolean)

**True** allows error to show on display,

**False** turns error off from display.

**Return Type** Boolean

**Default** True

**Examples**

```
Dim app As Application
Set app = New Application
app.DisplayAutomationErrors = False    'Turns off display
print app.DisplayAutomationErrors     'prints False
```

**C++ Syntax** HRESULT get\_DisplayAutomationErrors(VARIANT\_BOOL \* Val);  
HRESULT put\_DisplayAutomationErrors(VARIANT\_BOOL Val);

**Interface** IApplication2

## DisplayGlobalPassFail Property

---

**Description** Shows or hides the dialog which displays global pass/fail results. [Learn more about Global Pass/Fail.](#)

**VB Syntax** `app.DisplayGlobalPassFail = value`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (Boolean)

**True** - displays the pass/fail dialog.

**False** - hides the pass/fail dialog.

**Return Type** Boolean

**Default** False

**Examples**

```
Dim app As Application
Set app = New Application
app.DisplayGlobalPassFail = true 'shows dialog
```

**C++ Syntax** HRESULT get\_DisplayGlobalPassFail(VARIANT\_BOOL \* Val);  
HRESULT put\_DisplayGlobalPassFail(VARIANT\_BOOL Val);

**Interface** IApplication6

## DisplayRange Property

---

**Description** Sets or returns the range to be displayed on the PNA x-axis. All traces in the channel have this same x-axis scaling.

**VB Syntax** *FOM.DisplayRange = value*

**Variable** [\(Type\)](#) - Description

*object* An [FOM](#) (**object**)

*value* (**string**) - Range to be displayed on the PNA x-axis. Case insensitive.

**Return Type** String

**Default** "Receivers"

**Examples** `fom.DisplayRange = "Source" 'sets the x-axis to the frequency range of "source"`

`disprange = fom.DisplayRange 'Read`

**C++ Syntax** HRESULT get\_DisplayRange(BSTR \*pDspRange)  
HRESULT put\_DisplayRange(BSTR pDspRange)

**Interface** IFOM

---

Last Modified:

7-Mar-2007    Changed to receivers

## Distance Property

---

**Description** Set or query marker distance on a time domain trace.

The Write command moves the marker to the specified distance value. Once moved, you can read the [Y axis](#) value or [read the X-axis time](#) value. (Distance is calculated from the X-axis time value.)

The Read command reads the distance of the marker.

If the marker is set as delta, the WRITE and READ data is relative to the reference marker.

**VB Syntax** *mark.Distance = value*

**Variable** [\(Type\)](#) - Description

*mark* A [Marker](#) (object)

*value* (**double**) - Marker distance in the unit of measure specified with [DistanceMarkerUnit Property](#)

**Return Type** Double

**Default** Not Applicable

**Examples** `mark.Distance = 3e9 'Write`

`XVal = mark.Distance 'Read`

**C++ Syntax** HRESULT get\_Distance(double \*pVal);  
HRESULT put\_Distance(double newVal);

**Interface** IMarker2



## DistanceMarkerMode Property

---

**Description** Specifies the measurement type in order to determine the correct marker distance.

- Select Auto for S-Parameter measurements.
- Select Reflection or Transmission for arbitrary ratio or unratioed measurements.

This settings affects the display of ALL markers for only the ACTIVE measurement.

**VB Syntax** *trans.DistanceMarkerMode = value*

**Variable** [\(Type\)](#) - Description

*trans* A [Transform](#) (object)

*value* (enum As NADistanceMarkerMode) - Choose from:

- 0 - naDistanceMarkerModeAuto
- 1 - naDistanceMarkerModeReflection
- 2 - naDistanceMarkerModeTransmission

**Return Type** Enum

**Default** 0 - naDistanceMarkerModeAuto

**Examples**

```
trans.DistanceMarkerMode = naDistanceMarkerModeReflection 'Write
```

```
DMM = trans.DistanceMarkerMode 'Read
```

**C++ Syntax** HRESULT get\_DistanceMarkerMode(tagNADistanceMarkerMode \*pVal);  
 HRESULT put\_DistanceMarkerMode(tagNADistanceMarkerMode newVal);

**Interface** ITransform2

## DistanceMarkerUnit Property

---

**Description** Specifies the unit of measure for the display of marker distance values. This settings affects the display of ALL markers for only the ACTIVE measurement (unless Distance Maker Units are coupled using [CoupledParameters Property](#)).

**VB Syntax** *trans.DistanceMarkerUnit = value*

**Variable** [\(Type\)](#) - [Description](#)

*trans* A [Transform](#) (object)

*value* **(Enum As NADistanceMarkerUnit)** - Distance Marker Units. Choose from

0 - naDistanceMarkerUnitMeter

1 - naDistanceMarkerUnitFeet

2 - naDistanceMarkerUnitInch

**Return Type** Enum

**Default** 0 - naDistanceMarkerUnitMeter

**Examples** `trans.DistanceMarkerUnit = naDistanceMarkerUnitFeet 'sets the`

`U = trans.DistanceMarkerUnit 'Read`

**C++ Syntax** HRESULT get\_DistanceMarkerUnit(tagNADistanceMarkerUnit \*pVal);  
 HRESULT put\_DistanceMarkerUnit(tagNADistanceMarkerUnit newVal);

**Interface** ITransform2

## Divisor Property

---

**Description** Sets and returns the Divisor value to be used when coupling this range to the primary range.

This setting is valid only if the specified range is [coupled](#) to the primary range.

**VB Syntax** *FOMRange.Divisor = value*

**Variable** [\(Type\)](#) - Description

*object* An [FOMRange](#) (object)

*value* **(Double)** - Divisor value.-(Unitless)

**Return Type** Double

**Default** 0

**Examples** `fomRange.Divisor = .5 'Write`

`Div = fomRange.Divisor 'Read`

**C++ Syntax** HRESULT get\_Divisor(double \*pVal)  
HRESULT put\_Divisor(double \*pVal)

**Interface** IFOMRange

## Do1PortEcal Property

---

**Description** Specify ECAL or Mechanical calibration for the mixer characterization portion of a VMC calibration.

**VB Syntax** `VMC.Do1PortEcal = bool`

**Variable** [\(Type\) - Description](#)

*VMC* [VMCType](#) (object)

*bool* (Boolean)

**True** - ECAL

**False** - Mechanical

**Return Type** Boolean

**Default** **False**

**Examples** `value = VMC.Do1PortEcal`

**C++ Syntax** `HRESULT put_Do1PortEcal(VARIANT_BOOL bDoEcal);`  
`HRESULT get_Do1PortEcal(VARIANT_BOOL *bDoEcal);`

**Interface** VMCType

## Do2PortEcal Property

---

**Description** Specify ECAL or Mechanical calibration. For VMC, this selection only applies to the 2-port calibration portion. For mixer characterization (VMC), use [Do1PortEcal Property](#)

**VB Syntax** *object.Do2PortEcal = bool*

**Variable** [\(Type\) - Description](#)

*object* [SMCType](#) (object) or  
[VMCType](#) (object)

*bool* (Boolean)

**True** - ECAL

**False** - Mechanical

**Return Type** Boolean

**Default** **False**

**Examples** `value = VMC.Do2PortEcal`

**C++ Syntax** HRESULT put\_Do2PortEcal(VARIANT\_BOOL bDoEcal);  
HRESULT get\_Do2PortEcal(VARIANT\_BOOL \*bDoEcal);

**Interface** SMCType  
VMCType

## Domain Property

---

**Description** Returns the domain (frequency,time, power) of the measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**Domain**

**Variable** [\(Type\)](#) - **Description**

*value* (**Enum** as NADomainType) - variable to store the returned value  
0 - Frequency  
1 - Time  
2 - Power

*meas* A Measurement **(object)**

**Return Type** Enum as NADomainType

**Default** Not Applicable

**Examples** `Print meas.Domain 'prints the value of the domain enum`

**C++ Syntax** HRESULT get\_Domain(tagNADomainType \* Val);

**Interface** IMeasurement2

## DUTTopology Property

---

**Description** Returns the device topology setting.

**VB Syntax** *balTopology*.**DUTTopology** = *value*

**Variable** [\(Type\)](#) - [Description](#)

*balTopology* A [BalancedTopology](#) (object)

*value* (enum NADUTTopology) - Choose either:

0 naSEBal: Single-Ended - Balanced measurement

1 naSESEBal: Single-Ended - Single-Ended - Balanced measurement

2 naBalBal: Balanced - Balanced measurement

**Return Type** Enum as NADUTTopology

**Default** naSEBal

**Examples** `balTop.DUTTopology = naSESEBal 'Write`

`DutTop = balTop.DUTTopology 'Read`

**C++ Syntax** HRESULT get\_DUTTopology(tagNADUTTopology\* pVal)  
HRESULT put\_DUTTopology(tagNADUTTopology newVal)

**Interface** IBalancedTopology

## DwellTime Property

---

**Description** Sets or returns the dwell time at the start of each sweep point for all measurements in a channel. Dwell time is only available with Chan.[SweepGenerationMode](#) = **naSteppedSweep** (not **naAnalogSweep**).

Sets or returns the dwell time of a specified sweep segment.

**VB Syntax** *object.DwellTime* = *value*

**Variable** [\(Type\)](#) - Description

*object* A Channel (object)  
**or** CalSet (object) - Read-only property  
**or** Segment (object)

*value* **(double)** - Dwell Time in seconds. Choose any number between **0** and **86400**

**Return Type** Double

**Default** 0

**Examples** `chan.DwellTime = 3e-3 'sets the dwell time for the channel -Write`

`segs(3).DwellTime = 1e9 'sets the dwell time of segment 3 -Write`

`dwell = chan.DwellTime 'Read`

**C++ Syntax** HRESULT get\_DwellTime(double \*pVal)  
 HRESULT put\_DwellTime(double newVal)

**Interface** IChannel  
 ISegment  
 ICalSet3



## ECALCharacterization Property

---

<b>Description</b>	Specifies the characterization data within an ECal module to be used for the SMC calibration. Learn more about <a href="#">ECal User Characterization</a> .
<b>VB Syntax</b>	<i>SMC.ECALCharacterization(mod) = value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>SMC</i>	<a href="#">SMCType</a> (object)
<i>module</i>	1- ECal module
<i>value</i>	<p><b>(Long)</b> – Characterization data within the ECal module to be used for ECal operations. Choose from:</p> <ul style="list-style-type: none"> <li><b>0</b> – Factory Characterization</li> <li><b>1</b> – UserCharacterization1</li> <li><b>2</b> – UserCharacterization2</li> <li><b>3</b> – UserCharacterization3</li> <li><b>4</b> – UserCharacterization4</li> <li><b>5</b> – UserCharacterization5</li> </ul>
<b>Return Type</b>	Long
<b>Default</b>	0 - Factory Characterization
<b>Examples</b>	<code>SMC.ECALCharacterization(1) = 2</code>
<b>C++ Syntax</b>	HRESULT put_ECALCharacterization( long moduleNumber, long characterization); HRESULT get_ECALCharacterization( long moduleNumber, long* characterization);
<b>Interface</b>	ICalibrator2

## ECALCharacterization Property

---

**Description** Specifies the characterization data within an ECal module to be used, and the portion of the VMC calibration.

Learn more about [ECal User Characterization](#).

**VB Syntax** `VMC.ECALCharacterization (module,port) = value`

**Variable** [\(Type\)](#) - Description

*VMC* [VMCType](#) (object)

*module* **(long integer)**  
1 - ECAL module

*port* **(boolean)**  
**True** - 2-port calibration portion of the VMC  
**False** - 1-port (mixer characterization portion of the VMC cal

*value* **(Long)** – Characterization data within the ECal module to be used for ECal operations. Choose from:  
0 – Factory Characterization  
1 – UserCharacterization1  
2 – UserCharacterization2  
3 – UserCharacterization3  
4 – UserCharacterization4  
5 – UserCharacterization5

**Return Type** Long

**Default** 0 - Factory Characterization

**Examples** `VMC.ECALCharacterization (1,True) = 4`

**C++ Syntax** `HRESULT put_ECALCharacterization( long moduleNumber, long characterization);`  
`HRESULT get_ECALCharacterization( long moduleNumber, long* characterization);`

**Interface** ICalibrator2

## ECALCharacterizationEx Property

---

<b>Description</b>	<p>This property replaces <a href="#">ECALCharacterization Property</a>.</p> <p>Specifies the characterization data within an ECal module to be used for the calibration.</p> <p>Learn more about <a href="#">ECal User Characterization</a>.</p>
<b>VB Syntax</b>	<i>cal</i> . <b>ECALCharacterizationEx</b> ( <i>module</i> ) = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>cal</i>	<a href="#">Calibrator</a> (object)
<i>module</i>	<p><b>(long integer)</b> Optional argument. ECal module.</p> <p>Choose from modules <b>1</b> through <b>8</b></p> <p>Use <a href="#">IsECALModuleFoundEx</a> to determine the number of modules connected to the PNA</p> <p>Use <a href="#">GetECALModuleInfoEx</a> to returns the model and serial number of each module.</p>
<i>value</i>	<p><b>(Long)</b> – Characterization data within the ECal module to be used for ECal operations.</p> <p>Choose from:</p> <ul style="list-style-type: none"> <li><b>0</b> – Factory Characterization</li> <li><b>1</b> – UserCharacterization1</li> <li><b>2</b> – UserCharacterization2</li> <li><b>3</b> – UserCharacterization3</li> <li><b>4</b> – UserCharacterization4</li> <li><b>5</b> – UserCharacterization5</li> </ul>
<b>Return Type</b>	Long
<b>Default</b>	0 - Factory Characterization
<b>Examples</b>	<code>cal.ECALCharacterizationEx (4) = 2</code>
<b>C++ Syntax</b>	<p>HRESULT put_ECALCharacterizationEx( long moduleNumber, long characterization);</p> <p>HRESULT get_ECALCharacterizationEx( long moduleNumber, long* characterization);</p>
<b>Interface</b>	ICalibrator4

## ECalID Property

---

**Description** Selects the model and serial number of the ECal module to be characterized. This command does NOT set the model and serial number of the ECal module.

**VB Syntax** *userChar*.**ECalID** = *value*

**Variable** [\(Type\)](#) - Description

*userChar* An [ECalUserCharacterizer](#) Object

*value* (String) Model and serial number of the ECal module to be characterized.

**Return Type** String

**Default** "" (Empty String)

**Examples** `userChar.ECalID = "N4433A, 00001"`

**C++ Syntax** HRESULT put\_ECalID(BSTR id);

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## ECALIsolation Property

---

**Description** **Note:** The inherent isolation of the PNA is better than that attained with this command. ONLY use this command when using an external test set, and ONLY using a 8509x ECal module.

Specifies whether the acquisition of the ECal calibration should include isolation or not.

**VB Syntax** *cal.ECALIsolation* = *value*

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

*value* (**boolean**)

**False** - Exclude Isolation

**True** - Include Isolation

**Return Type** Boolean

**Default** False

**Examples**

```
Dim oPNA as AgilentPNA835x.Application
Dim oCal as Calibrator
Set oPNA = CreateObject("AgilentPNA835x.Application",
"MachineName")
Set oCal = oPNA.ActiveChannel.Calibrator
' Uncomment the following line to have the cal include isolation
' oCal.ECALIsolation = True
' Uncomment the following line to have the cal omit isolation
' oCal.ECALIsolation = False
oCal.DoECAL2Port ' Do the cal
```

**C++ Syntax** HRESULT put\_ECALIsolation ( VARIANT\_BOOL bIsolationState );  
HRESULT get\_ECALIsolation ( VARIANT\_BOOL \*bIsolationState );

**Interface** Calibrator

---

Last Modified:

16-Apr-2007    Un-obsolete

## ECALModuleNumberList Property

---

**Description** Returns a list of index numbers to be used for referring to the ECal modules that are currently attached to the PNA.

**VB Syntax** *clist* = *cal*.**ECALModuleNumberList**

**Variable** [\(Type\)](#) - **Description**

*clist* Variable to store the returned list of index numbers.

*cal* [Calibrator](#) (object)

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
clist = cal.ECALModuleNumberList
'If 2 modules are attached to the PNA
'then the returned list will be:
1,2
```

**C++ Syntax** HRESULT get\_ECALModuleNumberList( VARIANT \*modules);

**Interface** ICalibrator6

## EcalOrientation Property

---

**Description** Specifies which port of the ECal module is connected to which port of the PNA when the [AutoOrient](#) property = False.

**VB Syntax** `SMC.EcalOrientation (mod) = value`

**Variable** [\(Type\)](#) - **Description**

*SMC* [SMCType](#) (**object**)

*mod* (Long)

1 - Use ECAL Module for the calibration.

*value* (**string**) -Format this parameter in the following manner:

Aw,Bx,Cy,Dz

where

- A, B, C, and D are literal ports on the ECAL module
- w,x,y, and z are substituted for PNA port numbers to which the ECAL module port is connected.

Ports of the module which are not used are omitted from the string.

For example, on a 4-port ECal module with

- port A connected to PNA port 2
- port B connected to PNA port 3
- port C not connected
- port D connected to PNA port 1

the string would be: A2,B3,D1

**Return Type** String

**Default** "A1,B2"

**Examples** `SMC.EcalOrientation (1) = "A2,B1"`

**C++ Syntax** `HRESULT put_EcalOrientation(long IModuleNum, BSTR orientation);`  
`HRESULT get_EcalOrientation(long IModuleNum, BSTR *orientation);`

**Interface** SMCType



## EcalOrientation1Port Property

---

**Description** **For Mixer Characterization ONLY** Specifies which port of the ECal module is connected to which port of the PNA for the [Do1PortECAL](#) property when the [AutoOrient](#) property = False.

**VB Syntax** `VMC.EcalOrientation1Port (mod) = value`

**Variable** [\(Type\)](#) - Description

*VMC* [VMCType](#) (object)

*mod* (Long)

1 - Use ECAL Module for the calibration.

*value* **(string)** - Choose from:

"A1" - ECAL module port A is connected to PNA port 1

"B1" - ECAL module port A is connected to PNA port 1

**Return Type** String

**Default** "A1"

If anything other than port 1 is specified, "B1" will be used. For example, if "A2" is specified, "B1" is used.

**Examples** `VMC.EcalOrientation1Port(1) = "B1"`

**C++ Syntax** `HRESULT put_EcalOrientation1Port(long IModuleNum, BSTR orientation);`  
`HRESULT get_EcalOrientation1Port(long IModuleNum, BSTR`

**Interface** VMCType

## EcalOrientation2Port Property

---

**Description** Specifies which port of the ECal module is connected to which port of the PNA for the [Do2PortECAL](#) property when the [AutoOrient](#) property = False.

**VB Syntax** `VMC.EcalOrientation2Port (mod) = value`

**Variable** [\(Type\)](#) - Description

*VMC* [VMCType](#) (object)

*mod* (Long) Module being used for the calibration.  
Choose from 1 or 2.

*value* **(string)** -Format this parameter in the following manner:

Aw,Bx,Cy,Dz

where

- A, B, C, and D are literal ports on the ECAL module
- w,x,y, and z are substituted for PNA port numbers to which the ECAL module port is connected.

Ports of the module which are not used are omitted from the string.

For example, on a 4-port ECal module with

- port A connected to PNA port 2
- port B connected to PNA port 3
- port C not connected
- port D connected to PNA port 1

the string would be: A2,B3,D1

**Return Type** String

**Default** "A1,B2"

**Examples** `VMC.EcalOrientation1Port(1) = "A2,B1"`

**C++ Syntax** `HRESULT put_EcalOrientation2Port(long IModuleNum, BSTR orientation);`  
`HRESULT get_EcalOrientation2Port(long IModuleNum, BSTR *orientation);`

**Interface** VMCType

## ECALPortMapEx Property

---

**Description** This property replaces [ECALPortMap Property](#). Specifies which ports of the ECal module are connected to which ports of the PNA for the [DoECAL1PortEx](#) and [DoECAL2PortEx](#) methods when the [OrientECALModule](#) property = False.

This setting remains until the PNA is restarted or this command is sent again.

**Note:** For guided calibrations where Orient is OFF and the same ECal module is used in more than one Connection Step, you are not allowed to specify how the ECal module is connected. Instead, the PNA determines the orientation. The PNA does not verify that you made the connection properly.

This command, and [OrientECALModule Property](#), can be used to perform ECal orientation using the [Guided Calibration](#) interface.

**VB Syntax** `cal.ECALPortMapEx (module) = value`

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

*module* (**long integer**) Optional argument. ECal module.

Choose from modules **1** through **8**

Use [IsECALModuleFoundEx](#) to determine the number of modules connected to the PNA

Use [GetECALModuleInfoEx](#) to return the model and serial number of each module.

*value* (**string**) -Format this parameter in the following manner:

Aw,Bx,Cy,Dz

where

- A, B, C, and D are literal ports on the ECAL module
- w,x,y, and z are substituted for PNA port numbers to which the ECAL module port is connected.

Ports of the module which are not used are omitted from the string.

For example, on a 4-port ECal module with

- port A connected to PNA port 2
- port B connected to PNA port 3
- port C not connected
- port D connected to PNA port 1

the string would be: A2,B3,D1

**DoECAL1PortEx** or **DoECAL2PortEx** methods will fail if the port numbers passed to those methods are not in the string of this property and [OrientECALModule](#) property = False.

**Return Type** String

**Default** Not Applicable

**Examples**

```
Dim cal As Calibrator
Dim sPortMap As String
Set cal = PNAapp.ActiveChannel.Calibrator
cal.ECALPortMapEx = "a2,b1" 'Write
sPortMap = cal.ECALPortMap 'Read
```

**[C++ Syntax](#)** HRESULT put\_ECALPortMapEx( long moduleNumber, BSTR strPortMap);  
HRESULT get\_ECALPortMapEx( long moduleNumber, BSTR \*strPortMap);

**Interface** ICalibrator4

---

Last Modified:

7-May-2007    Added note about orient

## ElecDelayMedium Property

---

**Description** Sets or returns the electrical delay medium.

**VB Syntax** *meas.ElecDelayMedium = value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement **(object)**

*value* **( enum NACalStandardMedium )** choose from  
0 - naCoax  
1 - naWaveGuide

**Return Type** NACalStandardMedium

**Default** Not Applicable

**Examples** `Print meas.ElecDelayMedium 'prints the value of the electrical delay medium`

**C++ Syntax** HRESULT get\_ElecDelayMedium(tagNACalStandardMedium \*pVal);  
HRESULT put\_ElecDelayMedium(tagNACalStandardMedium newVal);

**Interface** IMeasurement2

## ElecDistanceDelay Property

---

**Description** Sets the electrical delay in physical length (distance) for the selected measurement.

**VB Syntax** *meas.ElecDistanceDelay* = *value*

**Variable**    [\(Type\)](#) - Description

*meas*    A [Measurement](#) (object)

*value*    (**double**) - Electrical Delay in distance.

Set units using [ElecDistanceDelayUnit](#).

**Return Type** Double

**Default** 0

**Examples** `meas.ElecDistanceDelay = 1e-3 'Write`

`edelay = meas.ElecDistanceDelay 'Read`

**C++ Syntax** HRESULT get\_ElecDistanceDelay(double \*pVal)  
HRESULT put\_ElecDistanceDelay(double newVal)

**Interface** IMeasurement11

---

Last Modified:

6-Feb-2009    MX New topic

## ElecDistanceDelayUnit Property

---

**Description** Sets and returns the units for specifying electrical delay in physical length (distance).

**VB Syntax** *meas.ElecDistanceDelayUnit* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*value* (enum naDistanceUnit) Choose from:

0 – naDistanceUnit\_Meter

1 – naDistanceUnit\_Feet

2 – naDistanceUnit\_Inch

**Return Type** Enum

**Default** 0 – naDistanceUnit\_Meter

**Examples** `meas.ElecDistanceDelayUnit = naDistanceUnit_Meter` 'Write

`edelay = meas.ElecDistanceDelayUnit` 'Read

**C++ Syntax** HRESULT get\_ElecDistanceDelayUnit(tagNADistanceUnit \*pVal)  
HRESULT put\_ElecDistanceDelayUnit(tagNADistanceUnit newVal)

**Interface** IMeasurement11

---

Last Modified:

6-Feb-2009    MX New topic



## ElectricalDelay Property

---

**Description** Sets the Electrical Delay for the active channel.

**VB Syntax** *meas.ElectricalDelay* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement **(object)**

*value* **(double)** - Electrical Delay in seconds. Choose any number between **-9.99** and **9.99**

**Return Type** Double

**Default** 0

**Examples**

```
meas.ElectricalDelay = 1e-3 'Write
```

```
edelay = meas.ElectricalDelay 'Read
```

**C++ Syntax** HRESULT get\_ElectricalDelay(double \*pVal)  
HRESULT put\_ElectricalDelay(double newVal)

**Interface** IMeasurement

## Element Property

---

**Description** Returns a handle to the specified [PathElement](#) object. Each element object contains a unique set of values.

The [Value Property](#) is used to set the value for each element.

[See a list of configurable elements](#) and values for various PNA models.

**VB Syntax** Set elem = *pathConfig*.**Element** (*element*)

**Variable** [\(Type\)](#) - Description

*elem* **(Object)** [IPathElement](#)

*pathConfig* A [PathConfiguration](#) **(object)**

*element* **(String)** Configurable element. Use [pathConfig.Elements](#) to return a list of configurable elements or [see a list of configurable elements](#) for various PNA models.

**Return Type** Object

**Default** Not Applicable

**Examples**

```
Dim elem as PathElement
Set elem = app.ActiveChannel.PathConfiguration.Element("Src1")
```

**C++ Syntax** HRESULT Element( BSTR elemName, IPathElement\*\* ppElement);

**Interface** IPathConfiguration

---

Last Modified:

14-Dec-2006    MX New topic

## Elements Property

---

**Description** Returns an array containing the names of configurable elements.

See a [list of configurable elements and settings](#) for various PNA models.

**VB Syntax** *values* = *pathConfig*.**Elements**

**Variable** [\(Type\)](#) - **Description**

*values* **(Variant array)** Variable to store the array of configurable elements.

*pathConfig* A [PathConfiguration](#) **(object)**

**Return Type** Variant array

**Default** Not Applicable

**Examples** `elems=pathconfig.Elements`

**C++ Syntax** HRESULT Elements( VARIANT\* pElements );

**Interface** IPathConfiguration

---

Last Modified:

14-Dec-2006    MX New topic

## Embed4PortA Property

---

**Description** Returns the PNA port number associated with 'a' based on the device topology.

To see 'a' for all topologies, and to specify the port connections, use

[Embed4PortList Property](#)

Specify topology using [Embed4PortTopology Property](#)

**VB Syntax** *value* = *fixture*.**Embed4PortA**

**Variable** [\(Type\)](#) - **Description**

*value* (Short Integer) Variable to store the returned PNA port number.

*fixture* A [Fixturing](#) (object)

**Return Type** Integer

**Default** Not Applicable

**Examples** `value = fixture.Embed4PortA 'Read`

**C++ Syntax** HRESULT get\_Embed4PortA(short \*portA );

**Interface** IFixturing2

## Embed4PortB Property

---

<b>Description</b>	Returns the PNA port number associated with 'b' based on the device topology. To see 'b' for all topologies, and to specify the port connections, use <a href="#">Embed4PortList Property</a> . Specify topology using <a href="#">Embed4PortTopology Property</a>
<b>VB Syntax</b>	<i>value</i> = <i>fixture</i> . <b>Embed4PortB</b>
<b>Variable</b>	<b>(Type) - Description</b>
<i>value</i>	(Short Integer) Variable to store the returned PNA port number.
<i>fixture</i>	A <a href="#">Fixturing</a> (object)
<b>Return Type</b>	Integer
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>value = fixture.Embed4PortB 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_Embed4PortB(short *portB );
<b>Interface</b>	IFixturing2

## Embed4PortC Property

---

**Description** Returns the PNA port number associated with 'c' based on the device topology.  
To see 'c' for all topologies, and to specify the port connections, use [Embed4PortList Property](#).

Specify topology using [Embed4PortTopology Property](#)

**VB Syntax** *value* = *fixture*.**Embed4PortC**

**Variable** [\(Type\)](#) - **Description**

*value* (Short Integer) Variable to store the returned PNA port number.

*fixture* A [Fixturing](#) (**object**)

**Return Type** Integer

**Default** Not Applicable

**Examples** `value = fixture.Embed4PortC 'Read`

**C++ Syntax** HRESULT get\_Embed4PortC(short \*portC );

**Interface** IFixturing2

## Embed4PortD Property

---

**Description** Returns the PNA port number associated with 'd' based on the device topology.  
To see 'd' for all topologies, and to specify the port connections, use [Embed4PortList Property](#).

Specify topology using [Embed4PortTopology Property](#)

**VB Syntax** *value* = *fixture*.**Embed4PortD**

**Variable** [\(Type\)](#) - **Description**

*value* (Short Integer) Variable to store the returned PNA port number.

*fixture* A [Fixturing](#) (**object**)

**Return Type** Integer

**Default** Not Applicable

**Examples** `value = fixture.Embed4PortD 'Read`

**C++ Syntax** HRESULT get\_Embed4PortD(short \*portD );

**Interface** IFixturing2

## Embed4PortList Property

---

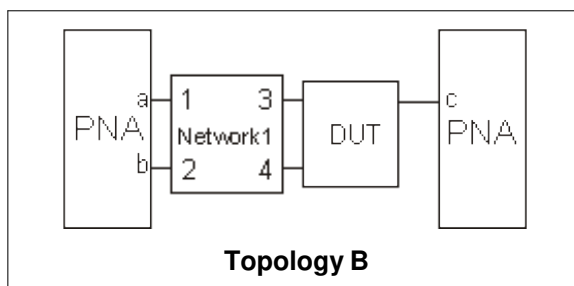
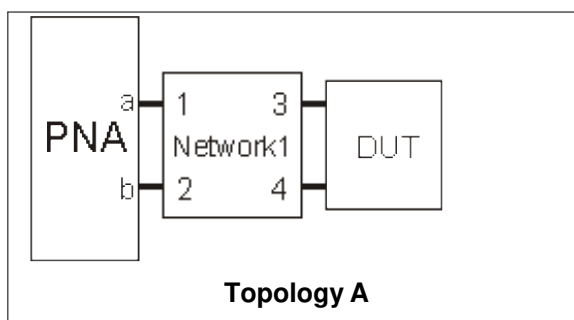
**Description** Specifies the PNA port connections for ALL topologies. The port assignment is dependent on the DUT topology. All four port numbers are required. However, for:

- Topology A, only the first **two** arguments are valid,
- Topology B, only the first **three** arguments are valid,
- Topology C, **ALL** arguments are valid.

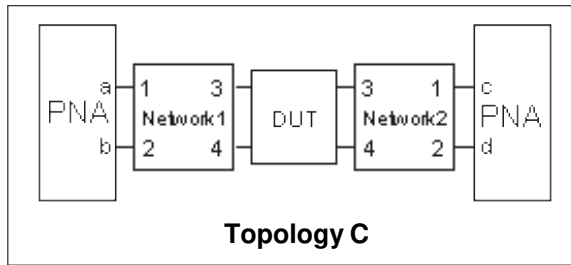
Specify topology using [Embed4PortTopology Property](#).

Read the port assignments using the following commands. A, B, C, and D, refer to the port; NOT the topology.

- [Embed4PortA Property](#)
- [Embed4PortB Property](#)
- [Embed4PortC Property](#)
- [Embed4PortD Property](#)







**VB Syntax** *fixture.Embed4PortList = p1, p2, p3, p4*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*p1* PNA Port number assigned to **a** in above graphic.

*p2* PNA Port number assigned to **b** in above graphic.

*p3* PNA Port number assigned to **c** in above graphic.

*p4* PNA Port number assigned to **d** in above graphic.

**Return Type** Four Integers

**Default** 1,2,3,4

**Examples** `fixture.4PortNetworkTopoCPorts = 4,3,2,1` **Write**

**C++ Syntax** HRESULT put\_4PortNetworkTopoCPorts(short ChannelNum, short pPortA, short pPortB, short pPortC, short pPortD)

**Interface** IFixturing2

## Embed4PortNetworkFilename Property

---

**Description** Specifies the 4-port touchstone file (\*.s4p) in which the network to embed or de-embed resides. If the specified file does not exist, an error occurs when type command is sent.

Following this command, send [Embed4PortNetworkMode Property](#).

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture*.Embed4PortNetworkFilename(*netNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*netNum* **(Integer)** Network position. Choose from **1** or **2**. See [Embed4PortTopology Property](#)

*value* **(String)** Full path, file name, and extension (.s4P) of the circuit.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents"

**Return Type** String

**Default** Not Applicable

**Examples** `fixture.Embed4PortNetworkFilename(2) = "C:/Program Files/Agilent/Network Analyzer/Documents/myFile.s4p" 'Write`

`value = fixture.Embed4PortNetworkFilename(1) 'Read`

**C++ Syntax** HRESULT get\_Embed4PortNetworkFilename( short networkNum, BSTR \*filename);  
HRESULT put\_Embed4PortNetworkFilename( short networkNum, BSTR filename);

**Interface** IFixturing2

## Embed4PortNetworkMode Property

---

**Description** Specify the type of processing to take place on the specified 4-port network. First specify the network filename with [FSim.Embed4PortNetworkFilename Property](#).

**VB Syntax** *fixture*.Embed4PortNetworkMode(*netNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*netNum* **(Integer)** Network position. Choose from **1** or **2**. See [Embed4PortTopology Property](#)

*value* **(Enum as NA4PortEmbedNetworkMode)** Processing mode. Choose from:

- **0** or **naNO\_NETWORK** - The same as disabling.
- **1** or **naEMBED\_NETWORK** - Add Network circuit.
- **2** or **naDEEMBED\_NETWORK** - Remove Network circuit

**Return Type** Enum

**Default** naNO\_NETWORK

**Examples** `fixture.Embed4PortNetworkMode(1) = naNO_NETWORK 'Write`

`value = fixture.Embed4PortNetworkMode(2) 'Read`

**C++ Syntax** HRESULT get\_Embed4PortNetworkMode( short networkNum, tagNA4PortEmbedNetworkMode \*eVal );  
 HRESULT put\_Embed4PortNetworkMode( short networkNum, tagNA4PortEmbedNetworkMode eVal );

**Interface** IFixturing2

## Embed4PortState Property

---

**Description** Turns ON or OFF 4-port Network embedding for all ports on the channel.

**VB Syntax** *fixture.Embed4PortState = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns Embedding OFF

**True** - Turns Embedding ON

**Return Type** Boolean

**Default** False (OFF)

**Examples** `fixture.Embed4PortState = False 'Write`

`value = fixture.Embed4PortState 'Read`

**C++ Syntax** HRESULT get\_Embed4PortState(VARIANT\_BOOL \*pVal)  
HRESULT put\_Embed4PortState(VARIANT\_BOOL newVal)

**Interface** IFixturing2

## Embed4PortTopology Property

---

**Description** Specifies the PNA / DUT topology. [Learn more about these and other PNA/DUT configurations.](#)

**VB Syntax** *fixture.Embed4PortTopology = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Enum as **NA4PortEmbedTopology**) PNA / DUT topology. Choose from:

**0** or **naTOPOLOGY\_A** - 2 PNA/DUT Ports

**1** or **naTOPOLOGY\_B** - 3 PNA/DUT Ports

**2** or **naTOPOLOGY\_C** - 4 PNA/DUT Ports

**Return Type** Enum

**Default** **naTOPOLOGY\_A** (2 PNA/DUT Ports)

**Examples** `fixture.Embed4PortTopology = naTOPOLOGY_A 'Write`

`value = fixture.Embed4PortTopology 'Read`

**C++ Syntax** HRESULT get\_Embed4PortTopology( tagNA4PortEmbedTopology \*eVal );  
HRESULT put\_Embed4PortTopology( tagNA4PortEmbedTopology eVal );

**Interface** IFixturing2

## Enable Property

---

**Description** Turns ON / OFF the trigger output.

**VB Syntax** *auxTrig.Enable* = *state*

**Variable** ([Type](#)) - Description

*auxTrig* An [AuxTrigger](#) (object)

*state* (boolean)

**True** - Trigger Output ON

**False** - Trigger Output OFF

**Return Type** Boolean

**Default** False

**Exaamples** `auxTrig.Enable = True 'Write`

`value = auxTrig.Enable 'Read`

**[C++ Syntax](#)** HRESULT get\_Enable(VARIANT\_BOOL \* enable);  
HRESULT put\_Enable(VARIANT\_BOOL enable);

**Interface** IAuxTrigger

---

Last Modified:

14-Dec-2006    MX New topic

## Enabled Property

---

**Description** Enables and disables (ON/OFF) the port mapping and control line output of the specified test set.

If the specified test set is not connected or not ON, then setting Enabled = True will report an error. All other properties can be set when the test set is not connected.

When this command is set to ON or OFF, then the display of the test set status bar ([ShowProperties Property](#)) is also set to ON or OFF.

**VB Syntax** *tset.Enabled = value*

**Variable** [\(Type\)](#) - Description

*tset* A [TestsetControl](#) object

OR

An [E5091Testset](#) object

*value* **(Boolean)**

**True** Enables test set control.

**False** Disables test set control.

**Return Type** Boolean

**Default** False

**Examples** [See E5091A Example Program](#)

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_Enabled(VARIANT\_BOOL \*state);  
HRESULT put\_Enabled(VARIANT\_BOOL state);

**Interface** ITestsetControl  
IE5091Testsets

## EnableLOPowerCal Property

---

**Description** Sets and returns whether or not the LO power cal step is included in the cal steps when an IMDX Cal or NFX Cal is performed.

**VB Syntax** *obj.EnableLOPowerCal(n) = value*

**Variable**    [\(Type\)](#) - Description

*obj*    A [SweptIMDCal](#) (object)

         A [NoiseCal](#) (object)

*n*    LO Stage. Choose 1. (Only single LO allowed)

*value*    (Boolean) Choose from:

**False** - Skips over the LO Power Cal when calibrating.

**True** - Includes a step for LO Power Cal when calibrating.

**Return  
Type**    Boolean

**Default**    PNA Rev. 9.1 and above: **False**  
              Before Rev 9.1: **True**

**Examples**    `imd.EnableLOPowerCal(1)= true 'Write`

`loPwrCal = imd.EnableLOPowerCal(1) 'Read`

**C++ Syntax**    HRESULT get\_EnableLOPowerCal (long stage, BOOL \*enable)  
                  HRESULT put\_EnableLOPowerCal (long stage, BOOL enable)

**Interface**    ISweptIMD2  
                  INoiseCal2

---

Last Modified:

21-Oct-2009    Added NoiseCal

27-Mar-2009    MX New topic



## EnableSourceUnleveledEvents Property

---

**Description** Specifies whether or not to report [Source Unleveled](#) errors as system events. These events can trigger an [OnSystemEvent](#) call.

This setting will revert to the default (False) setting on Instrument Preset.

**VB Syntax** *pref.EnableSourceUnleveledEvents = bool*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*bool* (Boolean) - Choose from:

**False** - Do NOT report Source Unleveled Errors.

**True** - Report Source Unleveled Errors.

**Return Type** Boolean

**Default** False

**Examples** `pref.EnableSourceUnleveledEvents = False 'Write`

`prefer = pref.EnableSourceUnleveledEvents 'Read`

**C++ Syntax** HRESULT put\_EnableSourceUnleveledEvents( VARIANT\_BOOL bsourcUnlEnable)  
 HRESULT get\_EnableSourceUnleveledEvents( VARIANT\_BOOL \*bsourcUnlEnable)

**Interface** IPreferences3

---

Last modified:

9-Apr-2010 Added Preset note

15-Nov-2006 MX New command

## EndOfSweepOperation Property

---

**Description** Set and read the action which should be taken at the end of the last frequency or power sweep in the measurement. This setting is used to protect a sensitive device from too much power during the sweep retrace.

**VB Syntax** `gca.EndOfSweepOperation = value`

**Variable** [\(Type\)](#) - [Description](#)

*gca* A [GainCompression](#) (object)

*value* (**NAGCAEndOfSweepOperation**)

- **naStandard (0)** Use the default PNA method. [Learn more.](#)
- **naSetToStartPower (1)** Sweep Start power
- **naSetToStopPower (2)** Sweep Stop power.
- **naSetRFOff (3)** Always turn power OFF while waiting.

**Return Type** Enum

**Default** naStandard

**Examples** `gca.EndOfSweepOperation = naSetToStartPower 'Write`

```
eos = gca.EndOfSweepOperation 'Read
```

**C++ Syntax** HRESULT get\_EndOfSweepOperation(tagNAGCAEndOfSweepOperation\* pVal)  
HRESULT put\_EndOfSweepOperation(tagNAGCAEndOfSweepOperation newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## ENRFile Property

---

**Description** Sets and returns the name of the ENR file associated with the noise source.

**VB Syntax** *noise*.ENRFile = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseCal](#) (object)

*value* (**string**) Full path and ENR filename.

**Return  
Type** String

**Default** Not Applicable

**Examples** `noise.ENRFile = "c:/ProgramFiles/Agilent/Network  
Analyzer/Documents/ENR/346C.enr" 'Write`

`ENR = noise.ENRFile 'Read`

**C++ Syntax** HRESULT get\_ENRFile(BSTR\* pValue)  
HRESULT put\_ENRFile(BSTR pNewValue)

**Interface** INoiseCal

---

Last Modified:

29-May-2007 MN New topic

## ENRID Property

---

**Description** Sets and returns ID of ENR table.

**VB Syntax** *enr*.ENRSN = *ID*

**Variable**    [\(Type\)](#) - Description

*enr*    An [ENRFile](#) (**object**)

*ID*    Identifier for the ENR table

**Return  
Type**    String

**Default**    Not Applicable

**Examples**    [See example program](#)

**C++ Syntax**    HRESULT get\_ENRID(BSTR \*Val);  
                  HRESULT put\_ENRID(BSTR Val);

**Interface**    IENRFile

---

Last Modified:

2-Aug-2007    MX New topic

## ENRSN Property

---

**Description** Sets and returns the serial number of the noise source for which the ENR file applies.

**VB Syntax** *enr.ENRSN = serialNumber*

**Variable** [\(Type\)](#) - Description

*enr* An [ENRFile](#) (object)

*serialNumber* (String) Serial number of the noise source.

**Return Type** String

**Default** Not Applicable

**Examples** [See example program](#)

**C++ Syntax** HRESULT get\_ENRSNBSTR \*Val);  
HRESULT put\_ENRSN(BSTR Val);

**Interface** IENRFile

---

Last Modified:

2-Aug-2007    MX New topic

## ErrorCorrection Property

---

**Description** Sets (or returns) error correction ON or OFF for the measurement.

**VB Syntax** *meas*.**ErrorCorrection** = *value*

**Variable**    [\(Type\)](#) - Description

*meas*    A Measurement (**object**)

*value*    (**boolean**)

**False** - Turns error correction OFF

**True** - Turns error correction ON

**Return Type** Boolean

**Default** See [Error Correction](#)

**Examples** `meas.ErrorCorrection = True` 'Write

`errcorr = meas.ErrorCorrection` 'Read

**C++ Syntax** HRESULT put\_ErrorCorrection (VARIANT\_BOOL bState)  
HRESULT get\_ErrorCorrection (VARIANT\_BOOL \*bState)

**Interface** IMeasurement

## ErrorCorrection (Channel) Property

---

**Description** Attempts to sets error correction ON or OFF for all of the measurements on the channel. This setting may not be successful for some measurements because the Cal Set currently in place may not contain the appropriate calibration data. To read the error correction state for a measurement, use [Error Correction Property](#).

**VB Syntax** *chan*.**ErrorCorrection** = *value*

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (object)

*value* **(boolean)**

**False** - Turns error correction OFF

**True** - Turns error correction ON

**Return Type** Boolean

**Default** [About Error Correction](#)

**Examples** `chan.ErrorCorrection = True`

**C++ Syntax** HRESULT put\_ErrorCorrection (VARIANT\_BOOL bState)

**Interface** IChannel7

Write/Read

## ExternalALC Property

---

**Description** Sets or returns the source of the analyzer leveling control.

**VB Syntax** *app.ExternalALC = value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (boolean) - Choose from:  
**True** - Leveling control supplied through the rear panel.  
**False** - Leveling control supplied inside the analyzer

**Return Type** Boolean

**Default** False

**Examples**

```
app.ExternalALC = True 'Write
```

```
extALC = app.ExternalALC 'Read
```

**C++ Syntax** HRESULT get\_ExternalALC(VARIANT\_BOOL \*pVal)  
HRESULT put\_ExternalALC(VARIANT\_BOOL newVal)

**Interface** IApplication



## ExternalTriggerConnectionBehavior Property

---

**Description** Configures the external triggering signal for the PNA.

- To control BNC1 and BNC2 with this command, then you **MUST** have [AuxTriggerScopelsGlobal](#) = True.
- [TriggerSouce Property](#) is automatically set to External when **ExternalTriggerConnectionBehavior** is sent.
- Edge triggering is only available on some PNA models.
- For more information, see [External Triggering](#).

**VB Syntax** `trigsetup.ExternalTriggerConnectionBehavior (conn) = value`

**Variable** [\(Type\)](#) - Description

*trigsetup* A [TriggerSetup](#) (object)

*conn* (enum **NATriggerConnection**) Rear Panel connector to send or receive trigger signals. Choose from:

Only one of the input connectors is active at a time. When a command is sent to one, the PNA automatically makes the other INACTIVE.

0 - **naTriggerConnectionAUXT** Trigger IN from rear-panel [AUX IO connector Pin 19](#)

1 - **naTriggerConnectionBNC1** Trigger IN from rear-panel [Trigger IN BNC connector](#)

2 - **naTriggerConnectionBNC2** Trigger OUT to rear-panel [Trigger OUT BNC connector](#). Only useful in point sweep mode.

3 - **naTriggerConnectionMATH** Trigger IN from rear-panel [Material Handler connector Pin 18](#)

*value* (enum **NAExternalTriggerBehavior**) -

0 - **naTriggerInactive** - Disables the specified connector.

**Choose from ONLY 1 through 4** when <conn> is set to either **naTriggerConnectionBNC1** or **naTriggerConnectionAUXT**

1 - **naTriggerInEdgeNegative** - Triggers the PNA when receiving a negative going signal

2 - **naTriggerInEdgePositive** - Triggers the PNA when receiving a positive going signal

3 - **naTriggerInLevelLow** - Triggers the PNA when receiving a low level signal

4 - **naTriggerInLevelHigh** - Triggers the PNA when receiving a High-level signal

**Choose from ONLY 5 through 8** when <conn> is set to **naTriggerConnectionBNC2**.

In addition to sending this command, you must also use [TriggerOutputEnabled Property](#) to enable

the BNC2 output.

5 - **naTriggerOutPulsePositiveAfter** - Sends a POSITIVE going TTL pulse at the END of each point during the sweep.

6 - **naTriggerOutPulsePositiveBefore** - Sends a POSITIVE going TTL pulse at the START of each point during the sweep.

7 - **naTriggerOutPulseNegativeAfter** - Sends a NEGATIVE going TTL pulse at the END of each point during the sweep.

8 - **naTriggerOutPulseNegativeBefore** - Sends a NEGATIVE going TTL pulse at the START of each point during the sweep.

**Return Type** Enum as NAExternalTriggerBehavior

**Default** BNC1 = **naTriggerInactive**  
BNC2 = **naTriggerInactive**  
AUXT = **naTriggerInLevelHigh**

When [Output is enabled](#)

BNC1 = **naTriggerInactive**

BNC2 = **naTriggerOutPulsePositiveAfter**

AUXT = **naTriggerInLevelHigh**

**Examples** `trigsetup.ExternalTriggerConnectionBehavior (naTriggerConnectionAUXT) = naTriggerInLevelLow 'Write`

```
trigBehav = trigsetup.ExternalTriggerConnectionBehavior
(naTriggerConnectionAUXT) 'Read
```

**C++ Syntax** HRESULT get\_ExternalTriggerConnectionBehavior(tagNATriggerConnection connection,tagNAExternalTriggerBehavior \*trigger);  
HRESULT put\_ExternalTriggerConnectionBehavior(tagNATriggerConnection connection,tagNAExternalTriggerBehavior trigger);

**Interface** ITriggerSetup

---

Last Modified:

25-Feb-2008    Added 'Global' note

## ExternalTriggerDelay Property

---

**Description** Sets and reads the trigger delay for all measurements in the CHANNEL. This delay is only applied while in [app.Source = naTriggerSourceExternal](#) and [trigsetup.Scope = naChannelTrigger](#) . After an external trigger is applied, the start of the sweep is delayed for the specified delay value plus any inherent latency.

To apply a trigger delay for all channels (Global), use [TriggerDelay Property](#).

**VB Syntax** *chan.ExternalTriggerDelay = value*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*value* **Double**- Trigger delay value in seconds. Range is from 0 to 107 seconds

**Return Type** Double

**Default** 0

**Examples** `chan.ExternalTriggerDelay = .003 'Write`

`delay = chan.ExternalTriggerDelay 'Read`

**C++ Syntax** HRESULT get\_ExternalTriggerDelay(double \*delay);  
HRESULT put\_ExternalTriggerDelay(double delay)

**Interface** IChannel6

## F1Frequency Property

---

**Description** Sets and returns the frequency of the F1 tone. Use with IMD sweep types:

- **naIMDToneCWSweep**
- **naIMDTonePowerSweep**

**VB Syntax** *object.F1Frequency = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*value* (Double) F1 tone frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return Type** Double

**Default** .9995 GHz

**Examples** `imd.F1Frequency = 100e6 'Write`

`value = imd.F1Frequency 'Read`

**C++ Syntax** HRESULT get\_F1Frequency(double \*pVal)  
HRESULT put\_F1Frequency(double newVal)

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## F2Frequency Property

---

**Description** Sets and returns the frequency of the F2 tone. Use with IMD sweep types:

- **naIMDToneCWSweep**
- **naIMDTonePowerSweep**

**VB Syntax** *object.F2Frequency = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*value* (Double) F2 tone frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return Type** Double

**Default** 1.0005 GHz

**Examples** `imd.F2Frequency = 200e9 'Write`

`value = imd.F2Frequency 'Read`

**C++ Syntax** HRESULT get\_F2Frequency(double \*pVal)  
HRESULT put\_F2Frequency(double newVal)

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## FastCWCount Property

---

**Description** Enables Fast CW sweep and sets the number of data points for the channel. [Sweep Type](#) must already be set to CWTime and FIFO must already be enabled.

### See Also

[FIFO and other Antenna Features](#)

[FIFO Object](#)

[Example program](#)

[N5264A Measurement Receiver](#)

**VB Syntax** *chan.FastCWCount = value*

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) Object

*value* (Long Integer) Number of data points to measure in Fast CW mode. This setting overwrites the standard number of points setting for the channel.  
Set to 0 to disable Fast CW.

**Return Type** Long Integer

**Default** 0

**Examples** `chan.FastCWCount = 1e3 'Write`

`value = chan.FastCWCount 'Read`

**C++ Syntax** HRESULT get\_FastCWCount(long \*value)  
HRESULT put\_FastCWCount(long value)

**Interface** IChannel16

---

Last Modified:

7-Oct-2008    MX New topic

## FastMode Property

---

**Description** Sets and returns the state of a separate IFBW setting for leveling sweeps. ON allows a higher (faster) IFBW than the measurement sweep. It also causes leveling sweeps to be noisier.

**VB Syntax** *RxLevel.FastMode(srcPort) = value*

**Variable** [\(Type\)](#) - **Description**

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set Fast Mode for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Boolean) Separate IFBW setting state. Choose from:

**True** - Use separate IFBW setting. Specify IFBW using [LevelingIFBW](#).

**False** - Use same IFBW as the measurement sweep. Specify IFBW using [IF Bandwidth](#).

**Return Type** Variant Boolean

**Default** True

**Examples** `rxLevel.FastMode (1) = True ' Write`

`value = rxLevel.FastMode 2 ' Read`

**C++ Syntax** HRESULT get\_FastMode(long port, VARIANT\_BOOL\* pVal);  
HRESULT put\_FastMode(long port, VARIANT\_BOOL newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## FilterBW Property

---

**Description** Returns the results of the SearchBandwidth method.

**VB Syntax** *filtBW* = *meas*.**FilterBW**

**Variable** [\(Type\)](#) - Description

*filtBW* **(single)** - Variable to store bandwidth data

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples** `filterBW = meas.FilterBW 'Read`

**C++ Syntax** HRESULT get\_FilterBW(float\* bw)

**Interface** IMeasurement



## FilterCF Property

---

**Description** Returns the Center Frequency result of the SearchBandwidth method.

**VB Syntax** *filtCF* = *meas*.**FilterCF**

**Variable** [\(Type\)](#) - **Description**

*filtCF* **(double)** - Variable to store bandwidth CF data

*meas* A Measurement **(object)**

**Return Type** Double

**Default** Not applicable

**Examples** `filtCF = meas.FilterCF 'Read`

**C++ Syntax** HRESULT get\_FilterCF(double\* centerFrequency)

**Interface** IMeasurement

## FilterLoss Property

---

**Description** Returns the Loss value of the SearchBandwidth method.

**VB Syntax** *filtLoss* = *meas*.**FilterLoss**

**Variable** [\(Type\)](#) - **Description**

*filtLoss* **(single)** - Variable to store bandwidth Loss data

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples** `filterLoss = meas.FilterLoss 'Read`

**C++ Syntax** HRESULT get\_FilterLoss(float\* loss)

**Interface** IMeasurement

## FilterQ Property

---

**Description** Returns the Q (quality factor) result of the SearchBandwidth method.

**VB Syntax** *filtQ* = *meas*.**FilterQ**

**Variable** [\(Type\)](#) - **Description**

*filtQ* **(single)** - Variable to store bandwidth Q data

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples** `filtQ = meas.FilterQ 'Read`

**C++ Syntax** HRESULT get\_FilterQ(float\* quality)

**Interface** IMeasurement

## FilterErrors Property

---

**Description** Returns the error string associated with the digital filters. The return string has three fields separated by commas: "stage1 status, stage2 status, stage3 status"

Each of these fields can contain one or more of the following error codes:

- **NO ERROR**
- **\*NUMBER-OF-COEFFICIENTS** - the number of coefficients is excessive for that filter-stage
- **\*COEFFICIENT VALUE** - one or more coefficients are out of range for that filter-stage
- **\*SUM-OF-COEFFICIENTS** - the sum of all coefficients is excessive for that filter-stage,
- **\*FREQUENCY** - the frequency for Stage 1 is out of range (only applies stage1 field),
- **\*PARAMETER** - one or more parameters are out of range (only applies to stage 3 field)

**VB Syntax** *value* = *spm4*.FilterErrors

**Variable** [\(Type\)](#) - Description

*value* Variable to store the returned errors.

*spm4* A [SignalProcessingModuleFour](#) (object)

**Return Type** String

**Default** Not Applicable

**Examples**

```
mode = spm4.FilterErrors 'Read
'example return strings"
NO ERROR, NO ERROR, NO ERROR
indicates no errors,
*SUM-OF-COEFFICIENTS, NO ERROR, NO ERROR
indicates that the sum of all filter coefficients exceed the
maximum value for the Stage-1 filter,
*COEFFICIENT *SUM-OF-COEFFICIENTS, NO ERROR, *PARAMETER
indicates a problems with Stage 1 coefficients and a problem
with one or more of the parameters associated with the Stage 3
filter.
```

**C++ Syntax** HRESULT get\_FilterErrors(BSTR\* dspErrors);

Last Modified:

1-Jan-2007    MX New topic

## FilterMode Property

---

**Description** Sets and returns whether the PNA configures the 3-stage digital filter settings or they will be configured manually. When making manual settings, also send [ADCCaptureMode Property](#) which routes the IF through the 3-stage filter.

**VB Syntax** `spm4.FilterMode = value`

**Variable** [\(Type\)](#) - Description

*spm4* A [SignalProcessingModuleFour](#) (object)

*value* (enum as **NAModes**) Filter mode. Choose from:

**naAUTO** PNA controls digital filter settings.

**naMANUAL** You control digital filter settings using other [SignalProcessingModuleFour](#) commands.

**Return  
Type** Enum

**Default** naAUTO

**Examples** `spm4.FilterMode = naAUTO 'Write`

`mode = spm4.FilterMode 'Read`

**C++ Syntax** HRESULT get\_FilterMode(tagNAModes\* dspMode);  
HRESULT put\_FilterMode(tagNAModes dspMode);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

24-Jan-2007    MX New topic

## FirmwareMajorRevision Property

---

**Description** Returns the major firmware revision number as an integer. For example, given a firmware revision number A.03.30, this command returns 3.

**VB Syntax** *value* = *cap*.FirmwareMajorRevision

**Variable** [\(Type\)](#) - [Description](#)

*value* (Long) - Variable to store the returned integer value of the firmware revision number.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.FirmwareMajorRevision 'Read`

**C++ Syntax** HRESULT get\_FirmwareMajorRevision(long \* majorRev );

**Interface** ICapabilities

## FirmwareMinorRevision Property

---

**Description** Returns the minor firmware revision number as an integer. For example, given a firmware revision number A.03.30, this command returns 30.

**VB Syntax** *value* = *cap*.FirmwareMinorRevision

**Variable** [\(Type\)](#) - [Description](#)

*value* (Long) - Variable to store the returned decimal value of the firmware revision number.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.FirmwareMinorRevision 'Read`

**C++ Syntax** HRESULT get\_FirmwareMinorRevision(long \* minorRev );

**Interface** ICapabilities



## FirmwareSeries Property

---

<b>Description</b>	Returns the alpha portion of the firmware revision number. For example, given a firmware revision number A.03.30, this command returns A.
<b>VB Syntax</b>	<i>value</i> = <i>cap</i> .FirmwareSeries
<b>Variable</b>	<b>(Type) - Description</b>
<i>value</i>	(String) - Variable to store the returned alpha value of the firmware revision number.
<i>cap</i>	A <a href="#">Capabilities</a> (object)
<b>Return Type</b>	String
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>value = cap.FirmwareSeries 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_FirmwareSeries(BSTR * series);
<b>Interface</b>	ICapabilities

## FixturingState Property

---

**Description** Turns all three fixturing functions (de-embedding, port matching, impedance conversion) ON or OFF for all ports on the specified channel. This does NOT affect port extensions.

**VB Syntax** *fixture.FixturingState = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (boolean)

**True** - Turns Fixturing ON

**False** - Turns Fixturing OFF

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.FixturingState = True 'Write`

`value = fixture.FixturingState 'Read`

**C++ Syntax** HRESULT get\_FixturingState(VARIANT\_BOOL \*pVal)  
HRESULT put\_FixturingState(VARIANT\_BOOL newVal)

**Interface** IFixturing

Read-only

## FootSwitch Property

---

**Description** Reads the Footswitch Input (pin 20 of the AUX IO connector).

**VB Syntax** *value* = *AuxIO.Footswitch*

**Variable** [\(Type\)](#) - Description

*value* **(boolean)** - Variable to store the returned value

**False** - foot switch is released

**True** - footswitch is depressed

*AuxIO* **(object)** - A Hardware Aux I/O object

**Return Type** Boolean

**Default** True

**Examples** `fs = aux.Footswitch`

**C++ Syntax** HRESULT get\_FootSwitch ( VARIANT\_BOOL\* State );

**Interface** IHWAuxIO

## FootswitchMode Property

---

<b>Description</b>	Determines what occurs when the footswitch is pressed. For more information see the <a href="#">FootSwitch In pin description</a> in the Auxiliary IO connector.
<b>VB Syntax</b>	<i>AuxIo</i> . <b>FootSwitchMode</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>value</i>	<p><b>(enum NAFootSwitchMode )</b></p> <p><b>0 - naIgnoreFootswitch</b> - Footswitch presses are ignored.</p> <p><b>1 - naSweepTrigger</b> - Footswitch presses trigger a sweep. The PNA must be in Manual Trigger Mode.</p> <p><b>2 - naRecallNextState</b> - Footswitch presses recall an instrument state. When more than one state is available, then each footswitch press recalls the next state, then starts over from the beginning. It is possible for a recalled state to override the current mode. If the recalled state is IGNore, then mode changes and additional footswitch presses are ignored.</p> <p><b>3 - naRunMacro</b> - Footswitch presses load and run a macro. When more than one macro is available, then each footswitch press loads and runs the next macro, then starts over from the beginning. It is possible for a Macro to override the current mode. If the macro contains a Preset, then the mode changes to the default setting IGNore and additional footswitch presses are ignored.</p>
<i>AuxIO</i>	<b>(object)</b> - A Hardware Aux I/O object
<b>Return Type</b>	<b>NAFootSwitchMode</b>
<b>Default</b>	<b>0 - naIgnoreFootswitch</b>
<b>Examples</b>	<pre>auxIo.FootSwitchMode = naIgnoreFootSwitch 'Write</pre>
<b><a href="#">C++ Syntax</a></b>	<pre>HRESULT get_FootSwitchMode(NAFootSwitchMode *pFootSwitchMode ) HRESULT put_FootSwitchMode(NAFootSwitchMode newFootSwitchMode)</pre>
<b>Interface</b>	IHWAuxIO3

## Format Property (marker)

---

**Description** Sets (or returns) the format of the marker.

**VB Syntax** *mark.Format* = *value*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**enum NAMarkerFormat**) - Choose from:

- 0 - naMarkerFormat\_LinMag
- 1 - naMarkerFormat\_LogMag
- 2 - naMarkerFormat\_Phase
- 3 - naMarkerFormat\_Delay
- 4 - naMarkerFormat\_Real
- 5 - naMarkerFormat\_Imaginary
- 6 - naMarkerFormat\_SWR
- 7 - naMarkerFormat\_LinMagPhase
- 8 - naMarkerFormat\_LogMagPhase
- 9 - naMarkerFormat\_ReallImaginary
- 10 - naMarkerFormat\_ComplexImpedance
- 11 - naMarkerFormat\_ComplexAdmittance
- 12 - naMarkerFormat\_Kelvin
- 13 - naMarkerFormat\_Fahrenheit
- 14 - naMarkerFormat\_Celsius

**Return Type** NAMarkerFormat

**Default** 1 - naMarkerFormat\_LogMag

**Examples**

```
mark.Format = naMarkerFormat_SWR 'Write
fmt = mark.Format 'Read
```

**C++ Syntax** HRESULT get\_Format(tagNAMarkerFormat \*pVal)  
 HRESULT put\_Format(tagNAMarkerFormat newVal)

**Interface** IMarker

---

Last Modified:

1-Oct-2007    Added temperature formats

## FormatUnit Property

---

**Description** Sets and returns the units for the specified data format. Measurements with display formats other than those specified are not affected.

**VB Syntax** *meas.FormatUnit (format) = value*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*format* (enum **NADataFormat**) - Choose:

1 - naDataFormat\_LogMag

*value* (enum **naFormatUnit**) - Choose from:

**0 - naFormatUnit\_dBm** Units are displayed in dBm. 0 dBm = 0.001 watt

**1 - naFormatUnit\_dBmV** Units are displayed in dBmV. 0 dBmV = 0.001 volt

**Return Type** Enum

**Default** 0 - naFormatUnit\_dBm

**Examples** `meas.FormatUnit(1) = naFormatUnit_dBmV 'Write`

`units = meas.FormatUnit(1) 'Read`

**C++ Syntax** HRESULT put\_FormatUnit(tagDataFormat format, tagFormatUnit unit)  
 HRESULT get\_FormatUnit(tagDataFormat format, tagFormatUnit\* unit)

**Interface** IMeasurement9

---

Last Modified:

26-Aug-2008    MX New topic

## Frequency Property

---

<b>Description</b>	Sets or returns the frequency associated with a Power Sensor CalFactor Segment or Sets or returns the frequency associated with a Power Loss Segment.
<b>VB Syntax</b>	<i>object</i> . <b>Frequency</b> = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>object</i>	One of the following objects: <a href="#">PowerSensorCalFactorSegment</a> <a href="#">PowerSensorCalFactorSegmentPMAR</a> <a href="#">PowerLossSegment</a> <a href="#">PowerLossSegmentPMAR</a>
<i>value</i>	<b>(double)</b> – Frequency in units of Hz. This can be any non-negative value (limited by the maximum value of double).
<b>Return Type</b>	Double
<b>Default</b>	0
<b>Examples</b>	<pre>seg.Frequency = 6e9 'Write freq = seg.Frequency 'Read</pre>
<b><u>C++</u> Syntax</b>	HRESULT put_Frequency(double newVal); HRESULT get_Frequency(double *pVal);
<b>Interface</b>	One of the above objects.

---

Last Modified:

25-Aug-2009    Added PMAR



## FrequencyCenter Property

---

**Description** Sets and returns the center frequency of the main tones. Use with IMD sweep types:

- `naIMDToneCWSweep`
- `naIMDTonePowerSweep`
- `naIMDDeltaFrequencySweep`

**VB Syntax** *object.FrequencyCenter = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IISpectrum](#) Object

*value* (Double) Tone center frequency in Hz. Both the F1 and F2 tones MUST be within the frequency range of the PNA.

**Return Type** Double

**Default** 1.0 GHz

**Examples** `imd.FrequencyCenter = 2e9 'Write`

`value = imd.FrequencyCenter 'Read`

**C++ Syntax** HRESULT get\_FrequencyCenter(double \*pVal)  
HRESULT put\_FrequencyCenter(double \*pVal)

**Interface** ISweptIMD  
IISpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## FrequencyCenterCenter Property

---

**Description** Sets and returns the sweep center frequency when sweeping the main tones. Use with sweep type = **naIMDToneCenterFreqSweep**.

**VB Syntax** *imd.FrequencyCenterCenter = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) Center frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return  
Type** Double

**Default** 13.255 GHz

**Examples** `imd.FrequencyCenterCenter = 10e9 'Write`

`value = imd.FrequencyCenterCenter 'Read`

**C++ Syntax** HRESULT get\_FrequencyCenterCenter(double \*pVal)  
HRESULT put\_FrequencyCenterCenter(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## FrequencyCenterSpan Property

---

**Description** Sets and returns the frequency span when sweeping the main tones. Use with sweep type = **naIMDToneCenterFreqSweep**.

**VB Syntax** *imd.FrequencyCenterSpan* = *value*

**Variable**    [\(Type\)](#) - Description

*imd*    A [SweptIMD](#) Object

*value*    (Double) Frequency span in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return  
Type**    Double

**Default**    26.489 GHz

**Examples**    `imd.FrequencyCenterSpan = 10e9 'Write`

`value = imd.FrequencyCenterSpan 'Read`

**C++ Syntax**    HRESULT get\_FrequencyCenterSpan(double \*pVal)  
                  HRESULT put\_FrequencyCenterSpan(double newVal)

**Interface**    ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## FrequencyCenterStart Property

---

**Description** Sets and returns the start frequency when sweeping the main tones. Use with sweep type = **naIMDToneCenterFreqSweep**.

**VB Syntax** *imd.FrequencyCenterStart* = *value*

**Variable**    [\(Type\)](#) - Description

*imd*    A [SweptIMD](#) Object

*value*    (Double) Start frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return  
Type**    Double

**Default**    10.5 MHz

**Examples**    `imd.FrequencyCenterStart = 20e6` 'Write

`value = imd.FrequencyCenterStart` 'Read

**C++ Syntax**    HRESULT get\_FrequencyCenterStart(double \*pVal)  
                  HRESULT put\_FrequencyCenterStart(double newVal)

**Interface**    ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## FrequencyCenterStop Property

---

**Description** Sets and returns the stop frequency when sweeping the main tones. Use with sweep type = **naIMDToneCenterFreqSweep**.

**VB Syntax** *imd.FrequencyCenterStop = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) Stop frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Return Type** Double

**Default** 26.4995 MHz

**Examples** `imd.FrequencyCenterStop = 20e9 'Write`

`value = imd.FrequencyCenterStop 'Read`

**C++ Syntax** HRESULT get\_FrequencyCenterStop(double \*pVal)  
HRESULT put\_FrequencyCenterStop(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

Read / Write

## FrequencyList Property

---

### Description

**VB Syntax** *guidedCal.FrequencyList* = *value*

**Variable**    [\(Type\)](#) - Description

*guidedCal*    [GuidedCalibration](#) (object)

*value*    (Variant) -

**Return  
Type**    Variant

### Default

### Examples

```
Dim value  
value = MySMC.FrequencyList
```

**[C++ Syntax](#)**    HRESULT get\_FrequencyList(Variant \*freqList)  
                  HRESULT put\_FrequencyList(Variant freqList)

**Interface**    IGuidedCalibration

## FrequencyOffsetDivisor Property **Superseded**

---

**Description** This method is replaced by properties on the [FOMRange Object](#).  
Specifies (along with [FrequencyOffsetMultiplier](#)) the value to multiply by the stimulus.  
See other [Frequency Offset](#) properties

**VB Syntax** *object.FrequencyOffsetDivisor = value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)  
**or**  
CalSet (**object**) - Read-only property  
*value* (**Double**) - Divisor value. Range is 1 to 1000

**Return Type** Double

**Default** 1

**Examples**

```
chan.FrequencyOffsetDivisor = 2 'Write
```

```
fOffsetDiv = chan.FrequencyOffsetDivisor 'Read
```

**C++ Syntax** HRESULT get\_FrequencyOffsetDivisor(double\*pval)  
HRESULT put\_FrequencyOffsetDivisor(double newVal)

**Interface** IChannel2  
|CalSet3

## FrequencyOffsetFrequency Property **Superseded**

---

**Description** This method is replaced by properties on the [FOMRange Object](#).  
Specifies an absolute offset frequency in Hz. For mixer measurements, this would be the LO frequency. See other [Frequency Offset](#) properties.

**VB Syntax** *object.FrequencyOffsetFrequency = value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)

**or**

CalSet (**object**) - Read-only property

*value* (**Double**) - Offset value. Range is +/- 1000 GHz. (Offsets can be positive or negative.)

**Return  
Type** Double

**Default** 0 Hz

**Examples** `chan.FrequencyOffsetFrequency = 2 'Write`

`fOffsetFreq = chan.FrequencyOffsetFrequency 'Read`

**C++ Syntax** HRESULT get\_FrequencyOffsetFrequency(double\*pval)  
HRESULT put\_FrequencyOffsetFrequency(double newVal)

**Interface** IChannel2  
|CalSet3



## FrequencyOffsetMultiplier Property **Superseded**

---

**Description** This method is replaced by properties on the [FOMRange Object](#).  
Specifies (along with [FrequencyOffsetDivisor](#)) the value to multiply by the stimulus. See other [Frequency Offset](#) properties.

**VB Syntax** *object*.FrequencyOffsetMultiplier = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)

**or**

CalSet (**object**) - Read-only property

*value* (**Double**) - Multiplier value. Range is 1 to 1000

**Return  
Type** Double

**Default** 1

**Examples** `chan.FrequencyOffsetMultiplier = 2 'Write`

`fOffsetMult = chan.FrequencyOffsetMultiplier 'Read`

**C++ Syntax** HRESULT get\_FrequencyOffsetMultiplier (double\*pval);  
HRESULT put\_FrequencyOffsetMultiplier (double newVal);

**Interface** IChannel2  
|CalSet3

**FrequencyOffsetCWOverride Property** **Superseded**

---

**Description** This method is replaced by properties on the [FOMRange Object](#).

Establishes a fixed (CW) stimulus frequency while measuring the Response over a swept frequency range. For example, a fixed-frequency PNA stimulus may be applied to the RF input of a mixer whose local oscillator (LO) is being swept. Because the IF output of the mixer will be swept, the PNA receivers must also be swept.

See other [Frequency Offset](#) properties.

**VB Syntax** *object*.**FrequencyOffsetCWOverride** = *value*

**Variable** [\(Type\)](#) - **Description**

*object* Channel (**object**)

**or**

CalSet (**object**) - Read-only property

*value* (**Enum as NaStates**) - Choose from:

naOFF (0) - Turns CW override OFF

naON (1) - Turns CW override ON

**Return Type** Enum

**Default** 0 Hz

**Examples**

```
chan.FrequencyOffsetCWOverride = 1 'Write
```

```
fOffsetOV = chan.FrequencyOffsetCWOverride 'Read
```

**C++ Syntax** HRESULT get\_FrequencyOffsetCWOverride (tagNAStates \*pstate)  
 HRESULT put\_FrequencyOffsetCWOverride (tag NAStates newState)

**Interface** IChannel2  
 |CalSet3

**FrequencyOffsetState Property** **Superseded**

---

**Description** This method is replaced by properties on the [FOMRange Object](#).

Enables Frequency Offset on ALL measurements that are present on the active channel. This immediately causes the source and receiver to tune to separate frequencies. The receiver frequencies are specified with other channel and offset settings. To make the stimulus settings, use Channel Start, Stop Frequency properties. See other [Frequency Offset](#) properties.

Tip: To avoid unnecessary errors, first make other frequency offset settings. Then turn Frequency Offset ON.

**VB Syntax** *object*.**FrequencyOffsetState** = *value***Variable** [\(Type\)](#) - Description*object* Channel (**object**)**or**CalSet (**object**) - Read-only property*value* (**Enum as NaStates**) - Choose from:**naOFF** (0) - Turns Frequency Offset OFF**naON** (1) - Turns Frequency Offset ON**Return  
Type** Enum**Default** naOFF (0)**Examples** `chan.FrequencyOffsetState = naON 'Write``Foffset = chan.FrequencyOffsetState 'Read`**C++ Syntax** HRESULT FrequencyOffsetState (tag NAStates \*pState);  
HRESULT FrequencyOffsetState (tag NAStates newState)**Interface** IChannel2  
|CalSet3

## FrequencySpan Property

---

**Description** Sets or returns the frequency span of the channel.  
Sets or returns the frequency span of the segment.

**VB Syntax** *object*.**FrequencySpan** = *value*

**Variable** [\(Type\)](#) - Description

*object* A Channel (**object**)  
**or**  
A Segment (**object**)

*value* (**double**) - Frequency span in Hertz. Choose any number between the **minimum** and **maximum** frequencies of the analyzer.

**Return Type** Double

**Default** Full frequency span of the analyzer

**Examples** `chan.FrequencySpan = 4.5e9 'sets the frequency span of a linear sweep for the channel object -Write`

`freqspan = chan.FrequencySpan 'Read`

**C++ Syntax** HRESULT get\_FrequencySpan(double \*pVal)  
HRESULT put\_FrequencySpan(double newVal)

**Interface** IChannel  
ISegment

## Shape Property

---

**Description** Specifies the shape of the gate filter.

**VB Syntax** *gat.Shape* = *value*

**Variable** [\(Type\)](#) - Description

*gat* A Gating (**object**)

*value* (**enum** NAGateShape) - Choose from:

**0** - naGateShapeMaximum

**1** - naGateShapeWide

**2** - naGateShapeNormal

**3** - naGateShapeMinimum

**Return Type** NAGateShape

**Default** 2 - Normal

**Examples** `gat.Shape = naGateShapeMaximum 'Write`

`filterShape = gat.Shape 'Read`

**C++ Syntax** HRESULT get\_Shape(tagNAGateShape \*pVal)  
HRESULT put\_Shape(tagNAGateShape newVal)

**Interface** IGating

## Type (gate) Property

---

**Description** Specifies the type of gate filter used.

**VB Syntax** *gat.Type* = *value*

**Variable** [\(Type\)](#) - Description

*gat* A Gating (**object**)

*value* (**enum NAGateType**) - Choose from:

**0 - naGateTypeBandpass** - Includes (passes) the range between the start and stop times.

**1 - naGateTypeNotch** - Excludes (attenuates) the range between the start and stop times.

**Return Type** NAGateType

**Default** Bandpass

**Examples** `gate.Type = naGateTypeNotch 'Write`

`filterType = gate.Type 'Read`

**C++ Syntax** HRESULT get\_Type(tagNAGateType \*pVal)  
HRESULT put\_Type(tagNAGateType newVal)

**Interface** IGating

## GPIBAddress Property

---

**Description** Sets and returns the PNA GPIB address bus.

**VB Syntax** *app.GPIBAddress (bus) = value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*bus* (Short Integer) GPIB bus. MUST be set to 0.

*value* (Short Integer) GPIB Address on the PNA. Choose a value between 0 and 30.

**Return Type** Short Integer

**Default** 16

**Examples** `address=app.GPIBAddress(0) 'Read`

`app.GPIBAddress(0)=16 'Write`

**C++ Syntax** HRESULT get\_GPIBAddress(short busIndex, short\* address);

HRESULT put\_GPIBAddress(short busIndex,short address);

**Interface** IApplication8

## GPIBMode Property

---

**Description** Changes the analyzer to a GPIB system controller or a talker/listener on the bus. The analyzer must be the controller if you want to use it to send commands to other instruments. The analyzer must be a talker/listener if you want to send it commands from another PC.

**Note:** This command has no affect in PNAs with dedicated Controller and Talker/Listener GPIB connectors.

**VB Syntax** *app.GPIBMode value*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

*value* **(enum NAGPIBMode)** -Choose either:  
**0** - naTalkerListener - the analyzer is a talker / listener  
**1** - naSystemController - the analyzer is the system controller

**Return Type** Long Integer

**Default** 0 - naTalkerListener

**Examples** `app.GPIBMode = naTalkerListener 'Write`

`mode = app.GPIBMode 'Read`

**C++ Syntax** HRESULT get\_GPIBMode(tagGPIBModeEnum\* eGpibMode)  
HRESULT put\_GPIBMode(tagGPIBModeEnum eGpibMode)

**Interface** IApplication

---

Last Modified:

10-Nov-2008    Removed link to obsolete topic.



## GPIBPortCount Property

---

**Description** Returns the number of GPIB ports that are present on the PNA rear-panel.

**VB Syntax** *value* = *cap*.**GPIBPortCount**

**Variable** [\(Type\)](#) - **Description**

*value* (Long) - Variable to store the returned integer value of the number of GPIB ports.

*cap* A [Capabilities](#) (**object**)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.GPIBPortCount 'Read`

**C++ Syntax** HRESULT get\_GPIBPortCount(long \* gpibPorts );

**Interface** ICapabilities3

---

Last Modified:

23-Oct-2008    Removed 1.1 GHz description

## HandshakeEnable Property

---

**Description** Turns handshake ON / OFF.

To enable handshake, the main trigger enable must ALSO be set using [Enable](#).

When ON, PNA acquisition waits indefinitely for the input line to be asserted before continuing with the acquisition.

**Note:** Use on PNA-X ONLY. Other models do NOT have an Aux Input.

**VB Syntax** *auxTrig.HandshakeEnable = state*

**Variable** [\(Type\)](#) - Description

*auxTrig* An [AuxTrigger](#) (object)

*state* (boolean) -

**True** - Handshake enabled

**False** - Handshake NOT enabled

**Return  
Type** Boolean

**Default** False

**Exaamples** `auxTrig.HandshakeEnable = True 'Write`

`value = auxTrig.HandshakeEnable 'Read`

**C++ Syntax** HRESULT get\_HandshakeEnable(VARIANT\_BOOL \* enable);  
HRESULT put\_HandshakeEnable(VARIANT\_BOOL enable);

**Interface** IAuxTrigger

---

Last Modified:

5-Sep-2008 Added Note

14-Dec-2006 MX New topic

## HighestOrderProduct Property

---

**Description** Returns the highest order product that can be measured by SweptIMD.

**VB Syntax** *value* = *imd*.HighestOrderProduct

**Variable** [\(Type\)](#) - Description

*value* (Long Integer) Variable in which to store the returned value.

*imd* A [SweptIMD](#) Object

**Return Type** Integer

**Default** Always returns 9

**Examples** `value = imd.HighestOrderProduct 'Read`

**[C++ Syntax](#)** HRESULT get\_HighestOrderProduct(long \*pVal)

**Interface** ISweptIMD

---

Last Modified:

19-Sep-2008    MX New topic

## ID Property

---

**Description** Returns the test set ID number. For GPIB testsets, the ID is equivalent to the GPIB address. For testset I/O testsets, the ID is the base address of the testset (0 for the first testset, 1 for the second, and so on).

**VB Syntax** *value* = *tset.ID*

**Variable** [\(Type\)](#) - Description

*value* (Long) variable to store the returned information.

*testsets(1)* A [TestsetControl](#) object.

OR

An [E5091Testset](#) object.

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `value = testset1.ID`

[See E5091A Example Program](#)

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_ID(long \*idNumber);

**Interface** ITestsetControl  
IE5091Testset

## IDString Property

---

<b>Description</b>	Returns the ID of the analyzer, including the Model number, Serial Number, and the Software revision number.  <b>Note:</b> Beginning with Rev 6.01, this command now returns the software revision with 6 digits instead of 4. For example, A.06.01.02.
<b>VB Syntax</b>	<i>value</i> = <i>app</i> .IDString
<b>Variable</b>	<u>(Type)</u> - Description
<i>app</i>	An <a href="#">Application</a> (object)
<i>value</i>	(string) - variable to contain the returned ID string
<b>Return Type</b>	String
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>id = app.IDString</code>
<b>C++ Syntax</b>	HRESULT IDString(BSTR* IDString)
<b>Interface</b>	IApplication

## IFBandwidthOption Property

---

**Description** Enables the IFBandwidth to be set on individual sweep segments. This property must be set True **before** *seg*.[IFBandwidth](#) = *value* is sent. Otherwise, this command will be ignored.

**VB Syntax** *segs*.IFBandwidthOption = *value*

**Variable** [\(Type\)](#) - Description

*segs* A Segments collection (**object**)

*value* (**boolean**)

**True** - Enables variable IFBandwidth setting for segment sweep

**False** - Disables variable IFBandwidth setting for segment sweep

**Return  
Type** Boolean

**Default** False

**Examples** `segs.IFBandwidthOption = True 'Write`

`IFOption = IFBandwidthOption 'Read`

**C++ Syntax** HRESULT get\_IFBandwidthOption(VARIANT\_BOOL \*pVal)  
HRESULT put\_IFBandwidthOption(VARIANT\_BOOL newVal)

**Interface** ISegments

## IFBandwidth Property

---

**Description** Sets or returns the IF Bandwidth of the channel.  
Sets or returns the IF Bandwidth of the segment.  
Returns the IF Bandwidth used in the Cal Set

**VB Syntax** *object*.IFBandwidth = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)

**or**

Segment (**object**)

**or**

CalSet (**object**) - Read-only property

*value* (**double**) - IF Bandwidth in Hz. The list of valid IF Bandwidths is different depending on the PNA model. [\(Click to see the lists.\)](#) If an invalid number is specified, the analyzer will round up to the closest valid number.

**Return Type** Double

**Default** See [Preset IFBW](#) for your PNA model.

**Examples**

```
chan.IFBandwidth = 3e3 'sets the IF Bandwidth of for the channel object
to 3 kHz. -Write
seg.IFBandwidth = 5 'sets the IF Bandwidth of the segment to 5 Hz. -
Write
```

```
ifbw = chan.IFBandwidth -Read
```

**C++ Syntax** HRESULT get\_IFBandwidth(double \*pVal);  
HRESULT put\_IFBandwidth(double newVal);

**Interface** IChannel  
ISegment  
|CalSet3

## IFFilterSampleCount Property

---

**Description** **Note:** This setting applies only to the E836X Opt. H11.

Sets or retrieves the number of taps in the IF filter when the [IFFilterSource](#) property is set to `nalIFFilterSourceManual`.

**VB Syntax** *IfConfig.IFFilterSampleCount = value*

**Variable** [\(Type\)](#) - Description

*IfConfig* [IFConfiguration](#) (object)

*value* **(long)** – The IF filter sample count. The minimum and maximum allowed values for this property can vary by model number, but can be queried using the [MinimumIFFilterSampleCount](#) and [MaximumIFFilterSampleCount](#) properties.

**Return Type** Long Integer

**Default** PNA Model number dependent

**Examples** `App.ActiveChannel.IFConfiguration.IFFilterSampleCount = 200 'Write`

`variable = App.ActiveChannel.IFConfiguration.IFFilterSampleCount 'Read`

[See an example program](#)

**C++ Syntax** `HRESULT get_IFFilterSampleCount( long * pSampleCount );`  
`HRESULT put_IFFilterSampleCount ( long sampleCount );`

**Interface** IIFConfiguration2



## IFFilterSamplePeriod Property

---

**Description** **Note:** This setting applies only to the E836X Opt. H11.

Sets or returns the IF filter sample period time. This time is only used if the [IFFilterSamplePeriodMode](#) is set to naManual.

**VB Syntax** *IfConfig.IFFilterSamplePeriod = value*

**Variable** [\(Type\)](#) - **Description**

*IfConfig* [IFConfiguration](#) (object)

*value* **(double)** – The sample period time in seconds. Valid sample period times can be queried using the [IFFilterSamplePeriodList](#) property.

**Return Type** Double

**Default** PNA Model number dependent.

**Examples** `App.ActiveChannel.IFConfiguration.IFFilterSamplePeriod = .000006 'Write`

`variable = App.ActiveChannel.IFConfiguration.IFFilterSamplePeriod 'Read`

[See an example program](#)

**C++ Syntax** HRESULT get\_IFFilterSamplePeriod( double \* pSamplePeriod );  
 HRESULT put\_IFFilterSamplePeriod ( double samplePeriod );

**Interface** IIFConfiguration2

## IFFilterSamplePeriodList Property

---

**Description** Retrieves the list of available IF filter sample periods for the instrument.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *variable* = *IfConfig*.IFFilterSamplePeriodList

**Variable** [\(Type\)](#) - [Description](#)

*variable* (Array) An array of permissible values that can be passed to the [IFFilterSamplePeriod](#) property.

*IfConfig* [IFConfiguration \(object\)](#)

**Return Type** Array

**Default** Not applicable

**Examples**

```
Dim Variable
Variable =
App.ActiveChannel.IFConfiguration.IFFilterSamplePeriodList 'Read
MsgBox "First IF Sample Period Value: " & Variable(0)
```

[See an example program](#)

**C++ Syntax** HRESULT get\_IFFilterSamplePeriodList( SAFEARRAY \*\* ppSamplePeriodList );

**Interface** IIFConfiguration2

## IFFilterSamplePeriodMode Property

---

**Description** Sets or returns the IF filter sample period mode.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *IfConfig.IFFilterSamplePeriodMode = value*

**Variable** [\(Type\)](#) - Description

*IfConfig* [IFConfiguration \(object\)](#)

*value* **(enum NAModes)** -

**0 - naAuto** - IF filter sample period is chosen automatically.

**1 - naManual** - the IF filter sample period is the value specified by the [IFSamplePeriod](#) property.

**Return  
Type** NAModes

**Default** 0 - naAuto

**Examples** `App.ActiveChannel.IFConfiguration.IFFilterSamplePeriodMode = naAuto`  
`'Write`

`variable = App.ActiveChannel.IFConfiguration.IFFilterSamplePeriodMode`  
`'Read`

[See an example program](#)

**C++ Syntax** HRESULT get\_IFFilterSamplePeriodMode( tagNAModes \* pMode);  
HRESULT put\_IFFilterSamplePeriodMode ( tagNAModes mode );

**Interface** IIFConfiguration2

## IFFilterSource Property

---

**Description** Sets or retrieves type of IF filter to be used.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *IfConfig.IFFilterSource* = *value*

**Variable** [\(Type\)](#) - **Description**

*object* [IFConfiguration \(object\)](#)

*value* **(NAIFFilterSource)** – The type of filter to use. Choose from:

**naIFFilterSourceAuto** – The IF filter type is automatically chosen.

**naIFFilterSourceManual** – The IF filter is a predetermined shape where the [IFFilterSampleCount](#) determines the number of taps in the filter.

**Return Type** NAIFFilterSource

**Default** naIFFilterSourceAuto

**Examples** `App.ActiveChannel.IFConfiguration.IFFilterSource =  
naIFFilterSourceManual 'Write`

`Variable = App.ActiveChannel.IFConfiguration.IFFilterSource 'Read`

[See an example program](#)

**C++ Syntax** HRESULT get\_IFFilterSource( tagNAIFFilterSource \* pFilterSource );  
HRESULT put\_IFFilterSource(tagNAIFFilterSource filterSource );

**Interface** IIFConfiguration2

## IFGainLevel Property

---

**Description** **Note:** This setting applies only to the E836x and selected N5230A models.

Manually sets the gain level for the specified receiver.

**VB Syntax** *IfConfig*.**IFGainLevel** (*id*) = *value*

**Variable** [\(Type\)](#) - **Description**

*IfConfig* An [IFConfiguration](#) (object)

*id* Receiver for which to set the gain level. Choose from: 'A', 'B', 'R1', 'R2'.

*value* (long Integer) Gain Level. Choose from:

0 - Lowest gain setting

1

2 - Highest gain setting

**Return Type** Long Integer

**Default** 0 (Lowest setting)

**Examples** `IfConfig.IFGainLevel ("A") = 1`

**C++ Syntax** HRESULT get\_IFGainLevel (BSTR IDString, \*GainLevel)

HRESULT put\_IFGainLevel (BSTR IDString, GainLevel)

**Interface** IFConfiguration

---

Last Modified:

14-Apr-2008 Clarified models

21-Sep-2007 Fixed example

## IFGainMode Property

---

**Description** **Note:** This setting applies only to the E836x and selected N5230A models.

Sets the gain state for ALL receivers to Auto or Manual.

**VB Syntax** *IfConfig*.**IFGainMode** (*id*) = *value*

**Variable** [\(Type\)](#) - Description

*IfConfig* An [IFConfiguration](#) (object)

*id* Receivers for which to set the state. Choose 'ALL'.

*value* (enum as NAModes)

Choose from:

0 - naAUTO

1 - naMANUAL (use [IFGainLevel Property](#) to manually set gain level)

**Return Type** NAModes

**Default** 0 - naAUTO

**Examples** `IfConfig.IFGainMode("ALL") = naAUTO`

**C++ Syntax** HRESULT get\_IFGainMode (BSTR IDString, NAModes \*gainMode)  
HRESULT put\_IFGainMode (BSTR IDString, NAModes gainMode)

**Interface** IIFConfiguration

---

Last Modified:

14-Apr-2008    Clarified models

## IFGateEnable Property

---

**Description** Sets or retrieves the state of the IF Gate.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *IfConfig*.IFGateEnable = *value*

**Variable** [\(Type\)](#) - **Description**

*IfConfig* [IFConfiguration \(object\)](#)

*value* **(Boolean)** – The state of the IF Gate.

**True** – The IF Gate is in use.

**False** – The IF Gate is not in use.

**Return Type** Boolean

**Default** False

**Examples** `App.ActiveChannel.IFConfiguration.IFGateEnable = True 'Write`  
`variable = App.ActiveChannel.IFConfiguration.IFGateEnable 'Read`

[See an example program](#)

**C++ Syntax** HRESULT get\_IFGateEnable( VARIANT\_BOOL \* pGateEnabled );  
HRESULT put\_IFGateEnable( VARIANT\_BOOL gateEnabled );

**Interface** IIFConfiguration

## IFFrequency Property

---

**Description** Sets and returns the IF frequency for ALL receiver paths being used for the specified channel. To set this frequency, [IFFrequencyMode Property](#) must be set to OFF (Manual).

**VB Syntax** *IfConfig.IFFrequency = value*

**Variable** [\(Type\)](#) - **Description**

*IfConfig* An [IFConfiguration](#) (object)

*value* (double) IF Frequency. Use [MaximumIFFrequency](#) and [MinimumIFFrequency](#) to determine the range of value for this command.

**Return Type** Double

**Default** 9 MHz

**Examples** `IfConfig.IFFrequency = 9.3e6`

**C++ Syntax** HRESULT get\_IFFrequency (double \*pVal);  
HRESULT put\_IFFrequency (double pVal);

**Interface** IFConfiguration3

---

Last Modified:

18-Jun-2007    MX New topic



## IFFrequencyMode Property

---

**Description** Sets and returns method for specifying the way the IF Frequency is determined.

**VB Syntax** *IfConfig.IFFrequencyMode = value*

**Variable** [\(Type\)](#) - Description

*IfConfig* An [IFConfiguration](#) (object)

*value* (enum as NAModes) IF Frequency mode. Choose from:

**0 - naAUTO** PNA determines the setting for the IF frequency.

**1 - naMANUAL** (use [IFFrequency Property](#) to manually set frequency.

**Return Type** Enum

**Default** 0 naAuto

**Examples** `IfConfig.IFFrequencyMode = naMANUAL`

**C++ Syntax** HRESULT get\_IFFrequencyMode (tagNAModes\* pdspMode);  
HRESULT put\_IFFrequencyMode (tagNAModes\* pdspMode);

**Interface** IFConfiguration3

---

Last Modified:



18-Jun-2007    MX New topic

## IFSideband Property

---

**Description** When two LOs are used, sets or returns whether to select the sum or difference for the IF1 product. (Input + or - LO1 = IF1)



- This setting corresponds to the   buttons on LO1.
- Also set [OutputSideband](#) to LOW or HIGH to determine the output frequency of the mixer.
- This setting is ignored when ONE LO is used.
- If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.IFSideband =value*

**Variable** [\(Type\)](#) - **Description**

*obj* A [Converter Object](#)

*value* **(enum as ConverterSideBand)** - Choose from:

**0** or **naLowSide** Minus (-) on the Mixer setup dialog

**1** or **naHighSide** Plus (+) on the Mixer setup dialog

**Return Type** Enum as ConverterSideBand

**Default** 0 - **naLowSide**

**Examples** `conv.IFSideband = naLowSide`

**C++ Syntax** HRESULT get\_IFSideband(ConverterSideBand \*pVal)  
HRESULT put\_IFSideband(ConverterSideBand newVal)

**Interface** IConverter

---

Last Modified:



26-Mar-2009    New topic

## IFSideband Property

---

**Description** When two LOs are used, sets or returns whether to select the sum or difference for the IF1 product. (Input + or - LO1 = IF1)



- This setting corresponds to the   buttons on LO1.
- Also set [OutputSideband](#) to LOW or HIGH to determine the output frequency of the mixer.
- This setting is ignored when ONE LO is used.
- If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** `mixer.IFSideband =value`

**Variable** [\(Type\)](#) - Description

*mixer* A Mixer **(object)**

*value* **(enum as FCASideBand)** - Choose from:  
**0 or LOW** Minus (-) on the Mixer setup dialog  
**1 or HIGH** Plus (+) on the Mixer setup dialog

**Return Type** Enum as FCASideBand

**Default** 0 - LOW

**Examples** `Print mixer.IFSideband` 'prints the value of the IFSideband

**C++ Syntax** `HRESULT get_IFSideband(FCASideBand *pVal)`  
`HRESULT put_IFSideband(FCASideBand newVal)`

**Interface** IMixer

---

Last Modified:

2-Oct-2008 Clarification

4-Mar-2008 Added note.

## IFSourcePath Property

---

<b>Description</b>	<p><b>Note:</b> This setting applies only to the E836X Opt. H11.</p> <p>Sets the source path of the specified receiver. An error is returned if <i>&lt;id&gt;</i> is not valid, or if <i>&lt;value&gt;</i> is not valid for the specified <i>&lt;id&gt;</i>.</p>
<b>VB Syntax</b>	<i>IfConfig</i> . <b>IFSourcePath</b> ( <i>id</i> ) = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>IfConfig</i>	An <a href="#">IFConfiguration</a> (object)
<i>id</i>	<p>Receiver for which to set the gain level. Choose from: 'A', 'B', 'R1', 'R2'..</p> <p><b>Note:</b> The A and R1 receivers are always switched together. B and R2 are also always switched together. For example, if you specify "A", R1 will also be switched.</p>
<i>value</i>	<p>(Enum as <b>NAIFSourcePath</b>) Source path. Choose from:</p> <p>0 - <b>naNormalIFPath</b> - the PNA decides the appropriate IF input paths.</p> <p>1 - <b>naExternalIFPath</b> - always use the rear panel IF inputs.</p>
<b>Return Type</b>	NAIFSourcePath
<b>Default</b>	0 - naNormalIFPath
<b>Examples</b>	<code>IfConfig.IFSourcePath('A') = naNormalIFPath</code>
<b>C++ Syntax</b>	<p>HRESULT get_IFSourcePath (BSTR IDString, naIFSourcePath *IFSourcePath)</p> <p>HRESULT put_IFSourcePath (BSTR IDString, naIFSourcePath IFSourcePath)</p>
<b>Interface</b>	IIFConfiguration

## IFStartFrequency Property

---

<b>Description</b>	Sets or returns the start frequency value of the mixer IF frequency.  Only applies to 2 stage mixers.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>mixer</i> .IFStartFrequency = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>mixer</i>	A Mixer <b>(object)</b>
<i>value</i>	<b>(double)</b> - Frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Print mixer.IFStartFrequency 'prints the value of the IFStartFrequency</pre>
<b>C++ Syntax</b>	HRESULT get_IFStartFrequency(double *pVal) HRESULT put_IFStartFrequency(double newVal)
<b>Interface</b>	IMixer

---

Last Modified:

4-Mar-2008    Added note.

## IFStopFrequency Property

---

<b>Description</b>	Sets or returns the stop frequency value of the mixer IF frequency.  Only applies to 2 stage mixers.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>mixer</i> .IFStopFrequency = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>mixer</i>	A Mixer <b>(object)</b>
<i>value</i>	<b>(double)</b> - IF stop frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Print mixer.IFStopFrequency 'prints the value of the IFStopFrequency</pre>
<b>C++ Syntax</b>	HRESULT get_IFStopFrequency(double *pVal) HRESULT put_IFStopFrequency(double newVal)
<b>Interface</b>	IMixer

---

Last Modified:

4-Mar-2008    Added note.

## ImpedanceStates Method

---

**Description** Sets the number of impedance states to use during calibrated measurements.

**VB Syntax** *noise.ImpedanceStates = value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(double)** - Impedance states. Choose between 4 and the maximum number allowed by the noise tuner device. If the specified number exceeds the capability of the device, the measurement will use the maximum number of states the device allows.

**Return Type** Double

**Default** 4

**Examples** `noise.ImpedanceStates = 10 'Write`

`AvgNoise = noise.ImpedanceStates 'Read`

**C++ Syntax** HRESULT get\_ImpedanceStates(double\* pVal)  
HRESULT put\_ImpedanceStates(double newVal)

**Interface** INoiseFigure

---

Last Modified:

29-May-2007 MN New topic

## ImpulseWidth Property

---

**Description** Sets or returns the Impulse Width of Time Domain transform windows

**VB Syntax** *trans*.**ImpulseWidth** = *value*

**Variable** [\(Type\)](#) - Description

*trans* A [Transform](#) (object)

*value* **(double)** - Impulse Width in seconds. Range of settings depends on the frequency range of your analyzer.

**Return Type** Double

**Default** .98 / Default Span

**Examples** `trans.ImpulseWidth = 200e-12 'sets the Impulse width of a transform window -Write`

`IW = trans.ImpulseWidth 'Read`

**C++ Syntax** HRESULT get\_ImpulseWidth(double \*pVal)  
HRESULT put\_ImpulseWidth(double newVal)

**Interface** ITransform



## IMToneIFBandwidth Property

---

**Description** Sets and returns the IF Bandwidth for measurement of the intermodulation products.

**VB Syntax** *imd.IMToneIFBandwidth = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) IF Bandwidth in Hz. Choose from: 1 | 2 | 3 | 5 | 7 | 10 | 15 | 20 | 30 | 50 | 70 | 100 | 150 | 200 | 300 | 500 | 700 | 1k | 1.5k | 2k | 3k | 5k | 7k | 10k | 15k | 20k | 30k | 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k

[Learn more about setting IFBW for IMD.](#)

If an invalid number is specified, the analyzer will round up to the closest valid number.

**Return Type** Double

**Default** 1 kHz

**Examples** `imd.IMToneIFBandwidth = 2e3 'Write`

`value = imd.IMToneIFBandwidth 'Read`

**C++ Syntax** HRESULT get\_IMToneIFBandwidth(double \*pVal)  
HRESULT put\_IMToneIFBandwidth(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## Include2ndOrderProduct Property

---

**Description** Sets and returns whether to include the second order products in the calibration. These frequencies of these products can be far from the main tones.

**VB Syntax** *imd.Include2ndOrderProduct* = *value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMDCal](#) (object)

*value* (Boolean) Choose from:

**False** - Do NOT include 2nd order products

**True** - Include 2nd order products

**Return  
Type** Boolean

**Default** **False** - Do NOT include 2nd order products

**Examples** `imd.Include2ndOrderProduct = true 'Write`

`incl = imd.Include2ndOrderProduct 'Read`

**C++ Syntax** HRESULT get\_Include2ndOrderProduct(VARIANT\_BOOL \* Val)  
HRESULT put\_Include2ndOrderProduct(VARIANT\_BOOL newVal)

**Interface** ISweptIMD

---

Last Modified:

9-Sep-2008    MX New topic

## IndexState Property

---

**Description** Determines the control of Material Handler connector Pin 20.

**VB Syntax** *handler.IndexState = value*

**Variable** [\(Type\)](#) - Description

*handler* **(object)** - A [Handler I/O](#) object

*value* **(boolean)**

False - Pin 20 is controlled by Output Port B6

True - Pin 20 is controlled by the Index signal

**Return Type** Boolean

**Default** False

**Examples**

```
handler.IndexState = False 'Write  
bState = handler.IndexState 'Read
```

**[C++ Syntax](#)** HRESULT put\_IndexState (BOOL \*pVal);  
HRESULT get\_IndexState (BOOL newVal);

**Interface** IHWMaterialHandlerIO2

## InputA Property - **Obsolete**

---

**Description** This property has NO replacement and no longer works correctly. (Sept. 2004)

Sets a Port Extension value for Receiver A

**VB Syntax** `portExt.InputA = value`

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.InputA = 10e-6 'Write`

`inA = portExt.InputA 'Read`

**C++ Syntax** HRESULT get\_InputA(double \*pVal)  
HRESULT put\_InputA(double newVal)

**Interface** IPortExtension

## InputB Property - **Obsolete**

---

**Description** This property has NO replacement and no longer works correctly. (Sept. 2004)

Sets the Port Extension value for Receiver B

**VB Syntax** `portExt.InputB = value`

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.InputB = 10e-6 'Write`

`inB = portExt.InputB 'Read`

**C++ Syntax** HRESULT get\_InputB(double \*pVal)  
HRESULT put\_InputB(double newVal)

**Interface** IPortExtension

## InputC Property **Obsolete**

---

**Description** This property has NO replacement and no longer works correctly. (Sept. 2004)

Sets the Port Extension value for Receiver C

**VB Syntax** *portExt.InputC = value*

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.InputC = 10e-6 'Write`

`inC = portExt.InputC 'Read`

**C++ Syntax** HRESULT get\_InputC(double \*pVal)  
HRESULT put\_InputC(double newVal)

**Interface** IPortExtension

## InputDenominator Property

---

<b>Description</b>	Sets or returns the denominator value of the Input Fractional Multiplier.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .InputDenominator = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
	<i>obj</i> A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
	<i>value</i> <b>(Long)</b> - Input denominator value.
<b>Return Type</b>	Long
<b>Default</b>	1
<b>Examples</b>	<pre>Print mixer.InputDenominator 'prints the value of the InputDenominator</pre>
<b>C++ Syntax</b>	HRESULT get_InputDenominator(long *pVal) HRESULT put_InputDenominator(long newVal)
<b>Interface</b>	IMixer IConverter

---

### Last Modified:

2-Feb-2009	Added converter
4-Mar-2008	Added note.

## InputFixedFrequency Property

---

**Description** Sets or returns the mixer fixed Input frequency value.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.InputFixedFrequency = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(double)** - Input Fixed Frequency in Hertz.

**Return Type** Double

**Default** Not Applicable

**Examples** `mixer.InputFixedFrequency = 1e9`

**C++ Syntax** HRESULT get\_InputFixedFrequency(double \*pVal)  
HRESULT put\_InputFixedFrequency(double newVal)

**Interface** IMixer6  
IConverter

---

Last Modified:

2-Feb-2009 Added converter

4-Mar-2008 Added note.



## IsInputGreaterThanLO Property

---

**Description** Specifies whether to use the Input frequency that is greater than the LO or less than the LO. To learn more, see the [mixer setup dialog box help](#).

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** `obj.IsInputGreaterThanLO (LO) = bool`

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*LO* (Integer) - LO stage number

Choose from 1 (default) or 2

*bool* **(Boolean)** -

**True** - Use the Input that is Greater than the specified LO.

**False** - Use the Input that is Less than the specified LO.

**Return Type** Boolean

**Default** True

**Examples** `mixer.IsInputGreaterThanLO(1) = True`

**C++ Syntax** HRESULT get\_IsInputGreaterThanLO(VARIANT\_BOOL \* val);  
HRESULT put\_IsInputGreaterThanLO(VARIANT\_BOOL val);

**Interface** IMixer2  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## InputNumerator Property

---

<b>Description</b>	Sets or returns the numerator value of the Input Fractional Multiplier.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .InputNumerator = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
	<i>obj</i> A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
	<i>value</i> <b>(Long)</b> - Input numerator value.
<b>Return Type</b>	Long
<b>Default</b>	1
<b>Examples</b>	<pre>Print mixer.InputNumerator 'prints the value of the InputNumerator</pre>
<b>C++ Syntax</b>	HRESULT get_InputNumerator(long *pVal) HRESULT put_InputNumerator(long newVal)
<b>Interface</b>	IMixer IConverter

---

Last Modified:

2-Feb-2009    Added converter

4-Mar-2008    Added note.

## InputPower Property

---

**Description** Sets or returns the value of the Input Power.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.InputPower = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)  
Or  
A [Converter Object](#)

*value* **(double)** - Input power in dBm.

**Return Type** Double

**Default** -15 dBm for IMixer  
-20 dBm for IConverter

**Examples** `Print mixer.InputPower` 'prints the value of the InputPower

**C++ Syntax** HRESULT get\_InputPower(double \*pVal)  
HRESULT put\_InputPower(double newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009 Added converter

4-Mar-2008 Added note.

## InputRangeMode Property

---

**Description** Sets or returns the Input sweep mode.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.InputRangeMode = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(Enum as MixerRangeMode)** - Input sweep mode. Choose from:

0 - **mixSwept**

1 - **mixFixed**

**Return Type** Enum

**Default** 0 - mixSwept

**Examples** `mixer.InputRangeMode = mixSwept`

**C++ Syntax** HRESULT get\_InputRangeMode(long \*pVal)  
HRESULT put\_InputRangeMode(long newVal)

**Interface** IMixer6  
IConverter

---

Last Modified:

2-Feb-2009 Added converter

4-Mar-2008 Added note.

## InputStartFrequency Property

---

<b>Description</b>	Sets and returns the start frequency value of the mixer Input frequency.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj.InputStartFrequency = value</i>
<b>Variable</b>	<b><a href="#">(Type)</a> - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Input start frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Start frequency of the PNA
<b>Examples</b>	<pre>mixer.InputStartFrequency = Start_Freq</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_InputStartFrequency(double *pVal) HRESULT put_InputStartFrequency(double newVal)
<b>Interface</b>	IMixer IConverter

---

### Last Modified:

2-Feb-2009    Added converter

4-Mar-2008    Added note.

## InputStartPower Property

---

<b>Description</b>	Sets and returns the Start Power value of the mixer Input Power.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .InputStartPower = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>obj</i>	A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Input start Power in dBm.
<b>Return Type</b>	Double
<b>Default</b>	Start Power of the PNA
<b>Examples</b>	<pre>mixer.InputStartPower = 0</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_InputStartPower(double *pVal) HRESULT put_InputStartPower(double newVal)
<b>Interface</b>	ICConverter

---

Last Modified:

2-Feb-2009    New topic

## InputStopFrequency Property

---

<b>Description</b>	Sets and returns the stop frequency value of the mixer Input frequency.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj.InputStopFrequency = value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Input stop frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Stop frequency of the PNA
<b>Examples</b>	<code>mixer.InputStopFrequency = Stop_Freq</code>
<b>C++ Syntax</b>	HRESULT get_InputStopFrequency(double *pVal) HRESULT put_InputStopFrequency(double newVal)
<b>Interface</b>	IMixer IConverter

---

Last Modified:

2-Feb-2009    Added Converter

4-Mar-2008    Added note.

## InputStopPower Property

---

<b>Description</b>	Sets and returns the Stop Power value of the mixer Input Power.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .InputStopPower = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>obj</i>	A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Input Stop Power in dBm.
<b>Return Type</b>	Double
<b>Default</b>	Stop Power of the PNA
<b>Examples</b>	<pre>mixer.InputStopPower = 0</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_InputStopPower(double *pVal) HRESULT put_InputStopPower(double newVal)
<b>Interface</b>	ICConverter

---

Last Modified:

2-Feb-2009    New topic



## InternalTestsetPortCount Property

---

**Description** Returns the number of PNA test ports. This does not include the ports on an [external test set](#).

**VB Syntax** *value* = *cap*.InternalTestsetPortCount

**Variable** [\(Type\)](#) - **Description**

*value* (Long) - Variable to store the returned number of PNA test ports.

*cap* A [Capabilities](#) (**object**)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.InternalTestsetPortCount 'Read`

**C++ Syntax** HRESULT get\_InternalTestsetPortCount(long \*numPorts );

**Interface** ICapabilities

## Interpolate Correction Property

---

**Description** Turns ON and OFF correction interpolation which calculates new error terms when stimulus values change after calibration.

When this property is ON and error correction is being applied, the calibration subsystem attempts to interpolate the error terms whenever the stimulus parameters are changed.

When this property is OFF under the same circumstances, error correction is turned OFF.

**VB Syntax** *meas.InterpolateCorrection = value*

**Variable** [\(Type\)](#) - **Description**

*meas* A [Measurement](#) (object)

*value* (**boolean**) - Choose from:

**True** - Turns correction interpolation ON

**False** - Turns correction interpolation OFF

**Return  
Type** Boolean

**Default** True

**Examples** `meas.InterpolateCorrection = False`

`calInterpolate = InterpolateCorrection 'Read`

**C++ Syntax** HRESULT get\_InterpolateCorrection(boolean \*pVal)  
HRESULT put\_InterpolateCorrection(boolean newVal)

**Interface** IMeasurement

## Interpolated Property

---

**Description** Turns marker Interpolation ON and OFF. Marker interpolation enables X-axis resolution beyond the discrete data values. The analyzer will calculate the x and y-axis data values between discrete data points. Use meas.[Interpolate](#) to change interpolation of **all** markers in a measurement. This command will override the measurement setting.

**VB Syntax** `mark.Interpolated = value`

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**boolean**)  
**False** - Turns interpolation OFF  
**True** - Turns interpolation ON

**Return Type** Boolean

**Default** True

**Examples** `mark.Interpolated = True 'Write`

`interpolate = mark.Interpolated 'Read`

**C++ Syntax** HRESULT get\_Interpolated(VARIANT\_BOOL \*pVal)  
HRESULT put\_Interpolated(VARIANT\_BOOL newVal)

**Interface** IMarker

**InterpolateNormalization Property** **Superseded**

**Description** **Note:** This property is replaced by [DoReceiverPowerCal Method](#).

Turns ON and OFF normalization interpolation which calculates new divisor data when stimulus values change after normalization.

When this property is ON and normalization is being applied, the Normalization algorithm attempts to interpolate the divisor data whenever the stimulus parameters are changed.

When this property is OFF under the same circumstances, normalization is turned OFF.

Normalization is currently supported only on measurements of unratiod power for the purpose of performing a receiver power calibration.

**VB Syntax** *meas.InterpolateNormalization* = *value*

**Variable** [\(Type\)](#) - Description

*meas* **(object)** - A Measurement object

*value* **(boolean)**

False – Turns normalization interpolation OFF

True – Turns normalization interpolation ON

**Return  
Type** Boolean

**Default** False -OFF

**Examples** `meas.InterpolateNormalization = False 'Write`

`normalized = meas.InterpolateNormalization 'Read`

**C++ Syntax** HRESULT put\_InterpolateNormalization(VARIANT\_BOOL bState);  
HRESULT get\_InterpolateNormalization(VARIANT\_BOOL \*bState);

**Interface** IMeasurement

## Interrupt Property

---

**Description** Reads the boolean that represents the state of the Interrupt In line (pin 13) on the external test set connector.

**VB Syntax** *value* = *ExtIO*.**Interrupt**

**Variable** [\(Type\)](#) - **Description**

*value* **(boolean)** - Variable to store the returned data

*ExtIO* **(object)** - An ExternalTestSetIO object

**Return Type** Boolean  
**False** - indicates the line is being held at a TTL High  
**True** - indicates the line is being held at a TTL Low

**Default** Not Applicable

**Examples** `value = ExtIO.Interrupt`

**C++ Syntax** HRESULT get\_Interrupt( VARIANT\_BOOL\* bValue);

**Interface** IHWExternalTestSetIO

## IsContinuous Property

---

**Description** Returns whether or not a channel is in continuous mode. To set the channel to continuous mode, use [Continuous Method](#).

**VB Syntax** *value* = *chan*.**IsContinuous**

**Variable** [\(Type\)](#) - **Description**

*value* **(boolean)** - Choose either:

**False** - Channel trigger is NOT set to continuous.

**True** - Channel trigger IS set to continuous.

*chan* Channel **(object)**

**Return Type** Boolean

**Default** Not Applicable

**Examples** `trig = chan.IsContinuous 'Read`

**C++ Syntax** HRESULT get\_IsContinuous ( VARIANT\_BOOL \*bContinuous);

**Interface** IChannel3

## IsECALModuleFoundEx Property

---

<b>Description</b>	<p>This property replaces <a href="#">IsECALModuleFound Property</a>.</p> <p>Returns true or false depending on whether communication was established between the PNA and the specified ECal module.</p>
<b>VB Syntax</b>	<i>modFound</i> = <i>cal</i> .IsECALModuleFoundEx ( <i>module</i> )
<b>Variable</b>	<a href="#">(Type) - Description</a>
<i>modFound</i>	<p><b>(boolean)</b> - Variable to store the returned test result.</p> <p><b>True</b> - The PNA identified the presence of the specified ECal module.</p> <p><b>False</b> - The PNA did NOT identify the presence of the specified ECal module.</p>
<i>cal</i>	<b>(object)</b> - A Calibrator object
<i>module</i>	<p><b>(long integer)</b> ECal module.</p> <p>Choose from modules <b>1</b> through <b>8</b></p> <p>Use <a href="#">GetECALModuleInfoEx</a> to return the model and serial number of each module.</p>
<b>Return Type</b>	Boolean
<b>Default</b>	Not applicable
<b>Examples</b>	<pre>Set cal = pna.ActiveChannel.Calibrator moduleFound = cal.IsECALModuleFoundEx(1)</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_IsECALModuleFoundEx( long moduleNumber, VARIANT_BOOL *bModuleFound);
<b>Interface</b>	ICalibrator4

## IsFrequencyOffsetPresent Property

---

**Description** Returns a value indicating the presence of Frequency Offset Option 080 in the remote PNA.

**VB Syntax** *value* = *cap*.IsFrequencyOffsetPresent

**Variable** [\(Type\)](#) - Description

*value* (Boolean) - Variable to store the returned value  
True - Frequency Offset Option 080 is present  
False - Frequency Offset Option 080 is not present

*cap* A [Capabilities](#) (object)

**Return Type** Boolean

**Default** Not Applicable

**Examples** `value = cap.IsFrequencyOffsetPresent (1) 'Read`

**C++ Syntax** HRESULT get\_IsFrequencyOffsetPresent(VARIANT\_BOOL \* present );

**Interface** ICapabilities



## IsHold Property

---

**Description** Returns whether or not a channel is in hold mode. To set the channel to hold mode, use [Hold Method](#).

**VB Syntax** *value* = *chan*.**IsHold**

**Variable** [\(Type\)](#) - **Description**

*value* **(boolean)** - Choose either:  
**False** - Channel trigger is NOT set to hold.  
**True** - Channel trigger IS set to hold.

*chan* Channel **(object)**

**Return Type** Boolean

**Default** Not Applicable

**Examples** `trig = chan.IsHold 'Read`

**C++ Syntax** HRESULT get\_IsHold ( VARIANT\_BOOL \*bHold);

**Interface** IChannel3

## IsMarkerOn Property

---

**Description** Returns whether or not a marker was used for the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.IsMarkerOn (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Boolean)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Return Type** **(String)**

**Default** Not Applicable

**Examples** `data= embedLO.IsMarkerOn 3 'read`

**C++ Syntax** HRESULT IsMarkerOn(long sweep, VARIANT\_BOOL\* markerOn);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## IsolationIncrementAveraging Property

---

**Description** Value by which to increment (increase) the channel's averaging factor during measurement of isolation in an ECal calibration.

**Note:** If <value> is greater than 1 and the channel currently has averaging turned OFF, averaging will be turned ON only during the isolation measurements and with the averaging factor equal to <value>.

**VB Syntax** `cal.IsolationIncrementAveraging = value`

**Variable** [\(Type\)](#) - Description

*cal* A [Calibrator](#) (object)

*value* **(Long)** Incremental Averaging factor. The maximum averaging factor is 65536 (2<sup>16</sup>).

**Return Type** Long Integer

**Default** 8

**Examples** `oCal.IsolationAveragingIncrement = 16 'Write`

`avgIncr = oCal.IsolationAveragingIncrement ' Read`

**C++ Syntax** HRESULT get\_IsolationAveragingIncrement(long \*pVal);  
HRESULT put\_IsolationAveragingIncrement(long newVal);

**Interface** ICalibrator7

---

Last Modified:

16-Apr-2007    MX New topic

## IsOn Property

---

**Description** Sets and returns the ON |OFF state of Embedded LO.

**VB Syntax** *obj.IsOn = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (**object**) or  
A [ConverterEmbeddedLO](#) (**object**)

*value* (**Boolean**)  
**False** - Turns Embedded LO OFF  
**True** - Turns Embedded LO ON

**Return Type** (**Boolean**)

**Default** False (OFF)

**Examples** `embedLO.IsOn = True 'write`

`data= embedLO.IsOn 'read`

**C++ Syntax** HRESULT get\_IsOn( VARIANT\_BOOL\* IsOn);  
HRESULT put\_IsOn( VARIANT\_BOOL IsOn);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## IsReceiverStepAttenuatorPresent Property

---

**Description** Returns a value indicating the presence of Receiver step attenuators in the remote PNA.

**VB Syntax** `value = cap.IsReceiverStepAttenuatorPresent (n)`

**Variable** [\(Type\)](#) - Description

*value* **(Boolean)** - Variable to store the returned value.  
True - Receiver step attenuators are present.  
False - Receiver step attenuators are not present.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - port number to query for step attenuators

**Return Type** Boolean

**Default** Not Applicable

**Examples** `value = cap.IsReceiverStepAttenuatorPresent (1) 'Read`

**C++ Syntax** HRESULT get\_IsReceiverStepAttenuatorPresent(long portNumber, VARIANT\_BOOL \* present );

**Interface** ICapabilities

## IsReferenceBypassSwitchPresent Property

---

**Description** Returns a value indicating the presence of a Reference Bypass Switch in the remote PNA.

**VB Syntax** `value = cap.IsReferenceBypassSwitchPresent (n)`

**Variable** [\(Type\)](#) - Description

*value* (Boolean) - Variable to store the returned value.

True - Reference Bypass Switch is present.

False - Reference Bypass Switch is not present.

*cap* A [Capabilities](#) (object)

*n* **(Long)** - port number to query for reference bypass switch

**Return Type** Boolean

**Default** Not Applicable

**Examples** `value = cap.IsReferenceBypassSwitchPresent (1) 'Read`

**C++ Syntax** HRESULT get\_IsReferenceBypassSwitchPresent(long portNumber, VARIANT\_BOOL \* present );

**Interface** ICapabilities

Read-only

## IsSParameter Property

---

**Description** Returns true if measurement represents an S-Parameter

**VB Syntax** *value* = *meas*.IsSparameter

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**Boolean**)

True - measurement is an S-Parameter

False - measurement is NOT an S-Parameter

**Return Type** Boolean

**Default** True

**Examples** `print app.IsSparameter`

**C++ Syntax** HRESULT IsSparameter( [out, retval] VARIANT\_BOOL \* bVal);

**Interface** IMeasurement2

## IterationNumber Property

---

**Description** Sets and returns the maximum leveling sweep iterations to be used in order to achieve the tolerance setting.

**VB Syntax** *RxLevel.IterationNumber(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the IterationNumber for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Long Integer) Max iterations. Choose a value between 1 and 25.

**Return Type** (Long Integer)

**Default** 5

**Examples** `rxLevel.IterationNumber (1) = 10 ' Write`

`value = rxLevel.IterationNumber 2' Read`

**C++ Syntax** HRESULT get\_IterationNumber(long port, long\* pVal);  
HRESULT put\_IterationNumber(long port, long newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic



## IterationsTolerance Property

---

**Description** This command, along with [MaximumIterationsPerPoint Property](#) deal with ADJUSTMENTS made to the source power.

Sets the maximum desired deviation from the sum of the [test port power](#) and the [offset](#) value. Power readings will continue to be made, and source power adjusted, until a reading is within this tolerance value or the [max number of readings](#) has been met. The last value to be read is the valid reading for that data point.

The following two commands allow for settling of power READINGS.

[ReadingsPerPoint Property](#)

[ReadingsTolerance Property](#)

**VB Syntax** *pwrCal.IterationsTolerance = value*

**Variable** [\(Type\)](#) - Description

*pwrCal* **(object)** - A [SourcePowerCalibrator](#) (object)

*value* **(Double)** – Tolerance value in dBm. Choose any number between 0 and 5

**Return Type** Double

**Default** .05 dB

**Examples**

```
Set powerCalibrator = pna.SourcePowerCalibrator
powerCalibrator.IterationsTolerance = .1 'Write
ReadTol = powerCalibrator.IterationsTolerance 'Read
```

**C++ Syntax** HRESULT get\_IterationsTolerance( double \*pVal);  
HRESULT put\_IterationsTolerance( double newVal);

**Interface** ISourcePowerCalibrator3

---

Last Modified:

17-Apr-2007    Clarified verbage

## KaiserBeta Property

---

**Description** Sets or returns the Kaiser Beta of Time Domain transform windows

**VB Syntax** *trans.KaiserBeta* = *value*

**Variable** [\(Type\)](#) - Description

*trans* A Transform (**object**)

*value* (**single**) - Kaiser Beta. Choose any number between **0** and **13**.

**Return Type** Single

**Default** 0

**Examples** `trans.KaiserBeta = 6 'sets the Kaiser Beta of a transform window  
-Write`

`KB = trans.KaiserBeta 'Read`

**C++ Syntax** HRESULT get\_KaiserBeta(float \*pVal)  
HRESULT put\_KaiserBeta(float newVal)

**Interface** ITransform

## L0 Property

---

**Description** Sets and Returns the L0 (L-zero) value (the first inductance value) for the calibration standard.  
To set the other inductance values, use [L1](#), [L2](#), [L3](#).

**VB Syntax** `calstd.L0 = value`

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for L0 in femtohenries(1E-15)

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.L0 = 15 'Write the value of L0 = 15 femtohenries`

`Induct0 = calstd.L0 'Read the value of L0`

**C++ Syntax** HRESULT get\_L0(float \*pVal)  
HRESULT put\_L0(float newVal)

**Interface** ICalStandard

## L1 Property

---

**Description** Sets and Returns the L1 value (the second inductance value) for the calibration standard.  
To set the other inductance values, use [L0](#), [L2](#), [L3](#).

**VB Syntax** *calstd.L1 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for L1.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.L1 = 15 'Write the value of L1`

`Induct1 = calstd.L1 'Read the value of L1`

**C++ Syntax** HRESULT get\_L1(float \*pVal)  
HRESULT put\_L1(float newVal)

**Interface** ICalStandard

## L2 Property

---

**Description** Sets and Returns the L2 value (the third inductance value) for the calibration standard.  
To set the other inductance values, use [L0](#), [L1](#), [L3](#).

**VB Syntax** *calstd.L2 = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for L2.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.L2 = 15 'Write the value of L2.`

`Induct2 = calstd.L2 'Read the value of L2`

**C++ Syntax** HRESULT get\_L2(float \*pVal)  
HRESULT put\_L2(float newVal)

**Interface** ICalStandard

## L3 Property

---

**Description** Sets and Returns the L3 value (the third inductance value) for the calibration standard.  
To set the other inductance values, use [L0](#), [L1](#), [L2](#).

**VB Syntax** `calstd.L3 = value`

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for L3.

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.L3 = 15 'Write the value of L3.`

`Induct3 = calstd.L3 'Read the value of L3`

**C++ Syntax** HRESULT get\_L3(float \*pVal)  
HRESULT put\_L3(float newVal)

**Interface** ICalStandard

## Label Property

---

**Description** Sets and Returns the label for the calibration standard. The label is used to prompt the user to connect the specified standard.

**VB Syntax** *calstd.Label* = *value*

**Variable** [\(Type\)](#) - **Description**

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**string**) - between 1 and 12 characters long. Cannot begin with a numeric.

**Return  
Type** String

**Default** Not Applicable

**Examples** `calstd.Label = "Short" 'Write`

`stdLabel = calstd.Label 'Read`

**[C++ Syntax](#)** HRESULT get\_Label(BSTR \*pVal)  
HRESULT put\_Label(BSTR newVal)

**Interface** ICalStandard

## Label Property

---

**Description** Sets or gets the display label for a given channel/testset combination. The label appears in a status bar at the bottom of the PNA display when the [ShowProperties](#) property is set to TRUE.

**VB Syntax** *tset.Label(chNum) = value*

**Variable** [\(Type\)](#) - Description

*tset* A [TestsetControl](#) object.

Obtained from the [ExternalTestsets](#) collection.

*chNum* **(Integer)** Channel number of the measurement.

*value* **(String)** The text of the label.

**Return Type** String

**Default** None

**Examples** 'The following sets the label for channel 5 corresponding to a given testset object.'

```
testset1.label(5) = 'High-power output'
```

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_Label(long channelNum, BSTR \*pLabel);  
HRESULT put\_Label(long channelNum, BSTR label);

**Interface** ITestsetControl



## LANConfiguration Property

---

**Description** Returns information about the current status of the PNA's computer networking configuration. This is the same set of information that is returned in an NA\_IPConfiguration data structure by the GetIPConfigurationStruct method.

**VB Syntax** *value* = *app*.**LANConfiguration**

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to contain the PNA's LAN configuration string

*app* An [Application](#) **(object)**

**Return Type** Comma-delimited string

**Default** Not Applicable

**Examples** `networkConfigInfo = app.LANConfiguration`

**C++ Syntax** HRESULT get\_LANConfiguration (BSTR \* pStrConfig);

**Interface** IApplication13

---

Last Modified:

2-Jun-2008    MX New topic

## LastModified Property

---

<b>Description</b>	Returns the time stamp of the last modification to this Cal Set.  This property always returns a time stamp based on the Greenwich Mean Time (GMT) regardless of the local time zone setting of the PNA.  The Cal Set properties that are viewed on the PNA user interface are converted to the local time of the PNA.
<b>VB Syntax</b>	<i>value</i> = <i>Object</i> . <b>LastModified</b>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>object</i>	Channel ( <b>object</b> )  <b>or</b> CalSet ( <b>object</b> ) - Read-only property
<i>value</i>	<b>(Variant)</b> – Variable to store the time stamp.
<b>Return Type</b>	Variant
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>date = CalSet.LastModified 'Read</code>
<b>C++ Syntax</b>	HRESULT get_LastModified(VARIANT* datetime)
<b>Interface</b>	ICalSet3

---

Last Modified:

25-Apr-2007    Added GMT note

## LevelingIFBW Property

---

<b>Description</b>	Sets and returns the IFBW to be used for leveling sweeps. Enable separate IFBW for leveling sweeps using <a href="#">FastMode Property</a> .
<b>VB Syntax</b>	<i>RxLevel.LevelingIFBW(srcPort) = value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>RxLevel</i>	A <a href="#">ReceiverLeveling</a> Object
<i>srcPort</i>	(Long Integer) Source port for which to set the LevelingIFBW for Receiver Leveling. <b>Note:</b> If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use <a href="#">chan.getPortNumber</a> to translate the string into a port number. To learn more see <a href="#">Remotely Specifying a Source Port</a>
<i>value</i>	(Double) IFBW for leveling sweeps in Hz. The list of valid IF Bandwidths is different depending on the PNA model. <a href="#">See list</a> . If an invalid number is specified, the PNA will round up to the closest valid number.
<b>Return Type</b>	(Double)
<b>Default</b>	100 kHz
<b>Examples</b>	<pre>rxLevel.LevelingIFBW (1) = 1e3 ' Write</pre> <pre>value = rxLevel.LevelingIFBW 2 ' Read</pre>
<b>C++ Syntax</b>	HRESULT get_LevelingIFBW(long port, double* pVal); HRESULT put_LevelingIFBW(long port, double newVal);
<b>Interface</b>	IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## LimitTestFailed Property

---

**Description** Returns the results of limit testing for the measurement.

**VB Syntax** *testFailed* = *meas*.**LimitTestFailed**

**Variable** [\(Type\)](#) - **Description**

*testFailed* **(boolean)** Variable to store the returned value

**False** - Limit Test Passed

**True** - Limit Test Failed

*meas* A Measurement **(object)**

**Return Type** Boolean

**Default** False returned if there is no testing in progress.

**Examples**

```
Dim testRes As Boolean
testRes = meas.LimitTestFailed
MsgBox (testRes)
```

**C++ Syntax** HRESULT get\_LimitTestFailed(VARIANT\_BOOL\*  
truelIfFailed)

**Interface** IMeasurement

## EndStimulus Property

---

**Description** When constructing a limit line, specifies the stimulus value for the end of the segment.

**VB Syntax** *limtseg*.**EndStimulus** = *value*

**Variable**    [\(Type\)](#) - Description

*limtseg*    A LimitSegment (**object**)

*value*    (**double**) - End Stimulus X-axis value. No units

**Return Type** Double

**Default** 0

**Examples**

```
Set limtseg = meas.LimitTest(1)
limtseg.EndStimulus = 8e9 'Write

EndStim = limtseg.EndStimulus 'Read
```

**C++ Syntax** HRESULT get\_EndStimulus(double \*pVal)  
HRESULT put\_EndStimulus(double newVal)

**Interface** ILimitSegment

## EndResponse Property

---

**Description** When constructing a limit line, specifies the amplitude value at the end of the limit segment.

**VB Syntax** *limtseg*.**EndResponse** = *value*

**Variable**    [\(Type\)](#) - Description

*limts*    A LimitSegment (**object**)

*value*    (**double**) - Y-axis value of the End Response limit. No units

**Return Type** Double

**Default** 0

**Examples**

```
Set limtseg = meas.LimitTest(1)
limtseg.EndResponse = 10 'Write

EndResp = limtseg.EndResponse 'Read
```

**C++ Syntax** HRESULT get\_EndResponse(double \*pVal)  
HRESULT put\_EndResponse(double newVal)

**Interface** ILimitSegment

## Type (limit) Property

---

**Description** Specifies the Limit Line type.

**VB Syntax** *limt(index).Type* = value

**Variable** [\(Type\)](#) - Description

*limt* A LimitSegment (**object**)

*index* (**variant**) - Limit line number in the LimitTest collection

*value* (**enum NALimitSegmentType**) - Limit Line type. Choose from:  
**0 - naLimitSegmentType\_OFF** - turns limit line OFF  
**1 - naLimitSegmentType\_Maximum** - limit line fails with a data point ABOVE the line  
**2 - naLimitSegmentType\_Minimum** - limit line fails with a data point BELOW the line

**Return Type** Long Integer

**Default** 0 - OFF

**Examples**

```
Set limts = meas.LimitTest
limts.Type = naLimitSegmentType_Maximum 'Write

limitType = limts.Type 'Read
```

**C++ Syntax** HRESULT put\_Type(tagNALimitSegmentType \*pVal)  
HRESULT get\_Type(tagNALimitSegmentType newVal)

**Interface** ILimitSegment

## LineDisplay Property

---

**Description** Turns the display of limit lines ON or OFF. To turn limit TESTING On and OFF, use [State Property](#).

**Note:** Trace data must be ON to view limit lines

**VB Syntax** *limitst.LineDisplay = state*

**Variable** [\(Type\)](#) - Description

*limitst* A LimitTest (**object**)

*state* (**boolean**)

**False** - Turns the display of limit lines OFF

**True** - Turns the display of limit lines ON

**Return Type** Long Integer

**Default** **True**

**Examples** `Limttest.LineDisplay = true 'Write`

`lineDsp = Limttest.LineDisplay 'Read`

**C++ Syntax** HRESULT get\_LineDisplay(VARIANT\_BOOL \*pVal)  
HRESULT put\_LineDisplay(VARIANT\_BOOL newVal)

**Interface** ILimitTest



## LoadCharFromFile Property

---

**Description** Sets and returns whether a Mixer characterization file is to be loaded. Specify and load the filename with [CharFileName Property](#)

**VB Syntax** *VMC.LoadCharFromFile = bool*

**Variable** [\(Type\)](#) - Description

*VMC* [VMCType](#) (object)

*bool* (Boolean)

**True** - Load from file

**False** - Perform mixer characterization

**Return Type** Boolean

**Default** False

**Examples** `value = VMC.LoadCharFromFile`

**[C++ Syntax](#)** HRESULT put\_LoadCharFromFile(VARIANT\_BOOL bLoadCharFromFile);  
HRESULT get\_LoadCharFromFile(VARIANT\_BOOL \*bLoadCharFromFile);

**Interface** VMCType

## LoadPort Property

---

**Description** Returns the load port number associated with an S-parameter reflection measurement. If the measurement is not a reflection S-parameter, the number returned by this property will have no meaning.

**VB Syntax** *loadPort* = *meas*.**LoadPort**

**Variable** [\(Type\)](#) - **Description**

*loadPort* **(long integer)** - The reflection measurement's load port number.

*meas* A Measurement **(object)**

**Return Type** Long Integer

**Default** Not Applicable

**Examples**

```
Set meas = pna.ActiveMeasurement
loadPort = meas.LoadPort
```

**C++ Syntax** HRESULT get\_LoadPort(long \*pPortNumber);

**Interface** IMeasurement

Write/Read

## LocalLockoutState Property

---

**Description** Prevents use of the mouse, keyboard, and front panel while your program is running. Use of these controls while this property is set TRUE causes an error message on the PNA display. To prevent these messages, see [About Error Messages](#).

**VB Syntax** `app.LocalLockoutState = bool`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*bool* (**boolean**) -Choose either:

**False** - User Interface is NOT locked out.

**True** - User Interface IS locked out.

**Return Type** Boolean

**Default** False

**Examples** `app.LocalLockoutState = True 'Write`

`block = app.LocalLockoutState 'Read`

**C++ Syntax** HRESULT get\_LocalLockoutState(VARIANT\_BOOL \*State);  
HRESULT put\_LocalLockoutState(VARIANT\_BOOL \*State);

**Interface** IApplication4

## Locator Property

---

**Description** Specifies the location of the power meter / sensor that is used during a source power calibration.

Use [Interface Property](#) to specify the type of interface.

**VB Syntax** *pwrMtrInterface.Locator = value*

**Variable** [\(Type\)](#) - Description

*pwrMtrInterface* **(object)** - A [PowerMeterInterface](#) (object)

*value* **(string)** Location of the power meter / sensor, depending on the type of [Interface](#).

- For **naGPIB**, address of the power meter. Choose any integer between 0 and 30.
- For **naUSB**, the ID string of the power sensor. Use [USBPowerMeterCatalog Property](#) to see a list of ID strings of connected power sensors.
- For **naLAN**, the hostname or IP address of the power meter.

**Return Type** String

**Default** Not applicable

**Examples** `pwrMeterInterface.Locator = "13" 'GPIB address`

`pwrMeterInterface.Locator = "Agilent Technologies, U2000A, MY12345678" 'USB ID string`

`pwrMeterInterface.Locator = "mymeter.agilent.com" 'LAN`

**C++ Syntax** `HRESULT put_Locator( BSTR pValue );`  
`HRESULT get_Locator(BSTR* pValue );`

**Locator** IPowerMeterInterface

---

Last Modified:

29-Sep-2008    Removed Rev from Example

24-Jul-2007    MX New topic

## LODeltaFound Property

---

**Description** Returns the LO frequency delta from the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**LODeltaFound** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.LODeltaFound 3 'read`

**C++ Syntax** HRESULT LODeltaFound(long sweep, double\* deltaLO);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## LODenominator Property

---

<b>Description</b>	Sets or returns the denominator value of the LO Fractional Multiplier.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj.LODenominator (n) = value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(Long)</b> - LO denominator value
<i>n</i>	<b>(Long)</b> - LO stage number Choose from 1 or 2
<b>Return Type</b>	Long
<b>Default</b>	1
<b>Examples</b>	<pre>Print mixer.LODenominator(1) 'prints the value of the first LODenominator</pre>
<b>C++ Syntax</b>	HRESULT get_LODenominator(long *pVal) HRESULT put_LODenominator(long newVal)
<b>Interface</b>	IMixer IConverter

---

Last Modified:

2-Feb-2009    Added Converter

4-Mar-2008    Added note.

## LOFixedFrequency Property

---

**Description** Sets or returns the LO frequency value.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.LOFixedFrequency (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)  
Or  
A [Converter Object](#)

*value* **(double)** - LO Frequency in Hertz.

*n* **(Long)** - LO stage number  
Choose from 1 or 2

**Return Type** Double

**Default** 0 Hz

**Examples** `Print mixer.LOFixedFrequency(1)` 'prints the value of the first LO fixed frequency

**C++ Syntax** HRESULT get\_LOFixedFrequency(double \*pVal)  
HRESULT put\_LOFixedFrequency(double newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## LOFrequencyDelta Property

---

**Description** Sets and returns LO Frequency Delta. There is usually no need to set this value. Read this value to determine the difference between the LO Frequency that is stated in the Mixer dialog box and the last measured LO Frequency.

**VB Syntax** *obj.LOFrequencyDelta = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* **(Double)** LO Frequency delta in Hertz.

**Return Type** **(Double)**

**Default** Not Applicable

**Examples**

```
embedLO.LOFrequencyDelta = 0 'write
```

```
value = embedLO.LOFrequencyDelta 'read
```

**C++ Syntax** HRESULT get\_LOFrequencyDelta(double\* val);  
HRESULT put\_LOFrequencyDelta(double val);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic



**LogMagnitudeOffset Property** **Superseded**

**Description** **Note:** This property is replaced by [DoReceiverPowerCal Method](#).

**For Receiver Calibration** - Sets or returns the value to offset the normalized unratioed power measurement data. The unratioed power measurement is effectively calibrated to the power level specified by the value of LogMagnitudeOffset as soon as the [Normalization property](#) is set to ON after calling the [DataToDivisor](#) method.

To offset the data trace magnitude a specified value, use [MagnitudeOffset Property](#)

**VB Syntax** *meas.LogMagnitudeOffset = value*

**Variable** [\(Type\)](#) - Description

*meas* **(object)** - A [Measurement](#) object

*value* **(double)** - Offset value in dBm.

**Return Type** Double

**Default** 0

**Examples** `meas.LogMagnitudeOffset = -10 'Write (-10 dBm)`

```
calpower = meas.LogMagnitudeOffset 'Read
meas.DataToDivisor 'Store meas data as measurement divisor
meas.Normalize = 1 'Measurement is now calibrated to -10 dBm
```

**C++ Syntax** HRESULT put\_LogMagnitudeOffset(double newVal);  
 HRESULT get\_LogMagnitudeOffset(double \*pVal);

**Interface** IMeasurement

## LOName Property

---

**Description** Sets or returns the name of the PNA internal source or external source to use as the LO in an FCA measurement.

**VB Syntax** *obj.LOName (n) =value*

**Variable** [\(Type\) - Description](#)

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*n* **(Long)** - LO stage number

Choose from 1 or 2

*value* **(string)** - LO Source name. Use [SourcePort Property](#) to return a list of valid source ports. An external source must be configured and selected to be valid. [Learn more about external source configuration.](#)

**Note:** If the port is defined by a string name, such as an external source or one of the Source 2 outputs on the 2-port, 2-source PNA-x model, then you must use [cap.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

**Return Type** String

**Default** "Not controlled"

**Examples** `mixer.LOName(1) = "MySource"`

**C++ Syntax** HRESULT get\_LOName(string \*pVal)  
HRESULT put\_LOName(string newVal)

**Interface** MIXer  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

24-Apr-2008 Added string names note

23-Jul-2007 Updated for external source config.



## LONumerator Property

---

**Description** Sets or returns the numerator value of the LO Fractional Multiplier.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.LONumerator (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(Long)** - LO denominator value

*n* **(Long)** - LO stage number

Choose from 1 or 2

**Return Type** Long

**Default** 1

**Examples** `Print mixer.LONumerator(1)` 'prints the value of the  
first LO Numerator

**C++ Syntax** HRESULT get\_LONumerator(long \*pVal)  
HRESULT put\_LONumerator(long newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## LOPower Property

---

**Description** Sets or returns the value of LO Power.

**VB Syntax** *obj.LOPower (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*n* **(Long)** - LO stage number

Choose from 1 or 2

*value* **(double)** - LO Power in dBm.

**Return Type** Double

**Default** -10dBm

**Examples** `Print mixer.LOPower(1)` 'prints the value of the LO Power

**C++ Syntax** HRESULT get\_LOPower(double \*pVal)  
HRESULT put\_LOPower(double newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009    added converter

## LORangeMode Property

---

**Description** Sets or returns the LO sweep mode.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.LORangeMode (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)  
Or  
A [Converter Object](#)

*n* **(Long)** - LO stage number.  
Choose from 1 or 2

*value* **(Enum as MixerRangeMode)** - LO sweep mode. Choose from:  
0 - **mixSwept**  
1 - **mixFixed**

**Return Type** Enum

**Default** 0 - mixSwept

**Examples** `mixer.LORangeMode(1)=mixSwept`

**C++ Syntax** HRESULT get\_LORangeMode(long \*pVal)  
HRESULT put\_LORangeMode(long newVal)

**Interface** IMixer3  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## Loss Property

---

**Description** Sets and Returns the insertion loss for the calibration standard.

**VB Syntax** *calstd.loss = value*

**Variable** [\(Type\)](#) - **Description**

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Insertion loss in Gohms / sec. (Giga Ohms per second of electrical delay)

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.loss = 3.5 'Write 3.5 Gohms of loss`

`stdLoss = calstd.loss 'Read the value of Loss`

**C++ Syntax** HRESULT get\_Loss(float \*pVal)  
HRESULT put\_Loss(float newVal)

**Interface** ICalStandard

## Loss (Power Segment) Property

---

<b>Description</b>	Sets or returns the loss value associated with a PowerLossSegment.
<b>VB Syntax</b>	<i>lossSeg.Loss = value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>lossSeg</i>	A <a href="#">PowerLossSegment</a> (object) A <a href="#">PowerLossSegmentPMAR</a> (object)
<i>value</i>	<b>(double)</b> – Loss value in dB. Loss is entered as a POSITIVE number.
<b>Return Type</b>	Double
<b>Default</b>	0
<b>Examples</b>	<pre>lossSeg.Loss = 0.5 'Write lossVal = lossSeg.Loss 'Read</pre>
<b>C++ Syntax</b>	HRESULT put_Loss(Double newVal); HRESULT get_Loss(Double *pVal);
<b>Interface</b>	IPowerLossSegment IPowerLossSegmentPMAR

---

### Last Modified:

27-Aug-2009	Added PMAR (9.0)
2-Jun-2008	Clarified negative value for loss



## LOStage Property

---

**Description** Sets or returns the number of LO stages present in the mixer.

**VB Syntax** *obj.LOStage = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(Long)** - Number of LO stages. Choose from **1** or **2**

**Return Type** Long

**Default** 1

**Examples** `mixer.LOStage = 1 'sets the LO Stage value to 1`

**C++ Syntax** HRESULT get\_LOStage(long \*pVal)  
HRESULT put\_LOStage(long newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009    Added converter

## LOStartFrequency Property

---

**Description** Sets or returns the LO start frequency value. This command can only be used with SMC (not VMC) measurements.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.LOStartFrequency (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(double)** - LO Start Frequency in Hertz.

*n* **(Long)** - LO stage number

Choose from 1 or 2

**Return Type** Double

**Default** Not Applicable

**Examples** `Print mixer.LOStartFrequency(1)` 'prints the value of the first LO start frequency

**C++ Syntax** HRESULT get\_LOStartFrequency(long id, double \*pVal)  
HRESULT put\_LOStartFrequency(long id, double newVal)

**Interface** IMixer3  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## LOStartPower Property

---

**Description** Sets or returns the Start value of a LO Power sweep.. Also set [imdx.SweepType](#) to naIMDLOPowerSweep.

**VB Syntax** *obj.LOStartPower (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Converter Object](#)

*n* **(Long)** - LO stage number. Choose 1

*value* **(double)** - LO start power in dBm.

**Return Type** Double

**Default** -20 dBm

**Examples**

```
convtr.LOStartPower(1) = -5 'Sets the LO Power sweep start value
start = convtr.LOStartPower(1) 'Reads the start value
```

**C++ Syntax** HRESULT get\_LOStartPower(long LOStage, double \*pVal)  
HRESULT put\_LOStartPower(long LOStage, double newVal)

**Interface** IConverter

---

Last Modified:

27-Mar-2009    MX New topic

## LOStopFrequency Property

---

**Description** Sets or returns the LO stop frequency value. This command can only be used with SMC (not VMC) measurements.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.LOStopFrequency (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*value* **(double)** - LO Stop Frequency in Hertz.

*n* **(Long)** - LO stage number

Choose from 1 or 2

**Return Type** Double

**Default** Not Applicable

**Examples** `Print mixer.LOStopFrequency(1)` 'prints the value of the first LO stop frequency

**C++ Syntax** HRESULT get\_LOStopFrequency(long id, double \*pVal)  
HRESULT put\_LOStopFrequency(long id, double newVal)

**Interface** IMixer3  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## LOStopPower Property

---

**Description** Sets or returns the Stop value of a LO Power sweep. Also set [imdx.SweepType](#) to naIMDLOPowerSweep.

**VB Syntax** *obj.LOStopPower (n) = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Converter Object](#)

*n* **(Long)** - LO stage number. Choose 1

*value* **(double)** - LO Stop power in dBm.

**Return Type** Double

**Default** -10 dBm

**Examples**

```
convtr.LOStopPower(1) = -5 'Sets the LO Power sweep Stop value
Stop = convtr.LOStopPower(1) 'Reads the Stop value
```

**C++ Syntax** HRESULT get\_LOStopPower(long LOStage, double \*pVal)  
HRESULT put\_LOStopPower(long LOStage, double newVal)

**Interface** IConverter

---

Last Modified:

27-Mar-2009    MX New topic

## LXIDeviceIDState Property

---

**Description** Opens and closes the [LAN Status dialog](#) with the LAN Status Indicator showing IDENTIFY.

The PNA supports this capability to satisfy a requirement of the LAN eXtensions for Instrumentation (LXI) standard. Changing the value of this property is the same operation that occurs when clicking the Toggle LXI Identification button on the Welcome web page presented by the PNA web server.

**VB Syntax** *app.LXIDeviceIDState = bool*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*bool* (boolean) Choose from:

**True** - Displays the [LAN Status dialog](#) with the Status Indicator showing IDENTIFY.

**False** -

- If the dialog had been opened by this property, then the LAN Status dialog is closed.
- If the dialog was opened manually, then it will stay open.

**Return Type** Boolean

**Default** False

**Examples**

```
app.LXIDeviceIDState = True 'Write  
value = app.LXIDeviceIDState 'Read
```

**C++ Syntax** HRESULT get\_LXIDeviceIDState( VARIANT\_BOOL\* pState);  
HRESULT put\_LXIDeviceIDState( VARIANT\_BOOL state);

**Interface** IApplication14

---

Last Modified:

2-Jun-2008    MX New topic

## MagnitudeOffset Property

---

**Description** Offsets the data trace magnitude by the specified value.  
To offset the data trace magnitude to a slope value that changes with frequency, use [MagnitudeSlopeOffset Property](#).  
To implement a Receiver Cal offset, use [LogMagnitudeOffset property](#).

**VB Syntax** *meas.MagnitudeOffset* = *value*

**Variable** [\(Type\)](#) - Description

*meas* **(object)** - A [Measurement](#) object

*value* **(double)** - Offset value in dB.

**Return  
Type** Double

**Default** 0

**Examples** `meas.MagnitudeOffset = 4 'Write`

`offs = meas.MagnitudeOffset 'Read`

**C++ Syntax** HRESULT put\_MagnitudeOffset(double newVal);  
HRESULT get\_MagnitudeOffset(double \*pVal);

**Interface** IMeasurement4

## MagnitudeSlopeOffset Property

---

**Description** Offsets the data trace magnitude to a value that changes linearly with frequency. The offset slope begins at 0 Hz.

To offset the entire data trace magnitude by a specified value, use [MagnitudeOffset Property](#).

**VB Syntax** *meas.MagnitudeSlopeOffset* = *value*

**Variable** [\(Type\)](#) - **Description**

*meas* **(object)** - A [Measurement](#) object

*value* **(double)** - Offset slope value in dB / 1GHz.

**Return  
Type** Double

**Default** 0

**Examples** `meas.MagnitudeSlopeOffset = 4 'Writes a slope offset of 4dB/1GHz.`

`offs = meas.MagnitudeSlopeOffset 'Read`

**C++ Syntax** HRESULT put\_MagnitudeSlopeOffset(double newVal);  
HRESULT get\_MagnitudeSlopeOffset(double \*pVal);

**Interface** IMeasurement4



## MainToneIFBandwidth Property

---

**Description** Sets and returns the IF Bandwidth for measurement of the Main tones.

**VB Syntax** *imd.MainToneIFBandwidth = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Double) IF Bandwidth in Hz. Choose from: 1 | 2 | 3 | 5 | 7 | 10 | 15 | 20 | 30 | 50 | 70 | 100 | 150 | 200 | 300 | 500 | 700 | 1k | 1.5k | 2k | 3k | 5k | 7k | 10k | 15k | 20k | 30k | 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k

[Learn more about setting IFBW for IMD.](#)

If an invalid number is specified, the analyzer will round up to the closest valid number.

**Return Type** Double

**Default** 1 kHz

**Examples** `imd.MainToneIFBandwidth = 2e3 'Write`

`value = imd.MainToneIFBandwidth 'Read`

**C++ Syntax** HRESULT get\_MainToneIFBandwidth(double \*pVal)  
HRESULT put\_MainToneIFBandwidth(double newVal)

**Interface** ISweptIMD

---

Last Modified:

19-Aug-2008    MX New topic

## MarkerAnnotation Property

---

**Description** Returns the Y-axis marker value from the specified tuning sweep. Use [IsMarkerOn](#) to confirm if a marker was used for the tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**MarkerAnnotation** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.MarkerAnnotation 3 'read`

**C++ Syntax** HRESULT MarkerAnnotation(long sweep, BSTR\* annotation);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## MarkerFormat Property

---

**Description** Sets (or returns) the format of all the markers in the measurement. To override this setting for an individual marker, use [mark.Format](#)

**VB Syntax** *meas.MarkerFormat* = *value*

**Variable** [\(Type\)](#) - [Description](#)

*meas* A Measurement (**object**)

*value* (**enum NAMarkerFormat**) - Choose from:

- 0 - naMarkerFormat\_LinMag
- 1 - naMarkerFormat\_LogMag
- 2 - naMarkerFormat\_Phase
- 3 - naMarkerFormat\_Delay
- 4 - naMarkerFormat\_Real
- 5 - naMarkerFormat\_Imaginary
- 6 - naMarkerFormat\_SWR
- 7 - naMarkerFormat\_LinMagPhase
- 8 - naMarkerFormat\_LogMagPhase
- 9 - naMarkerFormat\_ReallImaginary
- 10 - naMarkerFormat\_ComplexImpedance
- 11 - naMarkerFormat\_ComplexAdmittance
- 12 - naMarkerFormat\_Kelvin
- 13 - naMarkerFormat\_Fahrenheit
- 14 - naMarkerFormat\_Centigrade

**Return Type** Long Integer

**Default** 1 - naMarkerFormat\_LogMag

**Examples**

```
meas.MarkerFormat = naMarkerFormat_SWR 'Write
```

```
fmt = mark.Format 'Read
```

**C++ Syntax** HRESULT put\_MarkerFormat(tagNAMarkerFormat NewFormat)

**Interface** IMeasurement

---

Last Modified:

1-Oct-2007    Added temperature formats

## InterpolateMarkers Method

---

**Description** Turns **All** Marker Interpolation ON and OFF for the measurement. Marker interpolation enables X-axis resolution between the discrete data values. The analyzer will calculate the x and y-axis data values between discrete data points. To override this property for individual markers, use the [Interpolated](#) property.

**VB Syntax** *meas.Interpolate = value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**boolean**)

**False** - Turns interpolation OFF for all markers in the measurement

**True** - Turns interpolation ON for all markers in the measurement

**Return Type** Boolean

**Default** True (ON)

**Examples** `meas.Interpolate = 1`

**C++ Syntax** HRESULT InterpolateMarkers(VARIANT\_BOOL bNewVal)

**Interface** IMeasurement

## Number Property

---

**Description** Returns the number of the marker.

**VB Syntax** *marknum* = *mark*.**Number**

**Variable** [\(Type\)](#) - **Description**

*marknum* **(long)** - Variable to store marker number

*mark* A Marker **(object)**

**Return  
Type** Long Integer

**Default** Not applicable

**Examples** `marknum = mark.Number 'Read`

**C++ Syntax** HRESULT get\_Number(long \*pVal)

**Interface** IMarker

## MarkerPosition Property

---

**Description** Returns the X-axis marker position from the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**MarkerPosition** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.MarkerPosition 3 'read`

**C++ Syntax** HRESULT MarkerPosition(long sweep, double \*position);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## MarkerReadout Property

---

**Description** Enables or disables the readout of markers in the window. To show the marker on the screen use ShowMarkerReadout Method.

**VB Syntax** *win*.**MarkerReadout** = *state*

**Variable** [\(Type\)](#) - [Description](#)

*win* A NAWindow (object)

*state* **(boolean)**

**True** - enables marker readout

**False** - disables marker readout

**Return Type** Boolean

**Default** True

**Examples**

```
win.MarkerReadout = True 'Write
```

```
State = app.ActiveNAWindow.MarkerReadout 'Read
```

**C++ Syntax** HRESULT get\_MarkerReadout(VARIANT\_BOOL \*pVal)  
HRESULT put\_MarkerReadout(VARIANT\_BOOL newVal)

**Interface** INAWindow



## MarkerReadoutSize Property

---

**Description** Specifies the size of font used when displaying Marker Readout in the selected window.

**VB Syntax** *win.MarkerReadoutSize = value*

**Variable** [\(Type\)](#) - Description

*win* A NAWindow (object)

*value* (enum NAFontSize)

**0 - naDefault** - marker readout appears in default font size

**1 - naLarge** - marker readout appears in large font size

**Return Type** Long Integer

**Default** naDefault

**Examples** `win.MarkerReadoutSize = naDefault 'write default size for marker readout`

```
Dim Size As NAFontSize
Size = app.ActiveNAWindow.MarkerReadoutSize 'Read
```

**C++ Syntax** HRESULT get\_MarkerReadoutSize(tagNAFontSize \*pVal)  
HRESULT put\_MarkerReadoutSize(tagNAFontSize newVal)

**Interface** INAWindow

## MarkerState Property

---

**Description** Sets or returns the ON / OFF state of the specified marker.

**VB Syntax** *meas.MarkerState (n) = state*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*n* (Long Integer) Marker number to turn on or off.

*state* (boolean) -  
True - turns the specified marker ON  
False - turns the specified marker OFF

**Return Type** Boolean

**Default** False

**Examples** `meas.MarkerState(1) = True`

`reference = meas.MarkerState(2)`

**C++ Syntax** HRESULT get\_MarkerState(long markerNum, VARIANT\_BOOL bState)  
HRESULT put\_MarkerState(long markerNum, VARIANT\_BOOL\* bState)

**Interface** IMeasurement3

## Type (Marker) Property

---

**Description** Sets and reads the marker type.

**VB Syntax** *mark.Type = value*

**Variable** [\(Type\)](#) - Description

*chan* A Marker (**object**)

*value* (**enum NAMarkerType**) - Marker Type. Choose from:  
**0 - naMarkerType\_Normal** - the X-axis value for a normal marker will always be determined by the measurement data of the marker.  
**1 - naMarkerType\_Fixed** - retains and keeps its x-axis value at the time the marker type is set.

**Return  
Type** Long Integer

**Default** naMarkerType\_Normal

**Examples** `mark.Type = naMarkerType_Normal 'Write`

`MrkType = mark.Type 'Read`

**C++ Syntax** HRESULT get\_Type(tagNAMarkerType \*pVal)  
HRESULT put\_Type(tagNAMarkerType newVal)

**Interface** IMarker

## Stimulus Property

---

**Description** Sets and reads the X-Axis value of the marker. If the marker is a delta marker, the value will be relative to the reference marker.

**VB Syntax** *mark*.**Stimulus** = *value*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**double**) - X-Axis value. Choose any number within the full span of the channel or User Range (if set).

**Return Type** Double

**Default** First activated Marker turns ON in the middle of the X-axis range. Subsequent markers turn ON at the position of the most recently active marker.

**Examples** `mark.Stimulus = 3e9 'Write`

`XVal = mark.Stimulus 'Read`

**C++ Syntax** HRESULT get\_Stimulus(double \*pVal)  
HRESULT put\_Stimulus(double newVal)

**Interface** IMarker

## Value Property

---

**Description** Reads the Y-axis value of the marker. If the marker is a delta marker, the value will be relative to the reference marker.

You cannot set the Y-axis value of a marker. The marker remains at the position at the time you set [marker.Type](#).

**Note:** To accurately read the marker Y-axis value with [trace smoothing](#) applied, the requested format must match the [displayed format](#). Otherwise, the returned value is unsmoothed data. For example, to read the smoothed marker value when measuring group delay, both the display format and the marker format must be set to (Group) Delay.

**VB Syntax** *YValue = mark.Value (format)*

**Variable** [\(Type\)](#) - **Description**

*YValue* A variable to store the Y-axis value

*mark* A Marker (**object**)

*format* (**enum NAMarkerFormat**) - The format in which to return the marker's Y-axis value. The number in parenthesis following the format is the number of values that are returned in a variant array. Choose from:

- 0 - naMarkerFormat\_**LinMag** (1)
- 1 - naMarkerFormat\_**LogMag** (1)
- 2 - naMarkerFormat\_**Phase** (1)
- 3 - naMarkerFormat\_**Delay** (1)
- 4 - naMarkerFormat\_**Real** (1)
- 5 - naMarkerFormat\_**Imaginary** (1)
- 6 - naMarkerFormat\_**SWR** (1)
- 7 - naMarkerFormat\_**LinMagPhase** (2)
- 8 - naMarkerFormat\_**LogMagPhase** (2)
- 9 - naMarkerFormat\_**ReallImaginary** (2)
- 10 - naMarkerFormat\_**ComplexImpedance** (3)
- 11 - naMarkerFormat\_**ComplexAdmittance** (3)
- 12 - naMarkerFormat\_**Kelvin** (1)
- 13 - naMarkerFormat\_**Fahrenheit** (1)
- 14 - naMarkerFormat\_**Celsius** (1)

**Return Type** Variant - The previous list of formats indicates the number of values that are returned in a variant array

**Default** Not applicable

**Examples** `YVal = mark.Value 'Read`

**C++ Syntax** HRESULT get\_Value(tagNAMarkerFormat format, VARIANT \*pVal)

**Interface** IMarker

---

Last modified:

1-Oct-2007 Added temperature formats

Dec. 4, 2006 Added smoothing note

## MaximumFrequency Property

---

**Description** Sets and Returns the maximum frequency for the calibration standard.

**VB Syntax** *calstd.MaximumFrequency = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**double**) - Maximum frequency in Hertz.

**Return Type** Double

**Default** Not Applicable

**Examples** `calstd.MaximumFrequency = 9e9 'Write`

`maxFrequency = calstd.MaximumFrequency 'Read`

**C++ Syntax** HRESULT get\_MaximumFrequency(double \*pVal)  
HRESULT put\_MaximumFrequency(double newVal)

**Interface** ICalStandard

## MaximumFrequency Property

---

**Description** Returns the maximum frequency of the remote PNA.

**VB Syntax** `value = cap.MaximumFrequency`

**Variable** [\(Type\)](#) - Description

*value* (Double) - Variable to store the returned maximum frequency of the PNA.

*cap* A [Capabilities](#) (object)

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MaximumFrequency 'Read`

**C++ Syntax** `HRESULT get_MaximumFrequency(&frequencyMax);`

**Interface** ICapabilities



## MaximumFrequency Property

---

**Description** Set and returns the maximum usable frequency specified for the power sensor.

**VB Syntax** *pwrSensor*.**MaximumFrequency** = *value*

**Variable** [\(Type\)](#) - **Description**

*pwrSensor* A [PowerSensor](#) (Object) or  
a [PowerSensorAsReceiver](#) (Object)

*value* **(double)** -Frequency in Hertz.

**Return Type** Double

**Default** Device dependent

**Examples** `pwrSensor.MaximumFrequency = 6e9 'Write`  
`MaxFreq = pwrSensor.MaximumFrequency 'Read`

**C++ Syntax** HRESULT put\_MaximumFrequency(double newVal);  
HRESULT get\_MaximumFrequency(double \*pVal);

**Interface** IPowerSensor  
IPowerSensorPMAR

---

Last Modified:

3-Jun-2010    MX New topic

## MaximumIFFilterSampleCount Property

---

**Description** Returns the maximum allowed value for the [IFFilterSampleCount](#) property for the queried PNA.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *value* = *IfConfig*.MaximumIFFilterSampleCount

**Variable** [\(Type\)](#) - Description

*value* **(long)** The maximum allowed value that can be applied to the [IFFilterSampleCount](#) property.

*IfConfig* [IFConfiguration](#) (object)

**Return Type** Long Integer

**Default** PNA Model number dependent

**Examples**

```
variable =  
App.ActiveChannel.IFConfiguration.MaximumIFFilterSampleCount  
'Read
```

[See an example program](#)

**C++ Syntax** HRESULT get\_MaximumIFFilterSampleCount( long \* pMaxSampleCount );

**Interface** IIFConfiguration2

## MaximumIFFrequency Property

---

**Description** Returns the Maximum allowed value for the [IFFrequency Property](#)

**VB Syntax** *value* = *IfConfig*.**MaximumIFFrequency**

**Variable** [\(Type\)](#) - **Description**

*value* **(double)** Variable to store the returned maximum allowed frequency that can be applied to the [IFFrequency Property](#).

*IfConfig* [IFConfiguration](#) (object)

**Return Type** Double

**Default** Not Applicable

**Examples** `val = App.ActiveChannel.IFConfiguration.MaximumIFFrequency` 'Read

**C++ Syntax** HRESULT get\_**MaximumIFFrequency**( double \* pMaxFreq);

**Interface** IIFConfiguration3

---

Last Modified:

18-Jun-2007    MX New topic

## MaximumIterationsPerPoint Property

---

**Description** This command, along with [IterationsTolerance Property](#) deal with ADJUSTMENTS made to the source power.

Sets the maximum number of readings to take at each data point for iterating the source power. Power readings will continue to be made, and source power adjusted, until a reading is within the [IterationsTolerance](#) value or this max number of readings has been met. The last value to be read is the valid reading for that data point.

The following two commands allow for settling of power READINGS.

[ReadingsPerPoint Property](#)

[ReadingsTolerance Property](#)

**VB Syntax** *pwrCal.MaximumIterationsPerPoint = value*

**Variable** [\(Type\)](#) - Description

*pwrCal* **(object)** - A [SourcePowerCalibrator](#) (object)

*value* **(Long)** – Maximum number of readings. Choose any number between 1 and 100.

**Return Type** Long Integer

**Default** 5

**Examples**

```
Set powerCalibrator = pna.SourcePowerCalibrator
powerCalibrator.MaximumIterationsPerPoint = 5 'Write
MaxReads = powerCalibrator.MaximumIterationsPerPoint 'Read
```

**C++ Syntax** HRESULT get\_MaximumIterationsPerPoint( long \*pVal);  
HRESULT put\_MaximumIterationsPerPoint( long newVal);

**Interface** ISourcePowerCalibrator3

---

Last Modified:

17-Apr-2007    Clarified verbage

## MaximumNumberOfChannels Property

---

**Description** Returns the maximum possible number of channels that can be used in the PNA.

**VB Syntax** *value* = *cap*.**MaximumNumberOfChannels**

**Variable** [\(Type\)](#) - **Description**

*value* **(Long)** - Variable to store the returned maximum value for number of channels.

*cap* A [Capabilities](#) **(object)**

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.MaximumNumberOfChannels 'Read`

**C++ Syntax** HRESULT get\_MaximumNumberOfChannels(long \*  
maximumNumberOfChans );

**Interface** ICapabilities2

## MaximumNumberOfTracesPerWindow Property

---

**Description** Returns the maximum possible number of traces that can reside in any window.

**VB Syntax** *value* = *cap*.MaximumNumberOfTracesPerWindow

**Variable** [\(Type\)](#) - Description

*value* **(Long)** - Variable to store the returned maximum value for number of traces.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.MaximumNumberOfTracesPerWindow 'Read`

**C++ Syntax** HRESULT get\_MaximumNumberOfTracesPerWindow(long \*  
maximumNumberOfTraces );

**Interface** ICapabilities2

## MaximumNumberOfWindows Property

---

**Description** Returns the maximum possible number of windows that can be present on the PNA screen.

**VB Syntax** *value* = *cap*.MaximumNumberOfWindows

**Variable** [\(Type\)](#) - Description

*value* **(Long)** - Variable to store the returned maximum value for number of windows.

*cap* A [Capabilities](#) **(object)**

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.MaximumNumberOfWindows 'Read`

**[C++ Syntax](#)** HRESULT get\_MaximumNumberOfWindows(long \* maximumNumberOfWindows);

**Interface** ICapabilities2

## MaximumNumberOfPoints Property

---

**Description** Returns the maximum possible number of data points.

**VB Syntax** *value* = *obj*.MaximumNumberOfPoints

**Variable** [\(Type\)](#) - Description

*value* **(Long)** - Variable to store the returned maximum value for number of points.

*cap* A [Capabilities](#) (object)

or

A [GainCompression](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.MaximumNumberOfPoints 'Read`

**C++ Syntax** HRESULT get\_MaximumNumberOfPoints(long \*  
maximumNumberOfPoints);

**Interface** ICapabilities  
IGainCompression

---

Last Modified:

3-Dec-2007    MX New topic



## MaximumReceiverStepAttenuator Property

---

**Description** Returns the maximum amount of receiver attenuation.

**VB Syntax** *value* = *cap*.MaximumReceiverStepAttenuator (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(Double)** - Variable to store the returned value of maximum receiver attenuation.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - port number to query for step attenuators

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MaximumReceiverStepAttenuator 'Read`

**C++ Syntax** HRESULT get\_MaximumReceiverStepAttenuator(long portNumber, double \*attenuation );

**Interface** ICapabilities

## MaximumSourceALCPower Property

---

**Description** Returns a value indicating the maximum amount of source ALC power.

**VB Syntax** *value* = *cap*.MaximumSourceALCPower (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(Double)** - Variable to store the returned value for the maximum amount of source ALC power.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - source number to query for maximum ALC power

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MaximumSourceALCPower 'Read`

**C++ Syntax** HRESULT get\_MaximumSourceALCPower(long sourceNum, double \* power );

**Interface** ICapabilities

## MaximumSourceStepAttenuator Property

---

<b>Description</b>	Returns a value for the maximum amount of source attenuation.
<b>VB Syntax</b>	<i>value</i> = <i>cap</i> .MaximumSourceStepAttenuator ( <i>n</i> )
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>value</i>	<b>(Double)</b> - Variable to store the returned value for the maximum amount of source attenuation.
<i>cap</i>	A <a href="#">Capabilities</a> (object)
<i>n</i>	<b>(Long)</b> - port number to query for the maximum amount of source attenuation
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>value = cap.MaximumSourceStepAttenuator 2 'Read</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_MaximumSourceStepAttenuator(long portNumber, double * attenuation );
<b>Interface</b>	ICapabilities

---

Last Modified:

19-Sep-2007    Added port arg to ex.

## MaxPreciseTuningIterations Property

---

**Description** Sets and returns the maximum number of tuning iterations to achieve the precise tolerance.

**VB Syntax** *obj.MaxPreciseTuningIterations = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* (**Long**) Maximum number of tuning iterations.

**Return Type** (**Long**)

**Default** 5

**Examples** `embedLO.MaxPreciseTuningIterations = 3 'write`

`value = embedLO.MaxPreciseTuningIterations 'read`

**C++ Syntax** HRESULT get\_MaxPreciseTuningIterations long\* iter);  
HRESULT put\_MaxPreciseTuningIterations long iter);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## MaxProduct Property

---

**Description** Sets and returns the maximum intermod product frequencies to be calibrated.

**VB Syntax** *imd.MaxProduct = value*

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMDCal](#) (object)

*value* Maximum IM products to calibrate. Choose from:

2 - second order products

3 - third order products

5 - fifth order products

7 - seventh order products

9 - ninth order products

**Return Type** Long Integer

**Default** 3

**Examples** `imd.MaxProduct = 7 'Write`

`mprod = imd.MaxProduct 'Read`

**C++ Syntax** HRESULT get\_MaxProduct(long \* Val)  
HRESULT put\_MaxProduct(long newVal)

**Interface** ISweptIMD

---

Last Modified:

9-Sep-2008    MX New topic

## Mean Property

---

**Description** Returns the mean value of the measurement . To retrieve all 3 statistics value at the same time, use [meas.GetTraceStatistics](#)

**VB Syntax** *average* = *meas*.**Mean**

**Variable** [\(Type\)](#) - **Description**

*average* **(single)** - Variable to store mean value

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples**

```
Dim average as Single
average = meas.Mean 'Read
```

**C++ Syntax** HRESULT get\_Mean(float\* mean)

**Interface** IMeasurement

## MeasurementClass Property

---

**Description** Returns the measurement class name from the channel. Use [CreateCustomMeasurementEx](#) to create a measurement from a class other than standard S-Parameters.

**VB Syntax** `class = chan.MeasurementClass`

**Variable** [\(Type\)](#) - **Description**

*class* **(string)** - Variable to store the returned measurement class name.

*chan* [Channel](#) **(object)**

**Return Type** String

**Default** Not Applicable

**Examples**

```
class = chan.MeasurementClass 'Read  
For a standard S-Parameter channel, returns...  
"Standard"
```

**C++ Syntax** HRESULT get\_MeasurementClass();

**Interface** IChannel15

---

### Last Modified:

6-Oct-2008    Corrected IChannel

12-May-2008    MX New topic

## Medium Property

---

**Description** Sets and Returns the media type of the calibration standard.

**VB Syntax** *calstd.Medium = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**enum** NACalStandardMedium) - Medium of the transmission line of the standard. Choose from:  
**0 - naCoax** - Coaxial Cable  
**1 - naWaveGuide**

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `calstd.Medium = naCoax 'Write`

```
stdMedium = calstd.Medium 'Read
```

**C++ Syntax** HRESULT get\_Medium(tagNACalStandardMedium \*pVal)  
HRESULT put\_Medium(tagNACalStandardMedium newVal)

**Interface** ICalStandard



## MinimumFrequency Property

---

**Description** Sets and Returns the minimum frequency for the calibration standard.

**VB Syntax** *calstd.MinimumFrequency = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**double**) -Minimum frequency in Hertz.

**Return Type** Double

**Default** Not Applicable

**Examples** `calstd.MinimumFrequency = 300e3` 'Write

`minFrequency = calstd.MinimumFrequency` 'Read

**C++ Syntax** HRESULT get\_MinimumFrequency(double \*pVal)  
HRESULT put\_MinimumFrequency(double newVal)

**Interface** ICalStandard

## MinimumFrequency Property

---

**Description** Returns the minimum frequency of the remote PNA.

**VB Syntax** `value = cap.MinimumFrequency`

**Variable** [\(Type\)](#) - **Description**

*value* (Double) - Variable to store the returned minimum frequency of the PNA.

*cap* A [Capabilities](#) (**object**)

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MinimumFrequency 'Read`

**C++ Syntax** HRESULT get\_MinimumFrequency(double \*pVal)

**Interface** ICapabilities

## MinimumFrequency Property

---

**Description** Sets and returns the minimum usable frequency specified for the power sensor.

**VB Syntax** *pwrSensor.MinimumFrequency = value*

**Variable** [\(Type\)](#) - Description

*pwrSensor* A [PowerSensor](#) (Object) or  
A [PowerSensorAsReceiver](#) (Object)

*value* **(double)** -Frequency in Hertz.

**Return Type** Double

**Default** Device dependent

**Examples** `pwrSensor.MinimumFrequency = 300e3 'Write`  
`MinFreq = pwrSensor).MinimumFrequency 'Read`

**C++ Syntax** HRESULT put\_MinimumFrequency(double newVal);  
HRESULT get\_MinimumFrequency(double \*pVal);

**Interface** IPowerSensor  
IPowerSensorPMAR

---

Last Modified:

25-Aug-2009    Added PMAR

## MinimumIFFilterSampleCount Property

---

**Description** Returns the Minimum allowed value for the [IFFilterSampleCount](#) property for the queried PNA.

**Note:** This setting applies only to the E836X Opt. H11.

**VB Syntax** *value* = *IfConfig*.MinimumIFFilterSampleCount

**Variable** [\(Type\)](#) - Description

*value* **(long)** The minimum allowed value that can be applied to the [IFFilterSampleCount](#) property.

*IfConfig* [IFConfiguration](#) (object)

**Return Type** Long Integer

**Default** PNA Model number dependent

**Examples**

```
variable =  
App.ActiveChannel.IFConfiguration.MinimumIFFilterSampleCount  
'Read
```

[See an example program](#)

**C++ Syntax** HRESULT get\_MinimumIFFilterSampleCount( long \* pMinSampleCount );

**Interface** IIFConfiguration2

## MinimumIFFrequency Property

---

**Description** Returns the minimum allowed value for the [IFFrequency Property](#)

**VB Syntax** *value* = *IfConfig*.**MinimumIFFrequency**

**Variable** [\(Type\)](#) - **Description**

*value* **(double)** Variable to store the returned minimum allowed frequency that can be applied to the [IFFrequency Property](#).

*IfConfig* [IFConfiguration](#) (object)

**Return Type** Double

**Default** Not Applicable

**Examples** `val = App.ActiveChannel.IFConfiguration.MinimumIFFrequency` 'Read

**C++ Syntax** HRESULT get\_MinimumIFFrequency( double \* pMinFreq);

**Interface** IIFConfiguration3

---

Last Modified:

18-Jun-2007    MX New topic

## MinimumNumberOfPoints Property

---

**Description** Returns the minimum possible number of data points for a data trace.

**VB Syntax** *value* = *cap*.MinimumNumberOfPoints

**Variable** [\(Type\)](#) - Description

*value* (Long) - Variable to store the returned minimum value for number of points.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.MinimumNumberOfPoints 'Read`

**C++ Syntax** HRESULT get\_MinimumNumberOfPoints(double \*  
minimumNumberOfPoints );

**Interface** ICapabilities

## MinimumReceiverStepAttenuator Property

---

**Description** Returns a value indicating the minimum amount of receiver attenuation.

**VB Syntax** *value* = *cap*.MinimumReceiverStepAttenuator (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(Double)** - Variable to store the returned minimum value of receiver attenuation.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - port number to query for minimum value of receiver attenuation

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MinimumReceiverStepAttenuator 'Read`

**C++ Syntax** HRESULT get\_MinimumReceiverStepAttenuator(long portNumber, double \* attenuation );

**Interface** ICapabilities

## MinimumSourceALCPower Property

---

**Description** Returns a value indicating the minimum amount of source ALC power.

**VB Syntax** *value* = *cap*.MinimumSourceALCPower (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(Double)** - Variable to store the returned minimum value of source ALC power.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - source number to query for the minimum value of source ALC power

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.MinimumSourceALCPower 'Read`

**C++ Syntax** HRESULT get\_MinimumSourceALCPower(long sourceNum, double \* power );

**Interface** ICapabilities



## Mode Property

---

**Description** Sets the type of transform.

**VB Syntax** *trans.Mode* = *value*

**Variable** [\(Type\)](#) - **Description**

*trans* A Transform (**object**)

*value* (**enum NATransformMode**) - Choose from:

**0** - naTransformBandpassImpulse

**1** - naTransformLowpassImpulse

**2** - naTransformLowpassStep

**Return Type** NATransformMode

**Default** **0** - naTransformBandpassImpulse

**Examples** `trans.Mode = naTransformLowpassStep 'Write`

`transmode = trans.Mode 'Read`

**C++ Syntax** HRESULT get\_Mode(tagNATransformMode \*pVal)  
HRESULT put\_Mode(tagNATransformMode newVal)

**Interface** ITransform

## Mode Property

---

**Description** Sets and returns the stimulus mode for balanced measurements.

**VB Syntax** *balStim.Mode = value*

**Variable** [\(Type\)](#) - Description

*balStim* A [BalancedStimulus](#) (object)

*value* (**Enum NABALSTIMulus**) - Stimulus Mode. True modes are applicable only with [Opt 460 - iTMSA](#). When a True-Mode is selected, the Balanced port powers are automatically uncoupled.

Choose from:

0 - **naSEStim**: Single-Ended stimulus

1 - **naTMStim**: True-Mode stimulus

2 - **naFTMStim**: Forward only True-Mode stimulus

3 - **naRTMStim**: Reversed only True-Mode stimulus

**Return Type** Enum

**Default** 0 - **naSEStim**: Single-Ended stimulus

**Examples**

```
balStim.Mode = naTMStim 'Write
```

```
variable = balStim.Mode 'Read
```

[See an example iTMSA program](#)

**C++ Syntax** HRESULT get\_Mode(tagNABALSTIMulus \*eVal );  
HRESULT put\_Mode(tagNABALSTIMulus eVal );

**Interface** IBalancedStimulus

---

Last Modified:

15-May-2008    MX New topic

## Multiplier Property

---

**Description** Sets and returns the multiplier value to be used when coupling this range to the primary range. Learn more about multiplier value.

This setting is valid only if this range is [coupled](#) to the primary range.

**VB Syntax** *FOMRange.Multiplier = value*

**Variable** [\(Type\)](#) - Description

*object* An [FOMRange](#) (**object**)

*value* (**Double**) - Multiplier value.-(Unitless)

**Return Type** Double

**Default** 1

**Examples** `fomRange.Multiplier = 2 'Write`

`Mult = fomRange.Multiplier 'Read`

**C++ Syntax** HRESULT get\_Multiplier(double \*pVal)  
HRESULT put\_Multiplier(double pVal)

**Interface** IFOMRange

## Name Property

---

**Description** Sets or returns the Name of the Cal Set.

**VB Syntax** *CalSet*.**Name** = *value*

**Variable** [\(Type\)](#) - **Description**

*CalSet* **(object)** - A [Cal Set](#) object

*value* **(string)** - Name of the Cal Set.

**Return Type** String

**Default** Not Applicable

**Examples**

```
Dim pna
set
pna=CreateObject("AgilentPNA835x.Application")

Dim calsets
set calsets=pna.getcalmanager.calsets

Dim c
for each c in calsets
wscript.echo c.name
'Changes the name of CalSet_1
if c.name="CalSet_1" then c.name="New"
next
```

**[C++ Syntax](#)** HRESULT get\_Name(BSTR \*name)  
HRESULT put\_Name(BSTR name);

**Interface** ICalSet4

## Name (CalKit) Property

---

**Description** Sets and Returns a name for the selected calibration kit.

**VB Syntax** *calKit*.Name = *value*

**Variable** [\(Type\)](#) - **Description**

*calKit* A CalKit (**object**).

*value* (**string**) -Calibration Kit name. Any string name, can include numerics, period, and spaces; any length (although the dialog box display is limited to about 30 characters).

**Return  
Type** String

**Default** Not Applicable

**Examples** `calKit.Name = "MyCalKit" 'Write`

`KitName = calKit.Name 'Read`

**[C++ Syntax](#)** HRESULT get\_Name(BSTR \*pVal)  
HRESULT put\_Name(BSTR newVal)

**Interface** ICalKit

## Name (PathConfig) Property

---

**Description** Returns the name of the current configuration only if NO individual element settings had been changed since selecting or storing a configuration. When element settings change, the path configuration name is cleared.

**VB Syntax** *name* = *pathConfig*.**Name**

**Variable** [\(Type\)](#) - **Description**

*name* **(String)** Variable to store the returned configuration name.

*pathConfig* A [PathConfiguration](#) **(object)**

**Return Type** String

**Default** 'default' - name of the default factory configuration

**Examples** `name=pathConf.Name`

**C++ Syntax** HRESULT get\_Name(BSTR\* ppName)

**Interface** IPathConfiguration

---

Last Modified:

14-Dec-2006    MX New topic

## Name Property

---

**Description** Returns the name of the current element object

**VB Syntax** *name* = *pathElement*.**Name**

**Variable** [\(Type\)](#) - Description

*name* **(String)** Variable to store the returned element name.

*pathElement* A [PathElement](#) **(object)**

**Return Type** String

**Default** Not Applicable

**Examples** `name=pathElement.Name`

**C++ Syntax** HRESULT get\_Name(BSTR\* ppName)

**Interface** IPathElement

---

Last Modified:

15-Dec-2006    MX New topic

## Name Property

---

**Description** Returns the name of this FOM range object.

**VB Syntax** *value* = *FOMRange*.**Name**

**Variable** [\(Type\)](#) - **Description**

*value* **(string)** - Variable to store the returned range name.

*FOMRange* An [FOMRange](#) **(object)**

**Return Type** String

**Default** Not Applicable

**Examples** `Rname = fomRange.Name 'Read`

**C++ Syntax** HRESULT get\_Name(BSTR \*pRName)

**Interface** IFOMRange

---

Last Modified:

8-Mar-2007    Major Modifications



## Name (Measurement) Property

---

**Description** Sets (or returns) the Name of the measurement. Measurement names must be unique among the set of measurements. Measurement names cannot be an empty string.

**Note:** This is the same name as trace.Name; when one changes, the other changes.

**VB Syntax** *meas.Name* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**string**) - A user defined name of the measurement

**Return  
Type** String

**Default** "CH1\_S11\_1" - name of the default measurement

**Examples** `meas.Name = "Filter BPass" 'Write`

`MName = meas.Name 'Read`

**C++ Syntax** HRESULT get\_Name(BSTR \*pVal)  
HRESULT put\_Name(BSTR newVal)

**Interface** IMeasurement

## Name (trace) Property

---

**Description** Sets or returns the name of the Trace. Use the trace name to identify the trace and refer to the trace in the collection.

**Note:** This is the same name as meas.Name; when one changes, the other changes.

**VB Syntax** *trac.Name* = *value*

**Variable** [\(Type\)](#) - Description

*trac* A Trace (**object**)

*value* (**String**) Trace name

**Return  
Type** String

**Default** "CH1\_S11\_1" - name of the default measurement

**Examples** `trace.Name = "myTrace" 'Write`

`traceName = Name.Trace 'Read`

**C++ Syntax** HRESULT put\_Name(BSTR name)  
HRESULT get\_Name(BSTR \*name)

**Interface** ITrace

## NetworkFilename Property

---

**Description** Specifies the S2P filename to embed or de-embed on the input or output of your mixer measurement.

**VB Syntax** *object.NetworkFilename(n) = filename*

**Variable** [\(Type\) - Description](#)

*object* [SMCType](#) (object) or  
[VMCType](#) (object)

*n* (Integer) Apply network to input or output of mixer. Choose from:

**1** - Input of mixer

**2** - Output of mixer

*filename* (String) Filename of the S2P used for embedding or de-embedding. Use the full path name, file name, and .S2P suffix, enclosed in quotes.

**Return Type** String

**Default** Not Applicable

**Examples** `VMC.Filename(2) = "C:/Program Files/Agilent/Network Analyzer/Documents/WaveguideAdapt.S2P"`

**C++ Syntax** HRESULT put\_NetworkFilename(short networkNum, BSTR filename);  
HRESULT get\_NetworkFilename(short networkNum, BSTR \*filename);

**Interface** SMCType2  
VMCType2

---

Last Modified:

29-Feb-2008    Several edits

## NetworkMode Property

---

**Description** Allows you to embed (add) or de-embed (remove) circuit network effects on the input and output of your mixer measurement. [Learn more.](#)

**VB Syntax** *object.NetworkMode(n) = value*

**Variable** [\(Type\) - Description](#)

*object* [SMCType](#) (object) or  
[VMCType](#) (object)

*n* (Integer) Apply network to input or output of mixer. Choose from:

**1** - Input of mixer

**2** - Output of mixer

*value* (Enum as ENetworkMode) Choose from:

**NO\_NETWORK** Do nothing with effects of S2P file

**EMBED\_NETWORK** - Add effects of S2P file from the measurement results.

**DEEMBED\_NETWORK** - Remove effects of S2P file from the measurement results.

**Return Type** Enum

**Default** NO\_NETWORK

**Examples** `VMC.NetworkMode = EMBED_NETWORK`

**[C++ Syntax](#)** HRESULT put\_NetworkMode(short networkNum, enum ENetworkMode networkMode);  
HRESULT get\_NetworkMode(short networkNum, enum ENetworkMode \*networkMode);

**Interface** SMCType2  
VMCType2

## NoiseAverageFactor Property

---

**Description** Sets and reads the averaging of the noise receiver.

**VB Syntax** *noise.NoiseAverageFactor* = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* (**long integer**) - Averaging value. Choose a number between 1 and 99.

**Return Type** Long Integer

**Default** 1

**Examples** `noise.NoiseAverageFactor = 10 'Write`

`AvgNoise = noise.NoiseAverageFactor 'Read`

**[C++ Syntax](#)** HRESULT get\_NoiseAverageFactor(long\* pVal)  
HRESULT put\_NoiseAverageFactor(long newVal)

**Interface** INoiseFigure

---

Last Modified:

29-May-2007 MN New topic

## NoiseAverageState Property

---

**Description** Turns Noise Averaging ON and OFF.

**VB Syntax** *noise.NoiseAverageState* = *value*

**Variable**    [\(Type\)](#) - Description

*noise*    A [NoiseFigure](#) (object)

*value*    (boolean) - Averaging state.

**False** - Turns Noise Averaging OFF

**True** - Turns Noise Averaging ON

**Return Type** Boolean

**Default** False

**Examples**    `noise.NoiseAverageState = OFF 'Write`  
`NoiseAvgState = noise.NoiseAverageState 'Read`

**[C++ Syntax](#)**    HRESULT get\_NoiseAverageState(VARIANT\_BOOL\* on);  
                  HRESULT put\_NoiseAverageState(VARIANT\_BOOL on);

**Interface**    INoiseFigure

---

Last Modified:

21-Jun-2007    MX New topic

## NoiseBandwidth Property

---

**Description** Set the bandwidth of the noise receiver.

**VB Syntax** *noise.NoiseBandwidth* = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(double)** Bandwidth value.

For [NoiseReceiver](#) = **naNoiseReceiver** (Opt 029) choose from: 800 KHz, 2 MHz, 4 MHz, 8 MHz, or 24 MHz or the numerical equivalent, such as 8e6 and so forth.

For NoiseReceiver = **naStandardReceiver** (Opt 028) choose from: 720 kHz or 1.2 MHz

If the value does not match one of these, it is rounded up to the next valid bandwidth value.

**Return  
Type** Double

**Default** 4 MHz for **naNoiseReceiver**  
1.2 MHz for **naStandardReceiver**

**Examples** `noise.NoiseBandwidth = 2E6 'Write`  
`NoiseBW = noise.NoiseBandwidth 'Read`

**C++ Syntax** HRESULT get\_NoiseBandwidth(double \*pVal);  
HRESULT put\_NoiseBandwidth(double newVal);

**Interface** INoiseFigure

---

Last Modified:

21-Apr-2010 Added noise receiver options

20-Sep-2007 MX New topic

## NoiseGain Property

---

**Description** Sets and reads the gain state of the noise receiver. This setting is NOT used when [NoiseReceiver](#) = naStandardReceiver (Opt 028)

**VB Syntax** *noise.NoiseGain* = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(long integer)** - Gain value. Choose from:

**0** Low Gain select if the gain of your DUT is relatively high (>35 dB).

**15** Medium Gain select if the gain of your DUT is about average (20 dB to 45 dB).

**30** High Gain ..select if the gain of your DUT is relatively low (<30 dB).

If the value does not match one of these, it is rounded up to the next legal value.

[Learn more about Noise Receiver Gain setting.](#)

**Return Type** Long Integer

**Default** 30

**Examples** `noise.NoiseGain = 30 'Write`

`GainNoise = noise.NoiseGain 'Read`

**C++ Syntax** HRESULT get\_NoiseGain(long\* pVal)  
HRESULT put\_NoiseGain(long newVal)

**Interface** INoiseFigure

---

Last Modified:

29-May-2007 MN New topic



## NoiseSourceCalKitType Property

---

**Description** Set and read the Cal Kit that will be used for the Noise Source adapter.  
An adapter is always necessary to connect a 346C Noise Source to the PNA port 2. Select a Cal Kit that is the same type and gender as the noise source connector.  
If the Noise Source mates directly to PNA port 2, then set this type to "None".

**VB Syntax** *noise.NoiseSourceCalKitType = value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseCal](#) (object)

*value* (**string**) Cal Kit type. To read possible cal kit strings for the adapter:

1. Change the port connector type to that of the noise source using: [ConnectorType](#)
2. Then read the possible cal kit strings for that port using: [CompatibleCalKits](#)

**Return Type** String

**Default** Not applicable

**Examples** `noise.NoiseSourceCalKitType = "N4691-60004 ECAL" 'Write`

`calkit = noise.NoiseSourceCalKitType 'Read`

**C++ Syntax** HRESULT get\_NoiseSourceCalKitType(BSTR\* pValue)  
HRESULT put\_NoiseSourceCalKitType(BSTR pNewValue)

**Interface** INoiseCal

---

Last Modified:

29-May-2007 MN New topic

## NoiseSourceCold Property

---

**Description** Sets and returns the temperature of the noise source connector.

**VB Syntax** *noise.NoiseSourceCold = value*

**Variable**    [\(Type\)](#) - Description

*noise*    A [NoiseCal](#) (object)

*value*    **(double)** Noise source temperature in Kelvin.

**Return Type** Double

**Default** Not Applicable

**Examples** `noise.NoiseSourceCold = 295 'Write`

`temp = noise.NoiseSourceCold 'Read`

**[C++ Syntax](#)** HRESULT get\_NoiseSourceCold(double\* pTemp)  
HRESULT put\_NoiseSourceCold(double pNewTemp)

**Interface** INoiseCal

---

Last Modified:

21-Jun-2007    MX New topic

## NoiseSourceConnectorType Property

---

**Description** Set and read the Noise Source connector type and gender.

The Agilent 346C has an "APC 3.5 male" connector.

**VB Syntax** *noise.NoiseSourceConnectorType* = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseCal](#) (object)

*value* **(string)** Connector type. Use [ValidConnectorType](#) to return a list of valid connector types.

**Return Type** String

**Default** Not applicable

**Examples** `noise.NoiseSourceConnectorType = "APC 3.5 male" 'Write`

`connector = noise.NoiseSourceConnectorType 'Read`

**C++ Syntax** HRESULT get\_NoiseSourceConnectorType(BSTR\* pConnectorType)  
HRESULT put\_NoiseSourceConnectorType(BSTR pConnectorType)

**Interface** INoiseCal

---

Last Modified:

29-May-2007    MN New topic

## NoiseSourceState Property

---

**Description** Sets and Reads the state of the noise source (ON and OFF).

**VB Syntax** `app.NoiseSourceState = state`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (boolean)

**False (0)** - Turns Noise Source OFF

**True (1)** - Turns Noise Source ON

**Return Type** Boolean

**Default** False

**Examples** `app.NoiseSourceState = True 'Write`

`coupl = app.NoiseSourceState 'Read`

**C++ Syntax** HRESULT put\_NoiseSourceState(VARIANT\_BOOL bState)  
HRESULT get\_NoiseSourceState(VARIANT\_BOOL \*bState)

**Interface** IApplication13

---

Last Modified:

29-May-2007    MN New topic

## NoiseTuner Property

---

**Description** Sets and returns the noise tuner identifier, which is an ECal model and serial number string.

**VB Syntax** *noise.NoiseTuner* = *value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(string)** Noise Tuner. Return the connected ECal identifiers by sending GetCalKitTypeString and passing the module number.

**Return  
Type** String

**Default** Not Applicable

**Examples** `noise.NoiseTuner = "N4691-60004 ECal 02822" 'Write`

`noiseT = noise.NoiseTuner 'Read`

**C++ Syntax** HRESULT get\_NoiseTuner(BSTR\* pValue)  
HRESULT put\_NoiseTuner(BSTR pNewValue)

**Interface** INoiseFigure

---

Last Modified:

18-Jun-2007    MN New topic

## NoiseTunerIn Property

---

**Description** Sets and returns the port identifier of the ECal noise tuner that is connected to the PNA Source.

**VB Syntax** *noise.NoiseTunerIn = value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(string)** Noise Tuner port identifier that is connected to the PNA source, as it is labeled on the ECal module. For example, for 2-port ECal modules, choose either "A" or "B".

**Return  
Type** String

**Default** Not Applicable

**Examples** `noise.NoiseTunerIn = "A" 'Write`

`EcalPort = noise.NoiseTunerIn 'Read`

**C++ Syntax** HRESULT get\_NoiseTunerIn(BSTR\* pValue)  
HRESULT put\_NoiseTunerIn(BSTR pNewValue)

**Interface** INoiseFigure

---

Last Modified:

29-May-2007 MN New topic

## NoiseTunerOut Property

---

**Description** Sets and returns the port identifier of the ECal noise tuner that is connected to the DUT.

**VB Syntax** *noise.NoiseTunerOut = value*

**Variable** [\(Type\)](#) - Description

*noise* A [NoiseFigure](#) (object)

*value* **(string)** Noise Tuner port identifier that is connected to the DUT, as it is labeled on the ECal module. For example, for 2-port ECal modules, choose either "A" or "B".

**Return Type** String

**Default** Not Applicable

**Examples** `noise.NoiseTunerOut = "A" 'Write`

`EcalPort = noise.NoiseTunerOut 'Read`

**C++ Syntax** HRESULT get\_NoiseTunerOut(BSTR\* pValue)  
HRESULT put\_NoiseTunerOut(BSTR pNewValue)

**Interface** INoiseFigure

---

Last Modified:

29-May-2007 MN New topic

## NominalIncidentPowerState Property

---

**Description** Toggles the Nominal Incident Power setting ON and OFF. This setting is ONLY to be used with SMC measurements, not VMC. [Learn more about Nominal Incident Power.](#)

**VB Syntax** *mixer.NominalIncidentPowerState* = *bool*

**Variable** [\(Type\)](#) - [Description](#)

*mixer* A Mixer (**object**)

*bool* (**boolean**) - Nominal Incident Power State. Choose from:

1 -(True) Turn nominal incident power ON

0 -(False) Turn nominal incident power OFF

**Return Type** Boolean

**Default** 0 -(False)

**Examples** `mixer.NominalIncidentPowerState = True 'sets  
NominalIncidentPowerState to ON`

**C++ Syntax** HRESULT get\_NominalIncidentPowerState(VARIANT\_BOOL \*pVal)  
HRESULT put\_NominalIncidentPowerState(VARIANT\_BOOL val );

**Interface** IMixer4



## NormalizePoint Property

---

**Description** Sets and returns the sweep data point around which to perform broadband and precise tuning.

**VB Syntax** *obj.NormalizePoint = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* (**Long**) Mixer Sweep data point. Choose a data point number between 1 and the max number of data points in the sweep that has the least amount of expected noise.

**Return Type** (**Long**)

**Default** Center point in the sweep span

**Examples** `embedLO.NormalizePoint = 101 'write`

`value = embedLO.NormalizePoint 'read`

**C++ Syntax** HRESULT get\_NormalizePoint(long \*point);  
HRESULT put\_NormalizePoint(long point);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## Number (Measurement) Property

---

**Description** Returns the Number of the measurement. Measurement numbers are assigned internally.

**Note:** Measurement numbers are NOT the same as their number in the Measurements collection. Measurement number is used to identify the measurement associated with an event.

This property is used to identify measurements when events occur through the [OnMeasurementEvent](#) callback. For example:

```
OnMeasurementEvent (naEventId_MSG_LIMIT_FAILED, 3)
```

**VB Syntax** *measNum* = *meas*.**Number**

**Variable** [\(Type\)](#) - **Description**

*measNum* **(long)** - variable to store the measurement number

*meas* A Measurement **(object)**

**Return Type** Long Integer

**Default** "1" - number of the default measurement

**Examples** `measNum = meas.Number`

**C++ Syntax** HRESULT get\_Number(long \*MeasurementNumber)

**Interface** IMeasurement

## Number Property

---

<b>Description</b>	Returns the number of the Auxiliary Trigger connector pair currently being used with the instance of the <a href="#">AuxTrigger</a> object. Set the trigger pair with the AuxTrig object.
<b>VB Syntax</b>	<i>value</i> = <i>auxTrig</i> . <b>Number</b>
<b>Variable</b>	<b>(Type) - Description</b>
<i>auxTrig</i>	An <a href="#">AuxTrigger</a> <b>(object)</b>
<i>value</i>	<b>(Long Integer)</b> - Connector pair. PNA-X returns 1 or 2. All other models that do not have the Aux trigger connector returns 1.
<b>Return Type</b>	Single
<b>Default</b>	Not Applicable
<b>Exaamples</b>	<code>value = auxTrig.Number 'Read the value</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_Number(long *auxID);
<b>Interface</b>	IAuxTrigger

---

### Last Modified:

7-Apr-2009	Updated to "all other models.."
14-Dec-2006	MX New topic

## NumberOfFrequencyPoints Property

---

**Description** Set and read the number of frequency points for a Gain Compression channel. Applies to all acquisition modes.

**VB Syntax** `gca.NumberOfFrequencyPoints = value`

**Variable** [\(Type\)](#) - [Description](#)

*gca* A [GainCompression](#) (object)

*value* (**integer**) - Frequency points. Do not exceed the max number of points. [Learn more.](#)

**Return Type** Integer

**Default** 201

**Examples** `gca.NumberOfFrequencyPoints = 101` **Write**

`freqPts = gca.NumberOfFrequencyPoints` **Read**

**C++ Syntax** HRESULT get\_NumberOfFrequencyPoints(int\* pVal)  
HRESULT put\_NumberOfFrequencyPoints(int newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## NumberOfPoints Property

---

**Description** Sets or returns the Number of Points of the channel.  
Sets or returns the Number of Points of the segment.

### See Also

- [Measurement2 Interface](#) to learn how this method differs from [meas.NumberofPoints](#)
- [Gain Compression Number of Points](#).
- [Swept IMD limitations](#)

**VB Syntax** *object*.NumberOfPoints = *value*

### Variable (Type) - Description

*object* Channel (**object**)  
**or**  
CalSet (**object**) - Read-only property

*value* (**long**) - Number of Points.  
For channel, choose any number from **1** to the [PNA max number of points](#).  
For segment, the total number of points in all segments cannot exceed the PNA maximum. A segment can have as few as 1 point.

**Return Type** Long Integer

**Default** 201 for channel  
21 for segment

**Examples** `chan.NumberOfPoints = 201 'sets the number of points for all measurements in the channel. -Write`

`numofpts = chan.NumberOfPoints 'Read`

**C++ Syntax** HRESULT get\_NumberOfPoints(long \*pVal)  
HRESULT put\_NumberOfPoints(long newVal)

**Interface** IChannel  
ISegment  
ICalSet3

---

Last Modified:

5-Aug-2008    Fixed bad link

21-Jun-2007    Increased max

Read-only

## NumberOfPoints Property

---

**Description** Returns the number of data points of the measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**NumberOfPoints**

**Variable** [\(Type\)](#) - **Description**

*value* **(Long)** - variable to store the returned value

*meas* A Measurement **(object)**

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `Print meas.NumberOfPoints` 'prints the number of data points

**C++ Syntax** HRESULT get\_NumberOfPoints(long \*pVal);

**Interface** IMeasurement2

Read-only

## NumberOfPorts Property

---

**Description** Returns the number of hardware source ports on the PNA.

**VB Syntax** *value* = *app*.NumberOfPorts

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (**object**)

*value* (**long integer**) - variable to contain the returned value

**Return Type** (long integer)

**Default** Not Applicable

**Examples** `iNumPorts = app.NumberOfPorts`

**C++ Syntax** HRESULT NumberOfPorts( long\* NumPorts)

**Interface** IApplication



## NumberOfPorts Property

---

**Description** Returns the number of ports on the specified testset. Returns 0 if no test set is connected.

**VB Syntax** *value* = *tset*.**NumberOfPorts**

**Variable** [\(Type\)](#) - **Description**

*value* (Long) variable to store the returned information.

*tset* A [TestsetControl](#) object.

OR

An [E5091Testset](#) object.

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `value = testset1.NumberOfPorts`

[See E5091A Example Program](#)

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_NumberOfPorts(long \*numberOfPorts);

**Interface** ITestsetControl

IE5091Testset

## NumberOfPowerPoints Property

---

**Description** Set and read the number of data points in each power sweep. Applies ONLY to 2D [acquisition modes](#).

**VB Syntax** `gca.NumberOfPowerPoints = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (integer) - Power points. Do not exceed the max number of points.

For 2D sweeps, total = frequency x power. Max = 20,001

For Smart sweep, total = frequency. Max = 10,000.

**Return Type** Integer

**Default** 26

**Examples** `gca.NumberOfPowerPoints = 31 'Write`

`pwrPts = gca.NumberOfPowerPoints 'Read`

**C++ Syntax** HRESULT get\_NumberOfPowerPoints(int\* pVal)  
HRESULT put\_NumberOfPowerPoints(int newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## NumberOfSweeps Property

---

**Description** Returns the number of tuning sweeps used for the latest embedded LO measurement.

**VB Syntax** *value* = *embedLODiag*.**NumberOfSweeps**

**Variable** [\(Type\)](#) - **Description**

*value* **(Long)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) **(object)**

**Return Type** **(Long)**

**Default** Not Applicable

**Examples** `data= embedLODiag.NumberOfSweeps 'read`

**C++ Syntax** HRESULT get\_NumberOfSweeps(long \* numSweeps);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

12-Apr-2007    MX New topic

## Offset Property

---

**Description** Sets and returns the offset value to be used when coupling this range to the primary range.

This setting is valid only if the specified range is [coupled](#) to the primary range.

**VB Syntax** *FOMRange.Offset = value*

**Variable** [\(Type\)](#) - Description

*object* An [FOMRange](#) (object)

*value* **(Double)** - Offset value.-(Unitless)

**Return Type** Double

**Default** 0

**Examples** `fomRange.Offset = 1e9 'Write`

`Offs = fomRange.Offset 'Read`

**C++ Syntax** HRESULT get\_Offset(double \*pVal)  
HRESULT put\_Offset(double pVal)

**Interface** IFOMRange

---

Last Modified:

8-Mar-2007    Modified links.

## OffsetReceiverAttenuator Property

---

**Description** Set and return whether to offset the reference receiver by the amount of receiver attenuation.  
[Learn more.](#)

This setting remains until changed or until the hard drive is changed or reformatted.

**VB Syntax** *pref.OffsetReceiverAttenuator = value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* (Boolean) - Choose from:

**False** Do NOT offset the test port receivers.

**True** Offset the test port receivers.

**Return Type** Boolean

**Default** **True** PNA-X models

**False** E836xB and PNA-L models

**Examples** `pref.OffsetReceiverAttenuator = 1 'Write`

`Rcvroffset = pref.OffsetReceiverAttenuator 'Read`

**C++ Syntax** HRESULT get\_OffsetReceiverAttenuator(VARIANT\_BOOL \* val);  
HRESULT put\_OffsetReceiverAttenuator(VARIANT\_BOOL val);

**Interface** IPReferences6

---

Last Modified:

14-Jan-2007    MX New topic

## OffsetSourceAttenuator Property

---

**Description** Set and return whether to mathematically offset the reference receivers by the amount of source attenuation. [Learn more.](#)

This setting remains until changed or until the hard drive is changed or reformatted.

**VB Syntax** *pref.OffsetSourceAttenuator = value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* (Boolean) - Choose from:

**False** Do NOT offset the reference receivers.

**True** Offset the reference receivers.

**Return  
Type** Boolean

**Default** **True** PNA-X models

**False** E836xB and PNA-L models

**Examples** `pref.OffsetSourceAttenuator = 1 'Write`

`Rcvroffset = pref.OffsetSourceAttenuator 'Read`

**C++ Syntax** HRESULT get\_OffsetSourceAttenuator(VARIANT\_BOOL \* val);  
HRESULT put\_OffsetSourceAttenuator(VARIANT\_BOOL val);

**Interface** IPreferences6

---

Last Modified:

14-Jan-2007    MX New topic

Read/Write

## OmitIsolation Property

---

**Description** Sets and returns whether Isolation portion of the calibration will be performed or not.

**VB Syntax** *obj.OmitIsolation = bool*

**Variable** [\(Type\) - Description](#)

*obj* [SMCType](#) (object)

or

[VMCType](#) (object)

*bool* (Boolean)

**True** - Isolation is NOT performed

**False** - Isolation is performed

**Return Type** Boolean

**Default** True

**Examples** `value = SMC.OmitIsolation`

**[C++ Syntax](#)** HRESULT put\_OmitIsolation (VARIANT\_BOOL bState)  
HRESULT get\_OmitIsolation (VARIANT\_BOOL \*bState)

**Interface** SMCType  
VMCType

## OneReadoutPerTrace Property

---

**Description** Either show marker readout of only the active trace or all of the traces simultaneously.

**VB Syntax** `win.OneReadoutPerTrace = state`

**Variable** [\(Type\)](#) - Description

*win* A [NAWindow](#) (object)

*value* **(boolean)**

**True** - Shows the readout of only the active marker for each trace.

**False** - Shows up to 5 marker readouts per trace, up to 20 total readouts.

**Return Type** Boolean

**Default** False

**Examples**

```
win.OneReadoutPerTrace = True 'Write
State = app.ActiveNAWindow.OneReadoutPerTrace 'Read
```

**C++ Syntax** HRESULT get\_OneReadoutPerTrace(VARIANT\_BOOL \*pVal)  
HRESULT put\_OneReadoutPerTrace(VARIANT\_BOOL newVal)

**Interface** INAWindow



## Options Property

---

**Description** Returns a string identifying the analyzer option configuration.

**VB Syntax** *value* = *app*.Options

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* **(string)** - variable to contain the returned string

**Return Type** String

**Default** Not Applicable

**Examples** `availOptions = app.Options`

**[C++ Syntax](#)** HRESULT Options(BSTR\* OptionString)

**Interface** IApplication

## OrientECALModule Property

---

**Description** Specifies if the PNA should perform orientation of the ECal module during calibration. Orientation is a technique by which the PNA automatically determines which ports of the module are connected to which ports of the PNA. Orientation begins to fail at very low power levels or if there is much attenuation in the path between the PNA and the ECal module.

**Note:** For guided calibrations where Orient is OFF and the same ECal module is used in more than one Connection Step, you are not allowed to specify how the ECal module is connected. Instead, the PNA determines the orientation. The PNA does not verify that you made the connection properly.

This setting remains until the PNA is restarted or this command is sent again.

This command, and [ECALPortMapEx](#), can be used to perform ECal using the [Guided Calibration](#) interface.

**VB Syntax** *cal.OrientECALModule = value*

**Variable** [\(Type\)](#) - Description

*cal* A [Calibrator](#) (object)

*value* (boolean)

**False** – [DoECAL1PortEx](#) and [DoECAL2PortEx](#) methods will use the value of the [ECALPortMapEx](#) property to determine the port connections.

**True** - [DoECAL1PortEx](#) and [DoECAL2PortEx](#) methods will use auto Orientation technique to determine port connections.

**Return Type** Boolean

**Default** True

**Examples**

```
Dim cal As Calibrator
Dim bOrient As Boolean
Set cal = PNAapp.ActiveChannel.Calibrator
cal.OrientECALModule = False 'Write
bOrient = cal.OrientECALModule 'Read
```

**C++ Syntax** HRESULT put\_OrientECALModule(VARIANT\_BOOL bOrient);  
 HRESULT get\_OrientECALModule(VARIANT\_BOOL \*bOrient);

**Interface** ICalibrator3

---

Last Modified:

7-May-2007    Added note about Guided Cal.

## OutputFixedFrequency Property

---

<b>Description</b>	Sets or returns the mixer output fixed frequency value.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .OutputFixedFrequency = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Output Fixed Frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Print mixer.OutputFixedFrequency 'prints the output fixed frequency value of the mixer.</pre>
<b>C++ Syntax</b>	HRESULT get_OutputFixedFrequency(double *pVal) HRESULT put_OutputFixedFrequency(double newVal)
<b>Interface</b>	IMixer3 IConverter

---

### Last Modified:

2-Feb-2009    Added Converter

4-Mar-2008    Added note.

## OutputPort Property

---

**Description** Switches an input to one of the valid outputs on the specified E5091A. The following are valid input/output combinations. If a combination other than these are sent, an “invalid argument” error will occur.

Input	Valid Outputs
1	<b>A</b>
	<b>T1</b> - If Port 2 already is connected to T1, then Port 2 will be switched to T2.)
2	<b>T1</b> - If Port 1 already is connected to T1, then Port 1 will be switched to A.
	<b>T2</b>
3	<b>R1+</b>
	<b>R2+</b>
	<b>R3+</b> If option 007 (7port), R2 is selected.
4	<b>R1-</b>
	<b>R2-</b>
	<b>R3-</b> If option 007 (7port), R2 is selected.

**Note:** Do not confuse the similar Testset [OutputPorts Property](#), which sets or gets the port mapping for ALL ports.

**VB Syntax** `testsets(1).OutputPort (chNum,input) = output`

**Variable** [\(Type\)](#) - **Description**

*testsets(1)* An item from [Testsets](#) (**collection**)  
Learn how to [identify a testset in the collection](#).

*chNum* (**Long**) Channel number of the measurement.

*input* (**Long**) Testset Input port. Choose from 1|2|3|4.

*output* (Enum as NAE5091OutputPort) Output port to switch to specified Input. Choose from:  
**0** or **naE5091PortA** - Port A  
**1** or **naE5091PortT1** - Port T1  
**2** or **naE5091PortT2** - Port T2

**3** or **naE5091PortR1** - Port R1  
**4** or **naE5091PortR2** - Port R2  
**5** or **naE5091PortR3** - Port R3

**Return Type** Enum

**Default** Not Applicable

**Examples** [See Example Program](#)

**C++ Syntax** HRESULT get\_OutputPort(long channelNum, long inputPort, E5091OutputPort \*outPort);  
HRESULT put\_OutputPort(long channelNum, long inputPort, E5091OutputPort outPort);

**Interface** IE5091Testsets

## OutputPorts (Cal Set) Property

---

**Description** Returns the port mapping for the Cal Set.

**VB Syntax** *portMap* = *calset*.**OutputPorts**

**Variable** [\(Type\)](#) - **Description**

*portMap* (String) Variable to store the returned string. The returned values are the physical ports. The POSITION of the returned values corresponds to the logical ports.

For example, with an N44xx test set, if the returned string is "PNA 1,TS 2,PNA 2, TS 4" this means:

- PNA 1 is assigned to logical port 1
- TS 2 is assigned to logical port 2
- PNA 2 is assigned to logical port 3
- TS 4 is assigned to logical port 4

*calset* A [Cal Set](#) object.

**Return Type** String

**Default** Depends on the test set.

**Example** `portMap = calset.OutputPorts`

**C++ Syntax** HRESULT get\_OutputPorts(BSTR \*mapping);

**Interface** ICalses5

---

Last modified:

9/18/06 MQ Added for multiport

## OutputPorts Property

---

**Description** Sets or gets the port mappings for ALL ports. An “invalid argument” error will occur if you attempt to set an illegal port combination.

Refer to your testset documentation for valid port combinations.

**Note:** Do not confuse the similar E5091 [OutputPort Property](#), which sets or gets the port mapping for a single port.

**VB Syntax** *tset.OutputPorts(chNum) = portList*

**Variable** [\(Type\)](#) - Description

*tset* A [TestsetControl](#) object.

*chNum* **(Long)** Channel number of the measurement.

*portList* **(String)** A comma-separated list of port mappings. Spaces are ignored at the beginning and end of this text, and before or after commas. Space characters in other locations are not ignored.

**Return Type** String

**Default** Not Applicable

**Examples** [See External Testset Program](#)

**C++ Syntax** HRESULT get\_OutputPorts(long channelNum, BSTR \*outPorts);  
HRESULT put\_OutputPorts(long channelNum, BSTR outPorts);

**Interface** ITestsetControl



## OutputRangeMode Property

---

**Description** Sets or returns the Output sweep mode.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj.OutputRangeMode = value*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)  
Or  
A [Converter Object](#)

*value* **(Enum as MixerRangeMode)** - Output sweep mode. Choose from:  
0 - **mixSwept**  
1 - **mixFixed**

**Return Type** Enum

**Default** 0 - mixSwept

**Examples** `mixer.OutputRangeMode = mixSwept`

**C++ Syntax** HRESULT get\_OutputRangeMode(long \*pVal)  
HRESULT put\_OutputRangeMode(long newVal)

**Interface** IMixer6  
IConverter

---

Last Modified:

2-Feb-2009 Added Converter

4-Mar-2008 Added note.

## OutputSideband Property

---

**Description** Specify whether to select the sum (High) or difference (Low) products.

- When one LO is used: Input + or - LO1 = Output frequency

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *obj*.OutputSideband = *value*

**Variable** [\(Type\)](#) - Description

*obj* A [Converter Object](#)

*value* (**enum as ConverterSideBand**) - Choose from:

**0** - or **naLowSide** Minus (-) on the Mixer setup dialog

**1** - or **naHighSide** Plus (+) on the Mixer setup dialog

**Return Type** Enum as ConverterSideBand

**Default** naLowSide

**Examples** `Print converter.OutputSideband` 'prints the value of the OutputSideband

**C++ Syntax** HRESULT get\_OutputSideband(ConverterSideBand \*pVal)  
HRESULT put\_OutputSideband(ConverterSideBand newVal)

**Interface** IConverter

---

Last Modified:

26-Mar-2009    New topic

## OutputSideband Property

---

**Description** Specify whether to select the sum (High) or difference (Low) products.

- When one LO is used: Input + or - LO1 = Output frequency
- When two LOs are used: IF1 + or - LO2 = Output frequency

Use [IFSideband\\_Property](#) when two LOs are used to determine the IF1 frequency.

If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

**VB Syntax** *mixer*.**OutputSideband** = *value*

**Variable** [\(Type\)](#) - Description

*mixer* A [Mixer](#) (object)

*value* (**FCASideBand**) -

Choose from:

**0 - LOW** Minus (-) on the Mixer setup dialog

**1 - HIGH** Plus (+) on the Mixer setup dialog

**Return Type** Enum as FCASideBand

**Default** LOW

**Examples** `Print mixer.OutputSideband` 'prints the value of the `OutputSideband`

**C++ Syntax** HRESULT get\_OutputSideband(FCASideBand \*pVal)  
HRESULT put\_OutputSideband(FCASideBand newVal)

**Interface** IMixer

---

Last Modified:

2-Oct-2008 Clarified 1 vs 2 LOs

4-Mar-2008 Added note.

## OutputStartFrequency Property

---

<b>Description</b>	Sets or returns the mixer output start frequency.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .OutputStartFrequency = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Output Start Frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Print mixer.OutputStartFrequency 'prints the value of the OutputStartFrequency</pre>
<b>C++ Syntax</b>	HRESULT get_OutputStartFrequency(double *pVal) HRESULT put_OutputStartFrequency(double newVal)
<b>Interface</b>	IMixer IConverter

---

### Last Modified:

2-Feb-2009    Added Converter

4-Mar-2008    Added note.

## OutputStopFrequency Property

---

<b>Description</b>	Sets or returns the mixer Output Stop frequency.  If you are changing several mixer configuration settings, you can make all the changes first and then issue the <a href="#">Calculate</a> and <a href="#">Apply</a> commands as you would do from the user interface.
<b>VB Syntax</b>	<i>obj</i> .OutputStopFrequency = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>obj</i>	A <a href="#">Mixer Interface</a> pointer to the <a href="#">Measurement</a> (object) Or A <a href="#">Converter Object</a>
<i>value</i>	<b>(double)</b> - Output stop frequency in Hertz.
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Print mixer.OutputStopFrequency 'prints the value of the OutputStopFrequency</pre>
<b>C++ Syntax</b>	HRESULT get_OutputStopFrequency(double *pVal) HRESULT put_OutputStopFrequency(double newVal)
<b>Interface</b>	IMixer IConverter

---

### Last Modified:

2-Feb-2009    Added Converter

4-Mar-2008    Added note.

Read-only

## Parameter Property

---

**Description** Returns the measurement Parameter. To change the parameter, use meas.[ChangeParameter](#)

**VB Syntax** *measPar* = meas.**Parameter**

**Variable** [\(Type\)](#) - **Description**

*measPar* **(string)** - Variable to store Parameter string

*meas* A Measurement **(object)**

**Return Type** String

**Default** Not applicable

**Examples** `measPar = meas.Parameter 'Read`

**C++ Syntax** HRESULT get\_Parameter(BSTR \*pVal)

**Interface** IMeasurement

## Parameter (Embedded LO) Property

---

**Description** Returns the name of the parameter of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**Parameter** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.Parameter 3 'read`

**C++ Syntax** HRESULT Parameter(long sweep, BSTR \* param);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## Parent Property

---

**Description** Returns a handle to the parent object of the collection object being referred to in the statement. The parent property allows the user to traverse from an object back up the object hierarchy.

**VB Syntax** *collection*.**Parent**

**Variable** [\(Type\)](#) - **Description**

*collection*

- [CalFactorSegments collection](#)
- [CalFactorSegmentsPMAR Collection](#)
- [Channels collection](#)
- [E5091Testset Collection](#)
- [ECalModules Collection](#)
- [ExternalDevices Collection](#)
- [ExternalTestsets Collection](#)
- [Measurements collection](#)
- [NaWindows collection](#)
- [PowerLossSegments collection](#)
- [PowerLossSegmentsPMAR Collection](#)
- [PowerSensors collection](#)
- [Segments collection](#)
- [Traces collection](#)
- [PathConfigurationManager](#)
- [PowerMeterInterfaces Collection](#)

**Return Type** Object

**Default** Not Applicable

**Examples** `parentobj = chans.Parent 'returns a handle to the parent object (Application) of the chans collection. -Read`

**C++ Syntax**

```
HRESULT get_Parent(IApplication* *pApplication); //IChannels, IChannel,
IMeasurements, INAWindows, and IExternalDevices
HRESULT get_Parent(IChannel* *pChannel); //ITraces
HRESULT get_Parent(INAWindow* *pWindow); //ISegments
HRESULT get_Parent(IPowerSensor* *pSensor); //ICalFactorSegments(PMAR)
HRESULT get_Parent(ISourcePowerCalibrator* *pCalibrator); //IPowerSensors,
IPowerLossSegments, and IPowerMeterInterfaces
```



**Interface** All listed above

---

Last Modified:

31-Jul-2009 Added ExternalDevices (9.0)

5-Mar-2009 Added PowerMeterInterfaces

## PassFailLogic Property

---

**Description** Sets and reads the logic of the PassFail line on the [HANDLER IO connector](#) (pin 33) and [AUX IO connector](#) (pin 12). [Learn more about Global Pass/Fail.](#)

**Note:** This line is connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *object*.PassFailLogic = *value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** - An Aux I/O or Handler I/O object

*value* **(enum as NARearPanelIOLogic)** Choose from:

**0 - naPositiveLogic** - Causes the PassFail line to have positive logic (high = pass, low = fail).

**1 - naNegativeLogic** - Causes the PassFail line to have negative logic (high = fail, low = pass).

**Return Type** Long Integer

**Default** naPositiveLogic

**Examples**

```
aux.PassFailLogic = naNegativeLogic 'Write
Text1.Text = aux.PassFailLogic 'Read
```

**C++ Syntax** HRESULT put\_PassFailLogic ( tagNARearPanelIOLogic Mode );  
 HRESULT get\_PassFailLogic ( tagNARearPanelIOLogic\* Mode );

**Interface** IHWAuxIO  
 IHWMaterialHandlerIO

## PassFailMode Property

---

**Description** Sets and reads the mode of the PassFail line on the [HANDLER IO connector](#) (pin 33) and [AUX IO connector](#) (pin 12). [Learn more about Global Pass/Fail.](#)

**Note:** This line is connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *object*.**PassFailMode** = *value*

**Variable** [\(Type\)](#) - **Description**

*object* **(object)** - An Aux I/O or Handler I/O object

*value* **(enum as NAPassFailMode)**. Choose from:  
**0 - naDefaultPassNoWaitMode**- the line stays in PASS state. When a device fails, then the line goes to fail IMMEDIATELY.  
**1 - naDefaultPassWaitMode** - the line stays in PASS state. When a device fails, then the line goes to fail after the Sweep End line is asserted.  
**2 - naDefaultFailWaitMode**- the line stays in FAIL state. When a device passes, then the line goes to PASS state after the Sweep End line is asserted.

**Return Type** Long Integer

**Default** 0 - naDefaultPassNoWaitMode

**Examples**

```
HWAuxIO.PassFailMode = naDefaultPassNoWaitMode 'Write
mode = HWAuxIO.PassFailMode 'Read
```

**C++ Syntax** HRESULT put\_PassFailMode ( tagNAPassFailMode Mode );  
 HRESULT get\_PassFailMode ( tagNAPassFailMode\* Mode );

**Interface** IHWAuxIO  
 IHWMaterialHandlerIO

## PassFailPolicy Property

---

<b>Description</b>	Sets the policy used to determine how global pass/fail is computed. <a href="#">Learn more about Global Pass/Fail.</a>
<b>VB Syntax</b>	<i>object</i> . <b>PassFailPolicy</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>object</i>	<b>(object)</b> - An <a href="#">Aux I/O</a> or <a href="#">Handler I/O</a> object
<i>value</i>	<b>(enum as NAPassFailPolicy)</b> Choose from: 0 - <b>naPolicyAllTests</b> - - Pass/Fail Status returns PASS if all tests on all measurements pass. 1 - <b>naPolicyAllMeas</b> - Pass/Fail Status returns PASS if all measurements have associated tests, and all tests pass. FAIL is returned if even one measurement has no associated limit test. Only those measurements which are not in HOLD mode contribute to the pass/fail result.
<b>Return Type</b>	Long Integer
<b>Default</b>	naPolicyAllTests
<b>Examples</b>	<pre>matHndler.PassFailPolicy = naPolicyAllTests 'Write</pre> <pre>policy = aux.PassFailPolicy 'Read</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT put_PassFailPolicy ( tagNAPassFailPolicy Policy); HRESULT get_PassFailPolicy ( tagNARearPanelIOLogic* Policy);
<b>Interface</b>	IHWAuxIO4 IHWMaterialHandlerIO2

## PassFailScope Property

---

**Description** Sets and reads the Scope of the PassFail line on the [HANDLER IO connector](#) (pin 33) and [AUX IO connector](#) (pin 12). [Learn more about Global Pass/Fail.](#)

**Note:** The PassFail line is connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *object.PassFailScope = value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** - An Aux I/O or Handler IO object

*value* **(enum NAPassFailScope )** Choose from:

**0 - naChannelScope** - The PassFail line returns to its default state before sweeps on the next channel start. (A channel measurement may require several sweeps.)

**1 - naGlobalScope** - The PassFail line returns to its default state before the sweeps for the next [triggerable](#) channel start.

The default state of the PassFail line before a measurement occurs and after a failure occurs is set by the [PassFailMode](#) property.

**Return Type** enum NAPassFailScope

**Default** 1 - naGlobalScope

**Examples**

```
HWAuxIO.PassFailScope = naGlobalScope 'Write
scope = HWAuxIO.PassFailScope 'Read
```

**C++ Syntax**

```
HRESULT put_PassFailScope ( tagNAPassFailScope Scope );
HRESULT get_PassFailScope ( tagNAPassFailScope* Scope );
```

**Interface**

```
IHWAuxIO
IHWMaterialHandlerIO
```

## PassFailStatus Property

---

**Description** Returns the most recent pass/fail status value. Use this command as follows:

1. Set the PNA [trigger scope](#) to GLOBAL
2. Set the PNA [trigger source](#) to MANUAL or EXTERNAL.
3. Configure and enable [Limit Testing](#)
4. Trigger the PNA.
5. Use the [\\*OPC?](#) (with [SCPIStringParser object](#)) to determine when the sweep is complete.
6. Use the **PassFailStatus** property to obtain the global pass/fail result.

[Learn more about Global Pass/Fail.](#)

**VB Syntax** `var = object.PassFailStatus`

**Variable** [\(Type\)](#) - **Description**

**var** (enum as NAPassFailStatus) Variable to store returned status. One of the following will be returned:

**0 - naStatusFail** - all measurements not in HOLD mode have been swept, and one or more limit tests failed according to the specified [Pass/Fail policy](#).

**1 - naStatusPass** - all measurements not in HOLD mode have been swept, and all associated limit tests have passed.

**2 - naStatusNone** - status cannot be determined because measurements are in progress.

**object** **(object)** - An [Aux I/O](#) or [Handler I/O](#) object

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `status = aux.PassFailStatus 'Read`

**C++ Syntax** `HRESULT get_PassFailPolicy ( tagNAPassFailStatus* status);`

**Interface** IHWAuxIO4  
IHWMaterialHandlerIO2

## Path Property

---

**Description** Specifies an interface to use for the power meter / sensor during a source power calibration.

**VB Syntax** *pwrMtrInterface.Path = value*

**Variable** [\(Type\)](#) - **Description**

*pwrMtrInterface* **(object)** - A [PowerMeterInterface](#) (object)

*value* **(enum as NACommunicationPath)** Choose from:

**0 - naGPIB** - GPIB interface

**1 - naUSB** - USB interface

**2 - naLAN** - LAN interface

**Return Type** Enum

**Default** naGPIB

**Examples** [See example](#)

**[C++ Syntax](#)** HRESULT put\_Path(tagNACommunicationPath pNewVal);  
HRESULT get\_Path(tagNACommunicationPath \*pVal);

**Interface** IPowerMeterInterface

---

Last Modified:

2-Aug-2007    MX New topic

## PathCalMethod Property

---

**Description** **Note:** This command replaces [ThruCalMethod](#).

(Read-Write) Specifies the calibration method for each port pair.

**Note:** Before using this command, first do the following:

- Set the connector types: [ConnectorType](#)
- Set cal kit: [CalKitType](#)
- Set or query the thru path pairs: [ThruPortList](#)

After sending or querying this command, send the Thru method: [PathThruMethod](#)

See an example of a [4-port guided calibration using COM](#).

**VB Syntax** *guidedCal.PathCalMethod (port1, port2) = "caltype1[,caltype2]"*

**Variable** [\(Type\)](#) - [Description](#)

*guidedCal* [GuidedCalibration](#) (object)

*port1* First port of the pair to be calibrated.

*port2* Second port of the pair to be calibrated.

*"caltype1[,caltype2]"* (String) Cal types for 1st and 2nd ports of the pair, enclosed in a single pair of quotes. NOT case-sensitive.

**caltype1** Cal type for the pair if caltype2 is not specified. Otherwise, Cal type for port 1. Choose from:

- "TRL"
- "SOLT"
- "QSOLTN"
- "EnhRespN"

For the last two arguments, replace **N** with the port to be used as the source port, which MUST be one of the port pair.

**[caltype2]** Optional argument. Use only when performing an adapter removal cal on the pair. This argument specifies the Cal Type on the second port; **caltype1** then specifies the Cal Type of the first port.

Choose from the same arguments as **caltype1**.

**Return Type** **String** - Returns comma-separated cal types.



**Default** Not Applicable

**Example** `guidedCal.PathCalMethod(2,3) = "TRL" 'Write trl for port pair`  
`guidedCal.PathCalMethod(1,4) = "TRL,SOLT" 'Write adapter removal`  
`cal, consisting of trl on port 1 and solt on port 4`

`calmethod = guided.PathCalMethod(1,4) 'Read previous example,`  
`returns: "TRL,SOLT"`

**C++ Syntax** HRESULT get\_PathCalMethod(long firstport, long secondport, BSTR \*calMethod);  
HRESULT put\_PathCalMethod(long firstport, long secondport, BSTR calMethod);

**Interface** IGuidedCalibration3

---

Last modified:

April 9, 2007    MX New topic

## PathThruMethod Property

---

**Description** **Note:** This command replaces [ThruCalMethod](#).

(Read-Write) Specifies the calibration THRU method for each port pair.

Before using this command, first do the following:

- Set the connector types: [ConnectorType](#)
- Set cal kit: [CalKitType](#)
- Set or query the thru path pairs: [ThruPortList](#)
- Set or query the Cal Type: [PathCalMethod](#)

See an example of a [4-port guided calibration using COM](#).

**VB Syntax** *guidedCal.PathThruMethod (port1, port2) = "ThruType1[,ThruType2]"*

**Variable** [\(Type\) - Description](#)

*guidedCal* [GuidedCalibration](#) (object)

*port1* First port of the pair to be calibrated.

*port2* Second port of the pair to be calibrated.

*"ThruType1[,ThruType2]"* (String) Thru methods for 1st and 2nd ports of the pair, enclosed in a single pair of quotes. NOT case-sensitive.

**thruType1** Calibration thru method for the pair if thruType2 is not specified. Otherwise, thru method for port 1.

Choose from:

- **“Defined Thru”** A thru type for which there is a stored definition in the Cal Kit.
- **“Zero Thru”** Zero length thru, also known as flush-thru.
- **“Undefined Thru”** A thru type for which there is NOT a stored definition in the Cal Kit. Also known as Unknown Thru. Valid ONLY for SOLT [cal type](#).
- **“Undefined Thru using a Defined Thru”** Using an ECal module, measure the internal thru using the "Undefined Thru" method.

**ThruType2** (String) Optional argument. Use only when performing an adapter removal cal on the pair as determined by [PathCalMethod](#). The only valid arguments for ThruType1&2 is "Defined Thru, Defined Thru".

**Return Type** **String** - Returns comma-separated ThruTypes.

**Default** Not Applicable

**Example**

```
guidedCal.PathThruMethod(2,3) = "Zero Thru" 'Write for port pair  
guidedCal.PathThruMethod(1,4) = "Defined Thru, Defined Thru"  
'Write for adapter removal cal.  
  
calmethod = guided.PathThruMethod(1,4) 'Read previous example,  
return: "Defined Thru, Defined Thru"
```

**C++ Syntax** HRESULT get\_PathThruMethod(long firstport, long secondport, BSTR \*thruMethod);  
HRESULT put\_PathThruMethod(long firstport, long secondport, BSTR thruMethod);

**Interface** IGuidedCalibration3

---

Last modified:

April 9, 2007    MX New topic

## PeakExcursion Property

---

**Description** Sets and reads the peak excursion value for the specified marker. The Excursion value determines what is considered a "peak".

**VB Syntax** *mark*.**PeakExcursion** = *value*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**single**) - Peak Excursion. Choose any number between **-500** and **500**

**Return Type** Single

**Default** 3

**Examples** `mark.PeakExcursion = 1 'Write`

`PkExcur = mark.PeakExcursion 'Read`

**C++ Syntax** HRESULT get\_PeakExcursion(float \*pVal)  
HRESULT put\_PeakExcursion(float newVal)

**Interface** IMarker

## PeakThreshold Property

---

**Description** Sets peak threshold for the specified marker. If a peak (using the criteria set with [PeakExcursion](#)) is below this reference value, it will not be considered when searching for peaks.

**VB Syntax** *mark*.**PeakThreshold** = *value*

**Variable** [\(Type\)](#) - **Description**

*mark* A Marker (**object**)

*value* (**single**) - Peak Threshold. Choose any number between: **-500** and **500**

**Return Type** Single

**Default** -100db

**Examples** `mark.PeakThreshold = 1 'Write`

`PkThresh = mark.PeakThreshold 'Read`

**C++ Syntax** HRESULT get\_PeakThreshold(float \*pVal)  
HRESULT put\_PeakThreshold(float newVal)

**Interface** IMarker

## PeakToPeak Property

---

**Description** Returns the Peak to Peak value of the measurement. To retrieve all 3 statistics value at the same time, use [meas.GetTraceStatistics](#)

**VB Syntax** *pp* = *meas*.**PeakToPeak**

**Variable** [\(Type\)](#) - **Description**

*pp* **(single)** - Variable to store peak-to-peak value

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples** `pp = meas.PeakToPeak 'Read`

**C++ Syntax** HRESULT get\_PeakToPeak(float\* pp)

**Interface** IMeasurement

## Period Property

---

**Description** Sets the pulse-period (1/PRF) for ALL PNA-X internal pulse generators.  
The resolution of the period is 16.667nS.

**VB Syntax** *pulse.Period = value*

**Variable** [\(Type\)](#) - Description

*pulse* A [PulseGenerator](#) (object)

*value* **(Double)** Pulse period in seconds. Choose a value from about 33ns to about 70 seconds.

**Return Type** Double

**Default** 1e-3 sec

**Examples** `pulse.Period = 1ms 'Write`

`value = pulse.Period 'Read`

**C++ Syntax** HRESULT get\_Period(double\* period);  
HRESULT put\_Period(double period);

**Interface** IPulseGenerator

---

Last Modified:

1-Jan-2007    MX New topic

## PhaseOffset Property

---

**Description** Sets the Phase Offset for the active channel.

**VB Syntax** *meas.PhaseOffset* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement **(object)**

*value* **(double)** - PhaseOffset in degrees. Choose any number between:  
**-360** and **+360**

**Return Type** Double

**Default** 0

**Examples** `meas.PhaseOffset = 25 'Write`

`poffset = meas.PhaseOffset 'Read`

**C++ Syntax** HRESULT get\_PhaseOffset(double \*pVal)  
HRESULT put\_PhaseOffset(double newVal)

**Interface** IMeasurement



## PhaseAsFixture Property

---

<b>Description</b>	Sets and reads the state of phase offset as a fixture with True Mode balanced measurements. <a href="#">Learn more about iTMSA phase and power offset.</a>
<b>VB Syntax</b>	<i>balStim</i> . <b>PhaseAsFixture</b> = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>balStim</i>	A <a href="#">BalancedStimulus</a> ( <b>object</b> )
<i>value</i>	<b>(Boolean)</b> State of phase offset as a fixture. <b>False</b> Offset is applied but is NOT included as a fixture in the output calculations. <b>True</b> Offset is applied and included as a fixture in the output calculations.
<b>Return Type</b>	Boolean
<b>Default</b>	False
<b>Examples</b>	<pre>balStim.PhaseAsFixture = False 'Write var = balStim.PhaseAsFixture 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_PhaseAsFixture (VARIANT_BOOL *bVal) HRESULT put_PhaseAsFixture (VARIANT_BOOL bVal)
<b>Interface</b>	IBalancedStimulus

---

Last Modified:

27-May-2008    MX New topic

## PhaseSwpAsFixture Property

---

**Description** Sets and reads the state of phase offset as a fixture with True Mode balanced measurements.

**VB Syntax** *balStim.PhaseAsFixture = value*

**Variable** [\(Type\)](#) - [Description](#)

*balStim* A [BalancedStimulus](#) (object)

*value* **(Boolean)** State of phase offset as a fixture.

**False** Offset is applied but is NOT included as a fixture in the output calculations.

**True** Offset is applied and included as a fixture in the output calculations.

**Return Type** Boolean

**Default** False

**Examples**

```
balStim.PhaseAsFixture = False 'Write
var = balStim.PhaseAsFixture 'Read
```

**C++ Syntax** HRESULT get\_PhaseAsFixture (VARIANT\_BOOL \*bVal)  
HRESULT put\_PhaseAsFixture (VARIANT\_BOOL bVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## PhaseSwpState Property

---

**Description** Sets and reads the state of phase sweep. [Sweep type](#) must be set to CWTime.

**VB Syntax** *balStim.PhaseSwpState* = *value*

**Variable** [\(Type\)](#) - **Description**

*balStim* A [BalancedStimulus](#) (object)

*value* **(Boolean)** State of phase sweep.

**False** Phase sweep disabled.

**True** Phase sweep enabled.

**Return Type** Boolean

**Default** False

**Examples** `balStim.PhaseSwpState = False` 'Write

`var = balStim.PhaseSwpState` 'Read

**C++ Syntax** HRESULT get\_PhaseSwpState (VARIANT\_BOOL \*bVal)

HRESULT put\_PhaseSwpState (VARIANT\_BOOL bVal)

**Interface** IBalancedStimulus2

---

Last Modified:

3-Mar-2009    MX New topic (A.08.50)

## PointAveragingState Property

---

**Description** Turns point averaging ON or OFF for all measurements on the channel.

**VB Syntax** *chan.PointAveragingState = state*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*state* (Enum)

**False** - Turns point averaging OFF

**True** - Turns point averaging ON

**Return Type** Boolean

**Default** False

**Examples** `chan.PointAveragingState = True 'Write`

`averg = chan.PointAveragingState 'Read`

**C++ Syntax** HRESULT get\_PointAveragingState(BOOL \*pVal)  
HRESULT put\_PointAveragingState(BOOL newVal)

**Interface** IChannel16

---

Last Modified:

7-Oct-2008    MX New topic

## PointSweepState Property

---

**Description** Turns point sweep ON or OFF for all measurements on the channel. Point sweep measures both the forward and reverse parameters at each frequency point before stepping to the next frequency. The display trace is updated after the forward and reverse parameters are measured at that frequency point.

**VB Syntax** *chan.PointSweepState = state*

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (object)

*state* (boolean)

**False** - Turns point sweep OFF

**True** - Turns point sweep ON

**Return Type** Boolean

**Default** False

**Examples** `chan.PointSweepState = True 'Write`

`averg = chan.PointSweepState 'Read`

**C++ Syntax** HRESULT get\_PointSweepState(VARIANT\_BOOL \*pVal)  
HRESULT put\_PointSweepState(VARIANT\_BOOL newVal)

**Interface** IChannel16

---

Last Modified:

3-Nov-2008    MX New topic

## Port1 Property **Superseded**

---

**Description** This command is replaced by [PortDelay property](#).

Sets a Port Extension value for Port 1

**VB Syntax** *portExt.Port1 = value*

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.Port1 = 10e-6 'Write`

`prt1 = portExt.Port1 'Read`

**C++ Syntax** HRESULT get\_Port1(double \*pVal)  
HRESULT put\_Port1(double newVal)

**Interface** IPortExtension

## Port2 Property **Superseded**

---

**Description** This command is replaced by [PortDelay property](#).

Sets a Port Extension value for Port 2

**VB Syntax** `portExt.Port2 = value`

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.Port2 = 10e-6 'Write`

`prt2 = portExt.Port2 'Read`

**C++ Syntax** HRESULT get\_Port2(double \*pVal)  
HRESULT put\_Port2(double newVal)

**Interface** IPortExtension

## Port3 Property **Superseded**

---

**Description** This command is replaced by [PortDelay property](#).

Sets a Port Extension value for Port 3

**VB Syntax** *portExt.Port3 = value*

**Variable** [\(Type\)](#) - Description

*portExt* A Port Extension (**object**)

*value* (**double**) - Port Extension value in seconds. Choose any number between **-10** and **10**

**Return Type** Double

**Default** 0

**Examples** `portExt.Port3 = 10e-6 'Write`

`prt3 = portExt.Port3 'Read`

**C++ Syntax** HRESULT get\_Port3(double \*pVal)  
HRESULT put\_Port3(double newVal)

**Interface** IPortExtension



## Port1NoiseTunerSwitchPresetsToExternal Property

---

**Description** Sets the default setting for the Noise Tuner switch. [Learn more.](#)

**VB Syntax** *pref.Port1NoiseTunerSwitchPresetsToExternal = bool*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*bool* **(Boolean)** - Choose from:

**False** - Sets the default (preset) to INTERNAL

**True** - Sets the default (preset) to EXTERNAL

**Return Type** Boolean

**Default** True

**Examples** `pref.Port1NoiseTunerSwitchPresetsToExternal = False 'Write`

`prefer = pref.Port1NoiseTunerSwitchPresetsToExternal 'Read`

**C++ Syntax** HRESULT put\_Port1NoiseTunerSwitchPresetsToExternal( VARIANT\_BOOL bValue)  
HRESULT get\_Port1NoiseTunerSwitchPresetsToExternal( VARIANT\_BOOL \*bValue)

**Interface** IPreferences8

---

Last Modified:

1-Feb-2008    MX New topic

## Port2PdeembedCktModel Property

---

**Description** Select whether or not to load a 2-port De-embedding circuit model for the specified port number. Circuit model is applied when both "USER" is selected and the filename is specified. To set the filename, use [strPort2Pdeembed\\_S2PFile Property](#)

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.Port2PdeembedCktModel(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive circuit model.

*value* (Enum as NAFixturing2PdeembedCkt)

0 **naFix2PD\_USER** load a 2-port De-embedding circuit model

1 **naFix2PD\_NONE** no model

**Return Type** Long Integer

**Default** naFix2PD\_NONE

**Examples** `fixture.Port2PdeembedCktModel(2) = naFix2PD_USER 'Write`

`value = fixture.Port2PdeembedCktModel(1) 'Read`

**C++ Syntax** HRESULT get\_Port2PdeembedCktModel(short port tagNAFixturing2PdeembedCkt \*pVal)  
HRESULT put\_Port2PdeembedCktModel(short port tagNAFixturing2PdeembedCkt newVal)

**Interface** IFixturing

## Port2PdeembedState Property

---

**Description** Turns de-embedding ON or OFF for all ports on the channel.

**VB Syntax** *fixture.Port2PdeembedState = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns De-embedding OFF

**True** - Turns De-embedding ON

**Return Type** Boolean

**Default** False

**Examples** `fixture.Port2PdeembedState = False 'Write`

`value = fixture.Port2PdeembedState 'Read`

**C++ Syntax** HRESULT get\_Port2PdeembedState(VARIANT\_BOOL \*pVal)  
HRESULT put\_Port2PdeembedState(VARIANT\_BOOL newVal)

**Interface** IFixturing

## PortArbzMImag Property

---

**Description** Sets and returns the Imaginary portion of the impedance value for the specified single-ended port. Use [PortArbzReal](#) to set the real value. Or use [PortArbzZ0](#) to set both values together.

**VB Syntax** *fixture*.PortArbzImag(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Single-ended port number to receive impedance value.

*value* **(Double)** Real Impedance value. Choose a value between -1E18 and 1E18

**Return Type** Double

**Default** 0

**Examples** `fixture.PortArbzImag(2) = 75 'Write`

`value = fixture.PortArbzImag(1) 'Read`

**C++ Syntax** HRESULT get\_PortArbzImag( short portNum, double \*pVal)  
HRESULT put\_PortArbzImag( short portNum, double newVal)

**Interface** IFixturing3

## PortArbzReal Property

---

**Description** Sets and returns the Real portion of the impedance value for the specified single-ended port. Use [PortArbzImag](#) to set the imaginary value. Or use [PortArbzZ0](#) to set both values together.

**VB Syntax** *fixture*.PortArbzReal(*portNum*) = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Single-ended port number to receive impedance value.

*value* **(Double)** Real Impedance value. Choose a value between 0 to 1E7

**Return  
Type** Double

**Default** 50

**Examples** `fixture.PortArbzReal(2) = 75 'Write`

`value = fixture.PortArbzReal(1) 'Read`

**C++ Syntax** HRESULT get\_PortArbzReal( short portNum, double \*pVal)  
HRESULT put\_PortArbzReal( short portNum, double newVal)

**Interface** IFixturing3

## PortArbzState Property

---

**Description** Turns Port Impedance ON or OFF for all ports on the channel.

**VB Syntax** *fixture.PortArbzState = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns Port Impedance OFF

**True** - Turns Port Impedance ON

**Return Type** Boolean

**Default** False

**Examples** `fixture.PortArbzState = False 'Write`

`value = fixture.PortArbzState 'Read`

**C++ Syntax** HRESULT get\_PortArbzState(VARIANT\_BOOL \*pVal)  
HRESULT put\_PortArbzState(VARIANT\_BOOL newVal)

**Interface** IFixturing

## PortArbZ0 Property

---

**Description** Sets and returns the Real portion of the impedance value for the specified single-ended port. The imaginary portion is automatically set to 0.0.

To set both values separately, use [PortArbZReal](#) and [PortArbZImag](#).

**VB Syntax** *fixture.PortArbZ0(portNum) = value*

**Variable** [\(Type\)](#) - **Description**

*fixture* A [Fixturing](#) (object)

*portNum* **(Integer)** Single-ended port number to receive impedance value.

*value* **(Double)** Impedance value. Choose a value between 0 to 1E7

**Return  
Type** Double

**Default** 50

**Examples** `fixture.PortArbZ0(2) = 75 'Write`

`value = fixture.PortArbZ0(1) 'Read`

**C++ Syntax** HRESULT get\_PortArbZ0( short portNum, double \*pVal)  
HRESULT put\_PortArbZ0( short portNum, double newVal)

**Interface** IFixturing3

## PortCatalog Property

---

**Description** Returns a comma-separated list of the Output port selections that are available for a given logical input port.

Read the number of input ports for the test set using [NumberOfPorts Property](#).

**VB Syntax** *value* = *tset*.**PortCatalog**(*logPort*)

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to store the returned information.

*tset* A [TestsetControl](#) object.

*logPort* **(Long)** Logical Input port number for which to return valid output ports.

**Return Type** **String**

**Default** Not Applicable

**Examples** `value = testset1.PortCatalog 2`

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_PortCatalog(long inputPort, BSTR \*outPort);

**Interface** ITestsetControl

---

Last Modified:

18-Jan-2007    Fixed argument



## PortCLogic Property

---

**Description** Sets and reads the logic mode of Port C on the AUX IO connector and the Handler IO connector.

**Note:** Port C lines are connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *AuxIO.PortCLogic = value*

**Variable** [\(Type\)](#) - Description

*AuxIO* **(object)** - A Hardware Aux I/O object

*value* **(Enum as NaRearPanelIOLogic)** - Choose from:

**0 - naPositiveLogic** - The associated data line goes **HIGH** when writing a 1 to a PortC bit.

**1 - naNegativeLogic** - The associated data line goes **LOW** when writing a 1 to a PortC bit.

When Port C is in Output/Write mode, a change in logic causes the output lines to change state immediately. For example, Low levels change to High levels.

When Port C is in Input/Read mode, a change in logic will not cause the lines to change, but data read from Port C will reflect the change in logic.

**Return Type** Enum

**Default** 1 - naNegativeLogic

**Examples**

```
auxIO.PortCLogic = value 'Write
value = auxIo.PortCLogic 'Read
```

**C++ Syntax**

```
HRESULT put_PortCLogic ( tagNARearPanelIOLogic Mode );
HRESULT get_PortCLogic ( tagNARearPanelIOLogic* Mode );
```

**Interface** IHWAuxIO

## PortCMode Property

---

**Description** Sets and reads whether Port C is setup for writing or reading data on the AUX IO connector and the Handler IO connector.

**Note:** Port C lines are connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *AuxIO.PortCMode* = *value*

**Variable** [\(Type\)](#) - [Description](#)

*AuxIO* **(object)** - A Hardware Aux I/O object

*value* **(enum as NaPortMode)** - Choose from:

**0 - naInput** - set the port for reading

**1 - naOutput** - set the port for writing

**Return Type** Enum as NaPortMode

**Default** 1 - naInput

**Examples**

```
auxIo.get_PortCMode = naInput 'Write
value = auxIo.get_PortCMode 'Read
```

**C++ Syntax** HRESULT get\_PortCMode( tagNAPortMode\* pMode );  
 HRESULT put\_PortCMode( tagNAPortMode pMode );

**Interface** IHWAuxIO

## PortCoupleToSystemMedia Property

---

**Description** Sets and returns the state of coupling with the system Media type.

**VB Syntax** *fixture.PortCoupleToSystemMedia (port)= value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) Object

*port* (Integer) Port Number that will receive the coupling change.

*value* (Boolean) Coupling state. Choose from:

- **True** - Media type is coupled with the system setting.
- **False** - Media type is NOT coupled with the system setting.

**Return Type** Boolean

**Default** True

**Examples** `fixture.PortCoupleToSystemMedia(2) = True`

`bool = fixture.PortCoupleToSystemMedia(1)`

**C++ Syntax** HRESULT get\_PortCoupleToSystemMedia(short portNum, VARIANT\_BOOL \*pVal);  
HRESULT put\_PortCoupleToSystemMedia(short portNum, VARIANT\_BOOL newVal);

**Interface** IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic

## PortCoupleToSystemVelocity Property

---

**Description** Sets and returns the state of coupling with the system Velocity Factor.

**VB Syntax** *fixture.PortCoupleToSystemVelocity (port)= value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) Object

*port* (Integer) Port Number that will receive the coupling change.

*value* (Boolean) Coupling state. Choose from:

- **True** - Velocity Factor is coupled with the system setting.
- **False** - Velocity Factor is NOT coupled with the system setting.

**Return Type** Boolean

**Default** True

**Examples** `fixture.PortCoupleToSystemVelocity(2) = True`

`bool = fixture.PortCoupleToSystemVelocity(1)`

**C++ Syntax** HRESULT get\_PortCoupleToSystemVelocity(short portNum, VARIANT\_BOOL \*pVal);  
HRESULT put\_PortCoupleToSystemVelocity(short portNum, VARIANT\_BOOL newVal);

**Interface** IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic

## PortDelay Property

---

**Description** Sets and returns the Port Extensions Delay value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

This command replaces [Port 1](#) [Port 2](#) [Port 3](#) Properties.

**VB Syntax** *fixture.PortDelay(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive delay value.

*value* (Double) Delay value in seconds. Choose a value between -1E18 and 1E18.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortDelay(2) = .002 'Write`

`value = fixture.PortDelay(1) 'Read`

**C++ Syntax** HRESULT get\_PortDelay(short port double \*pVal)  
HRESULT put\_PortDelay(short port double newVal)

**Interface** IFixturing

## PortDescription Property

---

**Description** For each port of the ECal module that is going to be characterized, sets and reads the description of the adapters, cable, or fixture to be included in the user characterization. This description is stored with the characterization in the ECal module.

Set this description before sending [Initialize](#) or the default (empty string) will be used.

**VB Syntax** `userChar.PortDescription (port)= value`

**Variable** [\(Type\)](#) - Description

*userChar* An [ECalUserCharacterizer](#) Object

*port* (Enum) ECal port for which description is to be set. Choose from:

1 or **naECalPort\_A**

2 or **naECalPort\_B**

3 or **naECalPort\_C**

4 or **naECalPort\_D**

*value* (String) Descriptive text, limited to 24 characters maximum.

**Return Type** String

**Default** "" (Empty String)

**Examples** `userChar.PortDescription (naECalPort_C)= "3.5 mm adapter, SN 00001"`

**C++ Syntax** HRESULT get\_PortDescription(NAECalPort port, \*BSTR description);  
HRESULT put\_PortDescription(NAECalPort port, BSTR description);

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## PortDistance Property

---

**Description** Sets and returns the port extension delay in physical length (distance).

**VB Syntax** *fixture.PortDistance (port)= value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) Object

*port* (Integer) Port Number that will receive the delay setting.

*value* (Double) Physical length in distance. First specify units with [PortDistanceUnit](#).

**Return Type** Double

**Default** 0

**Examples**

```
fixture.PortDistance(2) = .01 'Write
value = fixture.PortDistance(2) 'Read
```

**[C++ Syntax](#)** HRESULT get\_PortDistance(short portNum, double \*pVal);  
HRESULT put\_PortDistance(short portNum, double newVal);

**Interface** IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic

## PortDistanceUnit Property

---

**Description** Sets and returns the units for specifying port extension delay in physical length (distance).

**VB Syntax** *fixture*.**PortDistanceUnit** = *value*

**Variable**    [\(Type\)](#) - Description

*fixture*    A [Fixturing](#) Object

*value*    (enum naDistanceUnit) Choose from:

0 – naDistanceUnit\_Meter

1 – naDistanceUnit\_Feet

2 – naDistanceUnit\_Inch

**Return Type**    Enum

**Default**    naDistanceUnit\_Meter

**Examples**    `fixture.PortDistanceUnit(2) = naDistanceUnit_Meter 'Write`  
                  `value = fixture.PortDistanceUnit(2) 'Read`

**C++ Syntax**    HRESULT get\_PortDistanceUnit(short portNum, tagNADistanceUnit\* \*pVal);  
                  HRESULT put\_PortDistanceUnit(short portNum, tagNADistanceUnit newVal);

**Interface**    IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic



## PortExtState Property

---

**Description** Turns Port Extension ON or OFF for all ports on the channel.

**VB Syntax** *fixture.PortExtState = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Boolean)

**False** - Turns Port Extensions OFF

**True** - Turns Port Extensions ON

**Return Type** Boolean

**Default** False

**Examples** `fixture.PortExtState = 0 'Write`

`value = fixture.PortExtState 'Read`

**C++ Syntax** HRESULT get\_PortExtState(VARIANT\_BOOL \*pVal)  
HRESULT put\_PortExtState(VARIANT\_BOOL newVal)

**Interface** IFixturing

## PortExtUse1 Property

---

**Description** Sets and returns the Use1 ON/OFF state for the use of the [PortLoss1](#) and [PortFreq1](#) values for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortExtUse1(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive Use1 ON / OFF state.

*value* **(Boolean)**

False - Turns Use1 OFF

True - Turns Use1 ON

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.PortExtUse1(2) = False 'Write`

`value = fixture.PortExtUse1(1) 'Read`

**C++ Syntax** HRESULT get\_PortExtUse1(short port VARIANT\_BOOL \*pVal)  
HRESULT put\_PortExtUse1(short port VARIANT\_BOOL newVal)

**Interface** IFixturing

## PortExtUse2 Property

---

**Description** Sets and returns the Use2 ON/OFF state for the use of the [PortLoss2](#) and [PortFreq2](#) values for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortExtUse2(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive Use2 ON / OFF state.

*value* **(Boolean)**

False - Turns Use1 OFF

True - Turns Use1 ON

**Return  
Type** Boolean

**Default** False

**Examples** `fixture.PortExtUse2(2) = False 'Write`

`value = fixture.PortExtUse2(1) 'Read`

**C++ Syntax** HRESULT get\_PortExtUse2(short port VARIANT\_BOOL \*pVal)  
HRESULT put\_PortExtUse2(short port VARIANT\_BOOL newVal)

**Interface** IFixturing

## PortFreq1 Property

---

**Description** Sets and returns Frequency1 value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortFreq1(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive extrapolated loss.

*value* (Double) Frequency1 value. Choose a frequency within the frequency span of the PNA.

**Return Type** Double

**Default** 1 GHz

**Examples** `fixture.PortFreq1(2) = naFix2PD_USER 'Write`

`value = fixture.PortFreq1(1) 'Read`

**C++ Syntax** HRESULT get\_PortFreq1(short port double \*pVal)  
HRESULT put\_PortFreq1(short port double newVal)

**Interface** IFixturing

## PortFreq2 Property

---

**Description** Sets and returns Frequency2 value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortFreq2(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive extrapolated loss.

*value* (Double) Frequency2 value. Choose a frequency within the frequency span of the PNA.

**Return Type** Double

**Default** 1 GHz

**Examples** `fixture.PortFreq2(2) = 10E9 'Write`

`value = fixture.PortFreq2(1) 'Read`

**C++ Syntax** HRESULT get\_PortFreq2(short port double \*pVal)  
HRESULT put\_PortFreq2(short port double newVal)

**Interface** IFixturing

## PortLabel Property

---

**Description** Sets and returns the label on the calibration kit Port for the calibration wizard.

**VB Syntax** *calKit.PortLabel (portNum) = value*

**Variable** [\(Type\)](#) - Description

*calKit* A CalKit (**object**)

*portNum* (**long integer**) - number of the port to be labeled. Choose either **1** or **2**

*value* (**string**) - Label that is visible in the calibration wizard.

**Return Type** String

**Default** Depends on the Cal Kit.

**Examples** `calKit.PortLabel = "MyCalKit" 'Write`

`kitLabel = calKit.PortLabel 'Read`

**C++ Syntax** HRESULT get\_PortLabel(long port, BSTR \*pVal)  
HRESULT put\_PortLabel(long port, BSTR newVal)

**Interface** ICalKit

## PortLogic Property

---

<b>Description</b>	Sets and returns the logic mode of data ports A-H on the HandlerIO connector. Port C of the Handler IO is connected internally to the Port C of the Aux IO connector. Therefore, it will have the same logic mode.
<b>VB Syntax</b>	<i>handler</i> . <b>PortLogic</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>handler</i>	<b>(object)</b> - A HandlerI/O object
<i>value</i>	<p><b>(enum as NaRearPanelIOLogic)</b> - Choose from:</p> <p><b>0 - naPositiveLogic</b> - When a value of one is written, the associated line goes High</p> <p><b>1 - naNegativeLogic</b> - When a value of one is written, the associated line goes Low</p> <p>For ports that are in output (write) mode, a change in logic causes the output lines to change state immediately. For example, Low levels change immediately to High levels.</p> <p>For ports that are in input (read) mode (C,D,E only), a change in logic will be reflected when data is read from that port. For example, if a line read 0, the next read after a logic change will read 1.</p>
<b>Return Type</b>	Long Integer
<b>Default</b>	1 - naNegativeLogic
<b>Examples</b>	<pre>handler.PortLogic = value 'Write value = handler.PortLogic 'Read</pre>
<b><a href="#">C++ Syntax</a></b>	<pre>HRESULT put_PortLogic( tagNARearPanelIOLogic Mode ); HRESULT get_PortLogic( tagNARearPanelIOLogic* Mode );</pre>
<b>Interface</b>	IHWMaterialHandlerIO

## PortLoss1 Property

---

**Description** Sets and returns the Loss1 value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortLoss1(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive Loss value

*value* **(Double)** Loss1 value in dB. Choose a value between -90 and 90.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortLoss1(2) = .002 'Write`

`value = fixture.PortLoss1(1) 'Read`

**C++ Syntax** HRESULT get\_PortLoss1(short port double \*pVal)  
HRESULT put\_PortLoss1(short port double newVal)

**Interface** IFixturing



## PortLoss2 Property

---

**Description** Sets and returns the Loss2 value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortLoss2(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive Loss value

*value* **(Double)** Loss2 value in dB. Choose a value between -90 and 90.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortLoss2(2) = .002 'Write`

`value = fixture.PortLoss2(1) 'Read`

**C++ Syntax** HRESULT get\_PortLoss2(short port double \*pVal)  
HRESULT put\_PortLoss2(short port double newVal)

**Interface** IFixturing

## PortLossDC Property

---

**Description** Sets and returns the Loss value at DC for the specified port number.

[Learn about Loss compensation values.](#)

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortLossDC(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive Loss value at DC.

*value* (Double) Loss value in ohms. Choose a value between -90 and 90

**Return Type** Double

**Default** 0

**Examples** `fixture.PortLossDC(2) = .002 'Write`

`value = fixture.PortLossDC(1) 'Read`

**C++ Syntax** HRESULT get\_PortLossDC(short port double \*pVal)  
HRESULT put\_PortLossDC(short port double newVal)

**Interface** IFixturing

## PortMatching\_C Property

---

**Description** Sets and returns the Capacitance value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortMatching\_C(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive capacitance value

*value* **(Double)** Capacitance value in farads. Choose a value between -1E18 to 1E18.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortMatching_C(2) = .00002 'Write`

`value = fixture.PortMatching_C(1) 'Read`

**C++ Syntax** HRESULT get\_PortMatching\_C(short port double \*pVal)  
HRESULT put\_PortMatching\_C(short port double newVal)

**Interface** IFixturing

## PortMatching\_G Property

---

**Description** Sets and returns the Conductance value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortMatching\_G(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive conductance value.

*value* **(Double)** Conductance value in siemens. Choose a value between -1E18 and 1E18.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortMatching_G = .002 'Write`

`value = fixture.PortMatching_G 'Read`

**C++ Syntax** HRESULT get\_PortMatching\_G(short port double \*pVal)  
HRESULT put\_PortMatching\_G(short port double newVal)

**Interface** IFixturing

## PortMatching\_L Property

---

**Description** Sets and returns the Inductance value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortMatching\_L(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive inductance value

*value* **(Double)** Inductance value in henries. Choose a value between -1E18 and 1E18.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortMatching_L = .002 'Write`

`value = fixture.PortMatching_L 'Read`

**C++ Syntax** HRESULT get\_PortMatching\_L(short port double \*pVal)  
HRESULT put\_PortMatching\_L(short port double newVal)

**Interface** IFixturing

## PortMatching\_R Property

---

**Description** Sets and returns the Resistance value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortMatching\_R(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive resistance value.

*value* **(Double)** Resistance value in ohms. Choose a value between -1E18 and 1E18.

**Return Type** Double

**Default** 0

**Examples** `fixture.PortMatching_R = .1 'Write`

`value = fixture.PortMatching_R 'Read`

**C++ Syntax** HRESULT get\_PortMatching\_R(short port double \*pVal)  
HRESULT put\_PortMatching\_R(short port double newVal)

**Interface** IFixturing

## PortMatchingCktModel Property

---

**Description** Sets and returns the Port Matching circuit model for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.PortMatchingCktModel(port) = value*

**Variable** (Type) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive circuit model.

*value* (Enum as **NAFixturingPortMatchCkt**) Circuit model. Choose from

0 **naFixPMC\_SLPC** Series L - Parallel C

1 **naFixPMC\_PCSL** Parallel C - Series L

2 **naFixPMC\_PLSC** Parallel L - Series C

3 **naFixPMC\_SCPL** Series C - Parallel L

4 **naFixPMC\_PLPC** Parallel L - Parallel C

5 **naFixPMC\_USER** Load S2P file - also set filename to load with [strPortMatch\\_S2PFile Property](#)

6 **naFixPMC\_NONE** No circuit model

**Return Type** Long Integer

**Default** **naFixPMC\_NONE**

**Examples** `fixture.PortMatchingCktModel(2) = naFixPMC_PLSC 'Write`

`value = fixture.PortMatchingCktModel(1) 'Read`

**C++ Syntax** HRESULT get\_PortMatchingCktModel(short port tagNAFixturingPortMatchCkt \*pVal)  
HRESULT put\_PortMatchingCktModel(short port tagNAFixturingPortMatchCkt newVal)

**Interface** IFixturing

## PortMatchingState Property

---

**Description** Sets and returns the Port Matching State on the channel.

**VB Syntax** *fixture*.PortMatchingState = *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (boolean)

**True** - Turns Port Matching ON

**False** - Turns Port Matching OFF

**Return Type** Boolean

**Default** False

**Examples** `fixture.PortMatchingState = True 'Write`

`value = fixture.PortMatchingState 'Read`

**C++ Syntax** HRESULT get\_PortMatchingState(VARIANT\_BOOL \*pVal)  
HRESULT put\_PortMatchingState(VARIANT\_BOOL newVal)

**Interface** IFixturing



## PortMedium Property

---

**Description** Sets and returns the media type of the added fixture or transmission line.

**VB Syntax** *fixture*.**PortMedium** (*port*)= *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) Object

*port* (Integer) Port Number for which media type is being set.

*value* (enum NACalStandardMedium) - Medium of the transmission line of the standard.  
Choose from:

**0 - naCoax** - Coaxial Cable

**1 - naWaveGuide** - Waveguide

**Return Type** Enum

**Default** **0 - naCoax**

**Examples** `fixture.PortMedium(2)= naCoax 'Write`

`value = fixture.PortMedium(2) 'Read`

**C++ Syntax** HRESULT get\_PortMedium(short portNum, tagNACalStandardMedium\* pVal);  
HRESULT put\_PortMedium(short portNum, tagNACalStandardMedium newVal);

**Interface** IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic

## PortMode Property

---

<b>Description</b>	Sets and returns whether Port C or Port D is used for writing or reading data on the Handler IO connector. The Handler IO Port C is connected internally to the Port C of the Aux IO connector. Therefore, the Aux IO connector will have the same input/output mode.
<b>VB Syntax</b>	<i>handler</i> . <b>PortMode</b> ( <i>port</i> ) = <i>value</i>
<b>Variable</b>	<a href="#">(Type) - Description</a>
<i>handler</i>	<b>(object)</b> - A Handler I/O object
<i>port</i>	<b>(enum as NAMatHandlerPort)</b> Port to be changed. Choose from: <b>2 -naPortC</b> <b>3- naPortD</b>
<i>value</i>	<b>(enum as NaPortMode)</b> - Choose from: <b>0 - naInput</b> - set the port for reading <b>1 - naOutput</b> - set the port for writing
<b>Return Type</b>	Long Integer
<b>Default</b>	1 - naInput
<b>Examples</b>	<pre>handler.PortMode(naPortC) = naInput 'Write value = handler.PortMode(naPortD) 'Read</pre>
<b>C++ Syntax</b>	HRESULT put_PortMode ( tagNAMatHandlerPort Port, tagNAPortMode Mode ); HRESULT get_PortMode ( tagNAMatHandlerPort Port, tagNAPortMode* Mode );
<b>Interface</b>	IHWMaterialHandlerIO

## PortsNeedingDeltaMatch Property

---

**Description** Returns the port numbers for which delta match correction is required. 0 (zero) is returned if the Cal does NOT require Delta Match correction for one of the following reasons:

- The Cal does NOT involve Unknown Thru or TRL. You specify this using [ThruCalMethod Property](#).
- The Cal DOES involve Unknown Thru or TRL, but the delta match data can be calculated by the Unknown Thru or TRL Cal. [Learn how this is possible](#). However, you can force the Cal to use the Delta Match data from a Cal Set.

**VB Syntax** *value* = *guided*.PortsNeedingDeltaMatch

**Variable** [\(Type\)](#) - **Description**

*value* (Variant) Variable to store the returned list of port numbers.

*guided* [GuidedCalibration](#) (object)

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
Dim ports As Variant
ports = guided.PortsNeedingDeltaMatch
```

**C++ Syntax** HRESULT get\_PortsNeedingDeltaMatch (VARIANT\* portList);

**Interface** IGuidedCalibration2

## PortVelocityFactor Property

---

**Description** Sets and returns the Port Extensions Velocity Factor value for the specified port number.

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture*.PortVelocityFactor(*port*) = *value*

**Variable** (Type) - Description

*fixture* A [Fixturing](#) (object)

*port* (Integer) Port number to receive velocity factor value.

*value* (Double) Velocity Factor value. Choose a number between: 0 and 10  
(.66 polyethylene dielectric; .7 teflon dielectric)

**Return Type** Double

**Default** 1

**Examples** `fixture.PortVelocityFactor(2) = .6 'Write`

`value = fixture.PortVelocityFactor(1) 'Read`

**C++ Syntax** HRESULT get\_PortVelocityFactor(short port double \*pVal)  
HRESULT put\_PortVelocityFactor(short port double newVal)

**Interface** IFixturing4

---

Last Modified:

12-Feb-2009    MX New topic

## PortWGCutoffFreqProperty

---

**Description** Sets and returns the cutoff (minimum) frequency of the added waveguide fixture or transmission line.

**VB Syntax** *fixture*.PortWGCutoffFreq(*port*)= *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) Object

*port* (Integer) Port Number for which media type is being set.

*value* (Double) Cutoff frequency in Hz.  
This value is ignored when [PortMedium Property](#) is set to **COAX** for the same port.

**Return Type** Double

**Default** System Media Cutoff Frequency

**Examples** `fixture.PortWGCutoffFreq(2)= 1e9 'Write`  
`value = fixture.PortWGCutoffFreq(2) 'Read`

**C++ Syntax** HRESULT get\_PortWGCutoffFreq(short portNum, double \*pVal);  
HRESULT put\_PortWGCutoffFreq(short portNum, double newVal);

**Interface** IFixturing4

---

Last Modified:

6-Feb-2009    MX New topic

## PowerSlope Property

---

**Description** Sets or returns the Power Slope value. Power Slope function increases or decreases the output power over frequency. Units are db/GHz. For example: PowerSlope = 2 will increase the power 2db/1GHZ.

**VB Syntax** *object.PowerSlope = value*

**Variable** [\(Type\)](#) - Description

*object* [Channel](#) (**object**)

**or**

[CalSet](#) (**object**) - Read-only property

*value* (**double**) - Power Slope. Choose any number between -2 and 2.  
**No slope = 0**

**Return  
Type** Double

**Default** 0

**Examples** `chan.PowerSlope = 2 'Write`

`pwrslp = chan.PowerSlope 'Read`

**C++ Syntax** HRESULT get\_PowerSlope(double \*pVal)  
HRESULT put\_PowerSlope(double newVal)

**Interface** IChannel  
|CalSet3

## PowerAcquisitionDevice Property

---

<b>Description</b>	<p>Returns the power sensor channel (A or B) that is currently selected for use at a specific frequency.</p> <p>If <a href="#">UsePowerSensorFrequencyLimits</a> is set to False, this property will return the sensor channel last used for a source power calibration. This setting corresponds to the <b>Use this sensor only</b> checkbox in the <a href="#">Power Sensor Settings</a> dialog.</p> <p>When performing an SMC calibration, use <a href="#">SetPowerAcquisitionDevice Method</a> to set the power sensor channel.</p>
<b>VB Syntax</b>	<i>sensor</i> = <i>pwrCal</i> . <b>PowerAcquisitionDevice</b> ( <i>dFreq</i> )
<b>Variable</b>	<b>(Type) - Description</b>
<i>sensor</i>	( <b>enum NPowerAcquisitionDevice</b> ) The currently selected sensor channel for the specified frequency. Choose from: 0 – naPowerSensor_A 1 – naPowerSensor_B
<i>pwrCal</i>	A SourcePowerCalibrator (object)
<i>dFreq</i>	(double) Frequency (Hz) for the power reading of interest.
<b>Return Type</b>	enum NPowerAcquisitionDevice
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>selectedSensor = pwrCal.PowerAcquisitionDevice(1.E9) 'Write</pre>
<b>C++ Syntax</b>	HRESULT get_PowerAcquisitionDevice(double dFreq, tagNPowerAcquisitionDevice* enumAcqDevice);
<b>Interface</b>	ISourcePowerCalibrator2

## PowerAsFixture Property

---

<b>Description</b>	Sets and reads the state of power offset as a fixture with True Mode balanced measurements. <a href="#">Learn more about iTMSA power and power offset.</a>
<b>VB Syntax</b>	<i>balStim</i> . <b>PowerAsFixture</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>balStim</i>	A <a href="#">BalancedStimulus</a> (object)
<i>value</i>	<b>(Boolean)</b> State of power offset as a fixture. <b>False</b> Offset is applied but is NOT included as a fixture in the output calculations. <b>True</b> Offset is applied and included as a fixture in the output calculations.
<b>Return Type</b>	Boolean
<b>Default</b>	False
<b>Examples</b>	<pre>balStim.PowerAsFixture = False 'Write</pre> <pre>var = balStim.PowerAsFixture 'Read</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_PowerAsFixture (VARIANT_BOOL *bVal) HRESULT put_PowerAsFixture (VARIANT_BOOL bVal)
<b>Interface</b>	IBalancedStimulus

---

Last Modified:

27-May-2008    MX New topic



## PowerLevel Property

---

**Description** Set and read the power level at which to perform the Source Power Cal portion of a Gain Compression Calibration or a SweptIMD Cal.

**VB Syntax** *object*.**PowerLevel** = *value*

**Variable**    [\(Type\)](#) - Description

*object*    A [GainCompressionCal](#) (object)

          A [SweptIMDCal](#) (object)

*value*    **(Double)** - Power level in dB. Choose a value from +30 to (-30). [Learn about choosing a power level.](#)

**Return  
Type**    Double

**Default**    0

**Examples**    `gca.PowerLevel = -5 'Write`

`pLevel = imd.PowerLevel 'Read`

**C++ Syntax**    HRESULT get\_PowerLevel(double\* pVal)  
                  HRESULT put\_PowerLevel(double newVal)

**Interface**    IGainCompressionCal  
                  ISweptIMDCal

---

Last Modified:

9-Sep-2008    Added IMD

11-Apr-2008    MX New topic

## PowerMax Property

---

**Description** Sets and returns the maximum power level for Safe Mode.

**VB Syntax** *RxLevel.PowerMax(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the max power for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Double) Maximum power level in dB.

**Return Type** (Double)

**Default** + 30

**Examples** `rxLevel.PowerMax (1) = 25 ' Write`

`value = rxLevel.PowerMax 2 ' Read`

**C++ Syntax** HRESULT get\_PowerMax(long port, double\* pVal);  
HRESULT put\_PowerMax(long port, double newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## PowerMeterChannel Property

---

**Description** Identifies which channel of the power meter the power sensor is connected to.

**VB Syntax** *chan* = *powerSensor*.**PowerMeterChannel**

**Variable** [\(Type\)](#) - **Description**

*chan* (**enum NAPowerAcquisitionDevice**) – Power meter channel identifier for sensor.  
Choose from:

**0 – naPowerSensor\_A**

**1 – naPowerSensor\_B**

*pwrSensor* A [PowerSensor](#) (Object) **or**  
A [PowerSensorAsReceiver](#) (Object)

**Return Type** NAPowerAcquisitionDevice

**Default** Not Applicable

**Examples** `meterChannel = pwrSensor.PowerMeterChannel`

**C++ Syntax** HRESULT PowerMeterChannel(tagNAPowerAcquisitionDevice \*pSensor);

**Interface** IPowerSensor  
IPowerSensorAsReceiver

---

Last Modified:

25-Aug-2009    Added PMAR

## PowerMeterGPIBAddress Property **Superseded**

---

**Description** This command is replaced with [PowerMeterInterface Object](#).  
Specifies the GPIB address of the power meter that will be referenced by the SourcePowerCalibrator object.  
When performing a source power cal, the PNA will search VISA interfaces that are configured in the Agilent IO Libraries on the PNA.

**VB Syntax** *powerCalibrator.PowerMeterGPIBAddress* = *value*

**Variable** [\(Type\)](#) - **Description**

*powerCalibrator* **(object)** - A SourcePowerCalibrator (object)

*value* **(long integer)** – Power meter GPIB address. Choose any number between 0 and 30.

**Return Type** Long integer

**Default** 13

**Examples**

```
Set powerCalibrator = pna.SourcePowerCalibrator
powerCalibrator.PowerMeterGPIBAddress = 13 'Write

pwrMtrAddress = powerCalibrator.PowerMeterGPIBAddress 'Read
```

**C++ Syntax** HRESULT put\_PowerMeterGPIBAddress(long newVal);  
HRESULT get\_PowerMeterGPIBAddress(long \*pVal);

**Interface** ISourcePowerCalibrator

---

Last Modified:

9-Jul-2007    Superseded

## PowerMin Property

---

**Description** Sets and returns the minimum power level for Safe Mode.

**VB Syntax** *RxLevel.PowerMin(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the min power for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Double) Minimum power level in dB.

**Return Type** (Double)

**Default** -95 dB

**Examples** `rxLevel.PowerMin (1) = -50 ' Write`

`value = rxLevel.PowerMin 2 ' Read`

**C++ Syntax** HRESULT get\_PowerMin(long port, double\* pVal);  
HRESULT put\_PowerMin(long port, double newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## PowerOffset Property

---

**Description** Sets and returns the power level offset value.

**VB Syntax** *RxLevel.PowerOffset(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the offset power for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Double) Power level Offset in dB.

**Return Type** (Double)

**Default** 0 dB

**Examples** `rxLevel.PowerOffset (1) = 10 ' Write`

`value = rxLevel.PowerOffset 2 ' Read`

**C++ Syntax** HRESULT get\_PowerOffset(long port, double\* pVal);  
HRESULT put\_PowerOffset(long port, double newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## PowerOnDuringRetraceMode Property

---

**Description** For single-band frequency or segment sweeps ONLY, specify whether to turn RF power ON or OFF during a retrace.

This setting remains until changed using this command, or until the hard drive is changed or reformatted.

**VB Syntax** *pref.PowerOnDuringRetraceMode = value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* **(Enum)** - Choose from:

0 - **naRetracePowerMode\_Auto** Power is left ON during retrace of single-band frequency or segment sweeps ONLY.

1 - **naRetracePowerMode\_OFF** Power is turned OFF during retrace of single-band frequency or segment sweeps ONLY.

**Return Type** Enum

**Default** 0 - naRetracePowerMode\_Auto

**Examples**

```
pref.PowerOnDuringRetraceMode = naPowerSweepRetraceMode_Start 'Write  
psMode = pref.naPowerOnDuringRetraceMode_Start 'Read
```

**C++ Syntax** HRESULT get\_PowerOnDuringRetraceMode (tagNARetracePowerMode\* preference);  
HRESULT put\_PowerOnDuringRetraceMode (tagNARetracePowerMode val)

**Interface** IPreferences4

---

Last modified:

Nov. 16, 2006      New command

## PowerSensorCalKitType Property

---

**Description** Set and read the cal kit to be used for calibrating at the port 1 reference plane when the power sensor connector is different from the DUT port 1. [Learn more.](#)

**VB Syntax** *object*.PowerSensorCalKitType = *value*

**Variable** [\(Type\)](#) - Description

*object* A [GainCompressionCal](#) (object)

A [SweptIMDCal](#) (object)

*value* (String) - Cal Kit. Use [CompatibleCalKits Property](#) to return a list of valid cal kits.

**Return Type** String

**Default** Not Applicable

**Examples** `gca.PowerSensorCalKitType = "85052B" 'Write`

`ctype = imd.PowerSensorCalKitType 'Read`

**C++ Syntax** HRESULT get\_PowerSensorCalKitType(BSTR\* Val)  
HRESULT put\_PowerSensorCalKitType(BSTR newVal)

**Interface** IGainCompressionCal2  
ISweptIMD

---

Last Modified:

9-Sep-2008 Added Swept IMD

11-Apr-2008 MX New topic



## PowerSensorConnectorType Property

---

<b>Description</b>	Set and read the power sensor connector type which is used to perform the Source Power Cal portion of a Gain Compression Calibration or a SweptIMD Cal.
<b>VB Syntax</b>	<i>object</i> . <b>PowerSensorConnectorType</b> = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>object</i> .	A <a href="#">GainCompressionCal</a> ( <b>object</b> ) A <a href="#">SweptIMDCal</a> ( <b>object</b> )
<i>value</i>	<b>(String)</b> - Power sensor connector type. Use <a href="#">ValidConnectorType Property</a> to return a list of valid connector types. Select "Ignored" to NOT compensate for the adapter.
<b>Return Type</b>	String
<b>Default</b>	Not applicable.
<b>Examples</b>	<pre>gca.PowerSensorConnectorType = "APC3.5 male" 'Write</pre> <pre>ctype = imd.PowerSensorConnectorType 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_PowerSensorConnectorType(BSTR* Val) HRESULT put_PowerSensorConnectorType(BSTR newVal)
<b>Interface</b>	IGainCompressionCal2 ISweptIMDCal

---

Last Modified:

3-Jun-2010    MX New topic

## PowerStep Property

---

**Description** Sets and returns the safe mode power step value.

**VB Syntax** *RxLevel.PowerStep(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the power step for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Double) Power step in dB.

**Return Type** (Double)

**Default** 1 dB

**Examples** `rxLevel.PowerStep (1) = 2 ' Write`

`value = rxLevel.PowerStep 2 ' Read`

**C++ Syntax** HRESULT get\_PowerStep(long port, double\* pVal);  
HRESULT put\_PowerStep(long port, double newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## PowerSweepRetracePowerMode Property

---

**Description** At the end of a power sweep, while waiting to trigger the next sweep, maintain source power at either the start power level or at the stop power level.

**VB Syntax** *pref*.PowerSweepRetracePowerMode = *value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* (Enum) - Choose from:

0 - **naPowerSweepRetraceMode\_Start** - maintain source at start power level.

1 - **naPowerSweepRetraceMode\_Stop** - maintain source at stop power level.

**Return Type** Enum

**Default** 0 - naPowerSweepRetraceMode\_Start

**Examples** `pref.PowerSweepRetracePowerMode = naPowerSweepRetraceMode_Start 'Write`

`psMode = pref.naPowerSweepRetraceMode_Start 'Read`

**C++ Syntax** HRESULT get\_PowerSweepRetracePowerMode (tagNAPowerSweepRetraceMode\* preference);  
HRESULT put\_PowerSweepRetracePowerMode (tagNAPowerSweepRetraceMode val)

**Interface** IPreferences3

---

Last modified:

Oct. 25, 2006    New command

## PreciseTuningTolerance Property

---

**Description** Sets and returns the tuning tolerance for precise tuning.

**VB Syntax** *obj.PreciseTuningTolerance = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (**object**) or  
A [ConverterEmbeddedLO](#) (**object**)

*value* (**Double**) Tuning tolerance in Hz.

**Return Type** (**Double**)

**Default** 1 Hz

**Examples** `embedLO.PreciseTuningTolerance = .5 'write`

`value = embedLO.PreciseTuningTolerance 'read`

**C++ Syntax** HRESULT get\_PreciseTuningTolerance(double\* tolerance);  
HRESULT put\_PreciseTuningTolerance(double tolerance);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## PreferInternalTriggerOnChannelSingle Property

---

**Description** Set and read the preference for the [chan.Single](#) trigger behavior. This setting persists until changed.

These preferences are important when performing a Guided calibration, as the PNA uses the **chan.Single** trigger command to measure standards.

- set `PreferInternalTriggerOnChannelSingle = False` to use an External trigger sweep to measure a cal standard.
- set `PreferInternalTriggerOnChannelSingle = True` to use an External sweep for the measurement, but rely on the PNA to send Internal trigger signals for calibrating.

To set this preference for an **Unguided** Calibration, use [PreferInternalTriggerOnUnguidedCal Property](#)

The **chan.Single** trigger command NEVER respects the Trigger Source = Manual setting. It always switches to Internal for one trigger, then back to Manual, regardless of this preference command.

**VB Syntax** `pref.PreferInternalTriggerOnChannelSingle = bool`

### Variable (Type) - Description

*pref* A [Preferences](#) (object)

*bool* (**Boolean**) - Choose from:

**0 - False** - the Single trigger property does respect the **Trigger Source = External** setting. For example, if [Trigger source = External](#), the single trigger method will wait for the External trigger signal and then allow only one sweep.

**1 - True** - the Single trigger command does NOT respect the **Trigger Source = External** setting. For example, when the Single method is sent, the PNA immediately switches to Internal sweep, responds to one trigger signal, then switches back to External.

**Return Type** Boolean

**Default** 0 - False

**Examples** `pref.PreferInternalTriggerOnChannelSingle = False 'Write`

`prefer = pref.PreferInternalTriggerOnChannelSingle 'Read`

**C++ Syntax** `HRESULT put_PreferInternalTriggerOnChannelSingle( VARIANT_BOOL bprefSingle)`  
`HRESULT get_PreferInternalTriggerOnChannelSingle( VARIANT_BOOL *bprefSingle)`

**Interface** IPreferences2



## PreferInternalTriggerOnUnguidedCal Property

---

**Description** Set and read the preference for the trigger behavior when performing an Unguided calibration.

**VB Syntax** *pref.PreferInternalTriggerOnUnguidedCal = bool*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*bool* **(Boolean)** - Choose from:

**0 - False** - The trigger behavior during an Unguided calibration DOES respect the setting of the [Trigger source](#) command. For example, during an Unguided Cal, when Trigger source = External, the PNA will wait for the External trigger signal and then allow only one sweep.

**1 - True** -The trigger behavior during an Unguided calibration does NOT respect the **Trigger Source = External** setting. For example, during an Unguided Cal, when Trigger source = External, the PNA immediately switches to Internal sweep, measures the standard with one trigger signal, then switches back to External trigger.

**Note:** When Trigger Source = Manual during a calibration, the PNA ALWAYS switches to Internal for one trigger to measure a standard, then back to Manual, regardless of this preference command.

**Return Type** Boolean

**Default** 0 - False

**Examples** `pref.PreferInternalTriggerOnUnguidedCal = False 'Write`

`prefer = pref.PreferInternalTriggerOnUnguidedCal 'Read`

**C++ Syntax** HRESULT put\_PreferInternalTriggerOnUnguidedCal( VARIANT\_BOOL bprefUnguided)  
HRESULT get\_PreferInternalTriggerOnUnguidedCal( VARIANT\_BOOL \*bprefUnguided)

**Interface** IPreferences2

## RangeCount Property

---

**Description** Returns the number of ranges that are available in the PNA.  
To see the range names, query the [Name](#) property of each range in the FOM collection.

**VB Syntax** *value* = *FOM.RangeCount*

**Variable** [\(Type\)](#) - Description

*object* An [FOM](#) (collection object)

*value* **(long)** - Variable to store the returned number of ranges.

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `NumRanges = fom.RangeCount 'Read`

**C++ Syntax** HRESULT get\_RangeCount(long \*count)

**Interface** IFOM

---

Last Modified:

8-Mar-2007    Added link to Name property



## rangeNumber Property

---

**Description** Returns the index number of the range within the [FOM collection](#).

**VB Syntax** *value* = *FOMRange*.rangeNumber

**Variable** [\(Type\)](#) - Description

*value* **(Long)** - Variable to store the returned range number.

*object* An [FOMRange](#) (**object**)

**Return Type** Long

**Default** Not Applicable

**Examples** `num = fomRange.rangeNumber 'Read`

**C++ Syntax** HRESULT get\_rangeNumber(long \*pVal)

**Interface** IFOMRange

## R1InputPath Property

---

**Description** PNA models with option 081, and all PNA-X models, have a switch in the test set that allows access to the port 1 reference receiver through the front panel Reference 1 connectors. This command throws that switch between the internal path to the receiver, or through the external connectors.

See other [Frequency Offset](#) properties.

**VB Syntax** `chan.R1InputPath = value`

**Variable** [\(Type\)](#) - **Description**

*chan* A Channel (**object**)

*value* (**Enum as naInputPath**) - Choose from: naPathInternal -  
(0) - internal path to the reference receiver naPathExternal  
(1) - path through external connectors

**Return Type** Enum

**Default** naPathInternal - (0)

**Examples**

```
chan.R1InputPath = naPathInternal 'Write
```

```
Inpath = chan.R1InputPath 'Read
```

**C++ Syntax** HRESULT get\_R1InputPath (tag NAInputPath \*pPath);  
HRESULT put\_R1InputPath (tag NAInputPath newPath);

**Interface** IChannel2

---

Last Modified:

30-Apr-2009     Added PNA-X models

## ReadingsPerPoint Property

---

**Description** This command, along with [ReadingsTolerance](#), allows for settling of the power sensor READINGS.

Specifies the maximum number of power readings that are taken at each stimulus point to allow for power meter settling. Each reading is averaged with the previous readings at that stimulus point. When this average meets the [ReadingsTolerance](#) value or this number of readings has been made, the average is returned as the valid reading.

**VB Syntax** *pwrSensor*.**ReadingsPerPoint** = *value*

**Variable** [\(Type\)](#) - **Description**

*pwrCal* A [SourcePowerCalibrator](#) (Object) or  
A [PowerSensorAsReceiver](#) (Object)

*value* **(long integer)** – Number of power readings. Choose any number between 3 and 100.

**Return Type** Long Integer

**Default** 3

**Examples**

```
pwrSensor.ReadingsPerPoint = 3 'Write  
numReadings = pwrSensor.ReadingsPerPoint 'Read
```

**C++ Syntax** HRESULT put\_ReadingsPerPoint(long newVal);  
HRESULT get\_ReadingsPerPoint(long \*pVal);

**Interface** ISourcePowerCalibrator  
IPowerSensorAsReceiver

---

Last Modified:

25-Aug-2009 Added PMAR

17-Apr-2007 Clarified verbage

## ReadingsTolerance Property

---

**Description** This command, along with [ReadingsPerPoint Property](#) allows for settling of the power sensor READINGS.

Each power reading is averaged with the previous readings at each stimulus point. When the average meets this tolerance value or the maximum [ReadingsPerPoint](#) has been made, the average is returned as the valid reading.

**VB Syntax** *pwrSens.ReadingsTolerance = value*

**Variable (Type) - Description**

*pwrCal* A [SourcePowerCalibrator](#) (Object) or  
A [PowerSensorAsReceiver](#) (Object)

*value* **(Double)** – Power meter settling tolerance value in dB. Choose any number between 0 and 5.

**Return Type** Double

**Default** .05 dB

**Examples**

```
pwrSens.ReadingsTolerance = .1 'Write  
ReadTol = pwrSensor.ReadingsTolerance 'Read
```

**C++ Syntax** HRESULT get\_ReadingsTolerance( double \*pVal);  
HRESULT put\_ReadingsTolerance( double newVal);

**Interface** ISourcePowerCalibrator3  
IPowerSensorAsReceiver

---

Last Modified:

25-Aug-2009 Added PMAR

17-Apr-2007 Clarified verbage

## ReadyForTriggerPolarity Property

---

**Description** Specifies the polarity of Ready for Trigger output.

All existing Ready for Trigger outputs for PNA-X and PNA-L models are configured simultaneously with this command. [See Capabilities Summary.](#)

The Ready for Trigger polarity can NOT be configured for E836x models.

**VB Syntax** *trigsetup.ReadyForTriggerPolarity = value*

**Variable** [\(Type\)](#) - **Description**

*trigsetup* A [TriggerSetup](#) (object)

*value* (Enum as NAlevel)

Choose from:

**0** - naLow - Outputs a TTL low when the PNA is ready for trigger.

**1** - naHigh - Outputs a TTL high when the PNA is ready for trigger.

**Return  
Type** Enum

**Default** **0** - naLow

**Examples** `trigsetup.ReadyForTriggerPolarity = naLow 'Write`

`pol = trigsetup.ReadyForTriggerPolarity 'Read`

**C++ Syntax** HRESULT get\_ReadyForTriggerPolarity(tagNAlevel \*pVal);  
HRESULT put\_ReadyForTriggerPolarity(tagNAlevel newVal);

**Interface** ITriggerSetup3

---

Last Modified:

14-Mar-2008    MX New topic

## ReadyForTriggerState Property

---

**Description** Determines the control of Material Handler connector Pin 21.

**VB Syntax** *handler*.**ReadyForTriggerState** = *value*

**Variable** [\(Type\)](#) - Description

*handler* **(object)** - A [Handler I/O](#) object

*value* **(boolean)**

False - Pin 21 is controlled by Output Port B7

True - Pin 21 is controlled by the Ready for Trigger signal

**Return Type** Boolean

**Default** False

**Examples**

```
handler.ReadyForTriggerState = False 'Write  
bState = handler.ReadyForTriggerState 'Read
```

**C++ Syntax**

HRESULT put\_ReadyForTriggerState (BOOL \*pVal);

HRESULT get\_ReadyForTriggerState (BOOL newVal);

**Interface** IHWMaterialHandlerIO2

## ReceiverAttenuator Property

---

**Description** Sets or returns the value of the specified receiver attenuator control.

**VB Syntax** *object*.**ReceiverAttenuator**(*rec*) = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)

or

CalSet (**object**) - Read-only property

*rec* (**long integer**) - Receiver with attenuator control to be changed. Choose from any of the available receivers in your PNA

1 - Receiver A

2 - Receiver B

Receiver attenuation can not be set using [logical receiver notation](#).

*value* (**double**) - Attenuator value in dB. Choose any Long Integer between 0 and 35 in 5dB steps:

If an invalid value is entered, the analyzer will select the next lower valid value. For example, if 19.9 is entered the analyzer will select 15 dB attenuation.

**Return Type** Double

**Default** 0 db

**Examples** `chan.ReceiverAttenuator(1) = 5 'Write`

`attn = chan.ReceiverAttenuator(2) 'Read`

**C++ Syntax** HRESULT get\_ReceiverAttenuator(long lport, double \*pVal)  
HRESULT put\_ReceiverAttenuator(long lport, double newVal)

**Interface** IChannel

|CalSet3

## ReceiverCount Property

---

**Description** Returns the number of receivers in the remote PNA. The returned number includes both test port receivers and reference receivers. [See the number of reference receivers in your PNA.](#)

**VB Syntax** *value* = *cap*.ReceiverCount

**Variable** [\(Type\)](#) - Description

*value* (Long) - Variable to store the returned number of receivers.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.ReceiverCount 'Read`

**C++ Syntax** HRESULT get\_ReceiverCount(long \* receiverCount );

**Interface** ICapabilities



## ReceiverStepAttenuatorStepSize Property

---

**Description** Returns a value indicating the step size of the attenuator.

**VB Syntax** *value* = *cap*.ReceiverStepAttenuatorStepSize (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(Double)** - Variable to store the returned value of the attenuator step size.

*cap* A [Capabilities](#) **(object)**

*n* **(Long)** - port number to query for the value of the attenuator step size.

**Return Type** Double

**Default** Not Applicable

**Examples** `value = cap.ReceiverStepAttenuatorStepSize(1) 'Read`

**C++ Syntax** HRESULT get\_ReceiverStepAttenuatorStepSize(long portNumber, double \* stepSize );

**Interface** ICapabilities

Read-only

## ReceivePort Property

---

**Description** Returns the receiver (response) port number of measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**Note:** Returning a receiver port is only supported for S-Parameter measurements. If the measurement is not an S-Parameter, then E\_NA\_BAD\_PARAMETER is returned.

**VB Syntax** *value* = *meas*.**ReceivePort**

**Variable** [\(Type\)](#) - **Description**

*value* **(Long)** - Variable to store the returned value

*meas* A [Measurement](#) **(object)**

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `rp = meas.ReceivePort`

**C++ Syntax** HRESULT ReceivePort(Long\* rcvPort);

**Interface** IMeasurement2

## ReduceIFBandwidth Property

---

**Description** Sets or returns the state of the [Reduced IF Bandwidth at Low Frequencies](#) setting.

**VB Syntax** *chan.ReduceIFBandwidth = state*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*state* (boolean)

**False** - Turns Reduce IFBW **OFF**

**True** - Turns Reduce IFBW **ON**

**Return Type** Boolean

**Default** True

**Examples** `chan.ReduceIFBandwidth = False 'Write`

`reduce = chan.ReduceIFBandwidth 'Read`

**C++ Syntax** HRESULT get\_ReduceIFBandwidth(BOOL \*pVal)  
HRESULT put\_ReduceIFBandwidth(BOOL newVal)

**Interface** IChannel5

---

Last Modified:

16-Aug-2007    Corrected Interface number

## ReferenceCalFactor Property

---

**Description** Sets and returns the Cal Factor (%) for the 50 MHz reference signal associated with this power sensor. Use this property **only** if the power sensor does not contain cal factors in EPROM (for example, HP/Agilent 848x sensors).

**VB Syntax** *pwrSensor*.**ReferenceCalFactor** = *value*

**Variable**    [\(Type\)](#) - Description

*pwrSensor*    A [PowerSensor](#) (Object)

                 A [PowerSensorAsReceiver](#) (Object)

*value*        **(double)** – Cal factor in units of percent. This can be any value between 1 and 150.

**Return Type**    Double

**Default**        100

**Examples**       `pwrSensor.ReferenceCalFactor = 98 'R`  
                  `RefFact = pwrSensor.ReferenceCalFactor 'Read`

**C++ Syntax**     HRESULT put\_ReferenceCalFactor(double newVal);  
                  HRESULT get\_ReferenceCalFactor(double \*pVal);

**Interface**        IPowerSensor  
                     IPowerSensorAsReceiver

---

Last Modified:

25-Aug-2009    Added PMAR

## ReferenceMarkerState Property

---

**Description** Turn ON or OFF the reference marker.

**VB Syntax** *meas.ReferenceMarkerState = state*

**Variable** [\(Type\)](#) - Description

*app* A Measurement (**object**)

*state* (boolean) -

**True** - turns the reference marker ON

**False** - turns the reference marker OFF

**Return Type** Boolean

**Default** **False**

**Examples** `meas.ReferenceMarkerState = True`

`reference = meas.ReferenceMarkerState`

**C++ Syntax** HRESULT get\_ReferenceMarkerState(VARIANT\_BOOL bState)  
HRESULT put\_ReferenceMarkerState(VARIANT\_BOOL\* bState)

**Interface** IMeasurement

## ReferenceReceiver Property

---

**Description** Sets and returns the receiver to be used with Receiver Leveling.

**VB Syntax** *RxLevel.ReferenceReceiver(srcPort) = value*

**Variable** [\(Type\) - Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the receiver for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (String) Receiver to use for the leveling sweeps. (Not case sensitive). Choose any receiver in your PNA. see the block diagram of your PNA, located at the bottom of all [Specs documents](#).

Receivers can also be referred to using logical receiver notation. This notation makes it easy to refer to receivers with an [external test set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port. [Learn more](#).

**Return Type** String

**Default** Not applicable

**Examples** `rxLevel.ReferenceReceiver (1) = "R1" ' Write`

`rxLevel.ReferenceReceiver (1) = "b2" ' Write`

`value = rxLevel.ReferenceReceiver 2' Read`

**C++ Syntax** HRESULT get\_ReferenceReceiver( long port, BSTR\* pVal);  
HRESULT put\_ReferenceReceiver( long port, BSTR newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## ReferenceValue Property

---

**Description** Sets or returns the value of the Y-axis Reference Level of the active trace.

**VB Syntax** *trce*.**ReferenceValue** = *value*

**Variable**    [\(Type\)](#) - Description

*trce*    A Trace (**object**)

*value*    (**double**) - Reference Value. Units and range depend on the current data format.

**Return Type** Double

**Default** Not applicable

**Examples** `meas.ReferenceValue = 0 'Write`

`rlev = meas.ReferenceValue 'Read`

**C++ Syntax** HRESULT get\_ReferenceValue(double \*pVal)  
HRESULT put\_ReferenceValue(double newVal)

**Interface** ITrace

## ReferencePosition Property

---

**Description** Sets or returns the Reference Position of the active trace.

**VB Syntax** *trce*.ReferencePosition = *value*

**Variable** [\(Type\)](#) - Description

*trce* A Trace (**object**)

*value* (**double**) - Reference position on the screen measured in horizontal graticules from the bottom of the screen. Choose from any number between: **0** and **10**.

**Return Type** Double

**Default** 0

**Examples** `meas.ReferencePosition = 5 'Middle of the screen -Write`

`rpos = meas.ReferencePosition -Read`

**C++ Syntax** HRESULT get\_ReferencePosition(double \*pVal)  
HRESULT put\_ReferencePosition(double newVal)

**Interface** ITrace



## RemoteCalStoragePreference Property

---

**Description** Specifies the default manner in which calibrations performed using COM or SCPI are to be stored. Cal data is always stored to the channel's Cal Register regardless of this setting. This setting survives instrument preset and reboot. It remains until changed by another invocation of this property.

**VB Syntax** *pref.RemoteCalStoragePreference = value*

**Variable** [\(Type\)](#) - Description

*cal* A [Preferences](#) (object)

*value* (Enum) - Choose from:

**0 - naPreferCalRegister** - Cal is saved ONLY to the channel's Cal Register.

**1 - naPreferNewUserCalSet** - Cal is automatically saved to a new User Cal Set file when performing a calibration using COM. The Cal Set name is automatically generated. This corresponds to pre-6.0 behavior. Use the [Name](#) property to change the name after the cal is complete.

**2 - naPreferReuseCurrentCalSet** - The cal is saved to the Cal Set is that is currently selected on the specific channel. This could be the channel's Cal Register. If the channel does not yet have a selected Cal Set, the cal will be saved to a new User Cal Set with an automatically-generated name.

**Return Type** Enum

**Default** 0 - naPreferCalRegister

**Examples** `pref.RemoteCalStoragePreference = naPreferNewUserCalSet 'Write`

`calStorageMode = pref.RemoteCalStoragePreference ' Read`

**C++ Syntax** HRESULT get\_RemoteCalStoragePreference(enum NARemoteCalStoragePreference\* preference);  
HRESULT put\_RemoteCalStoragePreference(enum NARemoteCalStoragePreference val);

**Interface** IPreferences7

Last Modified:

16-Apr-2007    MX New topic

## ResolutionBW Property

---

**Description** Sets and returns the Resolution Bandwidth for the IM Spectrum measurement.

**VB Syntax** *ims.ResolutionBW = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Double) Resolution BW in Hz. Choose from:  
60k | 100k | 150k | 300k, 600k | 1.0M | 3.0M  
If an invalid number is specified, the PNA will round up to the closest valid number.

**Return Type** Double

**Default** 600 kHz

**Examples** `ims.ResolutionBW = 150e3 'Write`

`value = ims.ResolutionBW 'Read`

**C++ Syntax** HRESULT get\_ResolutionBW(double \*pVal)  
HRESULT put\_ResolutionBW(double newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## ReverseLinearPowerLevel Property

---

**Description** Set and read the reverse power level to the DUT. This is applied to the DUT output port when making reverse measurements like S22.

**VB Syntax** `gca.ReverseLinearPowerLevel = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(double)** Reverse power level in dBm. Choose a value from +30 to (-30).

**Return Type** Double

**Default** -5

**Examples** `gca.ReverseLinearPowerLevel = -10 'Write`

`LinPwr = gca.ReverseLinearPowerLevel 'Read`

**C++ Syntax** HRESULT get\_ReverseLinearPowerLevel(double\* pVal)  
HRESULT put\_ReverseLinearPowerLevel(double newVal)

**Interface** IGainCompression

---

Last Modified:

21-Nov-2007    MX New topic

## SafeMode Property

---

**Description** Sets and returns the state of Safe Mode.

**VB Syntax** *RxLevel*.**SafeMode**(*srcPort*) = *value*

**Variable** [\(Type\)](#) - **Description**

*RxLevel* A [ReceiverLeveling](#) Object

*value* (Boolean) Choose from:

**True** - Safe mode ON

**False** - Safe mode OFF

*srcPort* (Long Integer) Source port for which to set the Safe Mode state for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

**Return Type** Variant Boolean

**Default** False

**Examples** `rxLevel.SafeMode (1) = True ' Write`

`value = rxLevel.SafeMode 2 ' Read`

**C++ Syntax** HRESULT get\_SafeMode(long port, VARIANT\_BOOL\* pLevelingSafeMode);

HRESULT put\_SafeMode(long port, VARIANT\_BOOL LevelingSafeMode);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## SafeSweepCoarsePowerAdjustment Property

---

**Description** Set and read the Safe Sweep COARSE power adjustment.

**VB Syntax** `gca.SafeSweepCoarsePowerAdjustment = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(Double)** Coarse power adjustment setting in dBm. Choose a value from +30 to (-30).

**Return Type** Double

**Default** 3.0

**Examples** `gca.SafeSweepCoarsePowerAdjustment = 2.0 'Write`

`SSCourse = gca.SafeSweepCoarsePowerAdjustment 'Read`

**C++ Syntax** HRESULT get\_SafeSweepCoarsePowerAdjustment(double\* value)

HRESULT put\_SafeSweepCoarsePowerAdjustment(double value)

**Interface** IGainCompression

---

Last Modified:

1-Dec-2007    MX New topic

## SafeSweepEnable Property

---

**Description** Set and read the (ON | OFF) state of Safe Sweep mode.

**VB Syntax** `gca.SafeSweepEnable = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(Boolean)** Safe Sweep state. Choose from:

**False** - Disable Safe Sweep

**True** - Enable Safe Sweep

**Return Type** Boolean

**Default** False

**Examples** `gca.SafeSweepEnable = True 'Write`

`SSEnable = gca.SafeSweepEnable 'Read`

**C++ Syntax** HRESULT get\_SafeSweepEnable(VARIANT\_BOOL\* value)

HRESULT put\_SafeSweepEnable(VARIANT\_BOOL value)

**Interface** IGainCompression

---

Last Modified:

1-Dec-2007    MX New topic

## SafeSweepFinePowerAdjustment Property

---

**Description** Set and read the Safe Sweep FINE power adjustment.

**VB Syntax** `gca.SafeSweepFinePowerAdjustment = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (**Double**) Fine power adjustment setting in dBm. Choose a value from +30 to (-30).

**Return Type** Double

**Default** 1.0

**Examples** `gca.SafeSweepFinePowerAdjustment = 0.1 'Write`

`SSfine = gca.SafeSweepFinePowerAdjustment 'Read`

**C++ Syntax** HRESULT get\_SafeSweepFinePowerAdjustment(double\* value)

HRESULT put\_SafeSweepFinePowerAdjustment(double value)

**Interface** IGainCompression

---

Last Modified:

1-Dec-2007    MX New topic

## SafeSweepFineThreshold Property

---

**Description** Set and read the compression level at which Safe Sweep changes from the COARSE power adjustment to the FINE power adjustment.

**VB Syntax** `gca.SafeSweepFineThreshold = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(Double)** Threshold setting in dBm. Choose a value from +30 to (-30).

**Return Type** Double

**Default** 0.5 dBm

**Examples** `gca.SafeSweepFineThreshold = 0.1 'Write`

`SSThresh = gca.SafeSweepFineThreshold 'Read`

**C++ Syntax** HRESULT get\_SafeSweepFineThreshold(double\* value)  
HRESULT put\_SafeSweepFineThreshold(double value)

**Interface** IGainCompression

---

Last Modified:

1-Dec-2007    MX New topic



## SB\_BalPortNegative Property

---

**Description** With a Single-ended - Balanced topology, returns the PNA port number that is connected to the Negative side of the DUT's Balanced Port.

Use [SetSBPorts Method](#) to set the port mapping for a Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SB_BalPortNegative`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.SB_BalPortNegative` 'Read

**C++ Syntax** HRESULT get\_SB\_BalPortNegative(long \*bVal)

**Interface** IBalancedTopology

## SB\_BalPortPositive Property

---

**Description** With a Single-ended - Balanced topology, returns the PNA port number that is connected to the Positive side of the DUT's Balanced Port.

Use [SetSBPorts Method](#) to set the port mapping for a Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SB_BalPortPositive`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTop.SB_BalPortPositive` 'Read

**C++ Syntax** HRESULT get\_SB\_BalPortPositive(long \*bVal)

**Interface** IBalancedTopology

## SB\_SEPort Property

---

**Description** With a Single-ended - Balanced topology, returns the PNA port number that is connected to the DUT's Single-ended port.

Use [SetSBPorts Method](#) to set the port mapping for a Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SB_SEPort`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTopology.SB_SEPort` 'Read

**C++ Syntax** HRESULT get\_SB\_SEPort(long \*bVal)

**Interface** IBalancedTopology

## SBalMeasurement Property

---

**Description** Sets and returns the measurement for the Single-Ended - Balanced topology.

**VB Syntax** *balMeas.SBalMeasurement* = *value*

**Variable** [\(Type\)](#) - **Description**

*balMeas* A [BalancedMeasurement](#) (object)

*value* **(String)** - Single-ended - Balanced Measurement parameter. Not case-sensitive.  
Choose from:

Sss11	Ssd12	Ssc12
Sds21	Sdd22	Sdc22
Scs21	Scd22	Sc22
lmb	CMRR1 (Sds21/Scs21)	CMRR2 (Ssd12/Ssc12)

**Return Type** Sss11

**Default** Not Applicable

**Examples**

```
balMeas.SBalMeasurement = "Ssd12" 'Write
```

```
variable = balMeas.SBalMeasurement 'Read
```

**C++ Syntax** HRESULT get\_SBalMeasurement(BSTR \*pVal)  
 HRESULT put\_SBalMeasurement(BSTR newVal)

**Interface** IBalancedMeasurement

## Scope Property

---

**Description** Sets or returns the scope of a trigger signal. This determines whether a trigger signal affects a single channel or all channels in the PNA.

**Note:** [Trigger Modes](#) Point and EverySweep require that Trigger.Scope be set to naChannelTrigger.

**VB Syntax** *trigsetup.Scope = value*

**Variable** [\(Type\)](#) - Description

*trigsetup* A [TriggerSetup](#) (object)

*value* (enum NATriggerType) - Trigger type. Choose from:

**0 - naGlobalTrigger** - a trigger signal is applied to all triggerable channels

**1 - naChannelTrigger** - a trigger signal is applied to the current channel. The next trigger signal will be applied to the next channel; not necessarily the next channel in numeric sequence (1-2-3-4 and so forth).

**Return Type** Long Integer

**Default** naGlobalTrigger

**Examples** `trigsetup.Scope = naGlobalTrigger 'Write`

`trigtyp = trigsetup.Scope 'Read`

**C++ Syntax** HRESULT get\_Scope(tagNATriggerType \*pTrigger)  
HRESULT put\_Scope(tagNATriggerType trigger)

**Interface** ITriggerSetup

---

Last Modified:

6-Nov-2007      Updated for new sweep mode

## SearchFunction Property

---

**Description** Emulates the Tracking function in the marker search dialog box. The value you choose for SearchFunction will determine the type of search that takes place when the [Tracking](#) property is set true.

The tracking function finds the selected search function every sweep. In effect, turning Tracking ON is the same as executing one of the "Search..." methods (such as SearchMin, SearchMax) for every sweep.

**VB Syntax** *mark.SearchFunction = value*

**Variable** [\(Type\)](#) - Description

*mark* A [Marker](#) (object)

*value* (enum **NAMarkerFunction**) - search function. Choose from:

- 0 - naMarkerFunction\_None
- 1 - naMarkerFunction\_Min
- 2 - naMarkerFunction\_Max
- 3 - naMarkerFunction\_Target
- 4 - naMarkerFunction\_NextPeak
- 5 - naMarkerFunction\_PeakRight
- 6 - naMarkerFunction\_PeakLeft
- 7 - naMarkerFunction\_Compression

**Return Type** Long Integer

**Default** 0 - naMarkerFunction\_None

**Examples** `mark.SearchFunction = naMarkerFunction_Target` 'When this marker is set to track, it will track the Target value.

`searchfunction = mark.SearchFunction` 'Read

**C++ Syntax** HRESULT get\_SearchFunction(tagNAMarkerFunction \*pVal)  
 HRESULT put\_SearchFunction(tagNAMarkerFunction newVal)

**Interface** IMarker

## SearchSummary Property

---

**Description** Returns the status of a compression search.

This command can be used to indicate when a GCA search is complete.

**Note:** The returned value reflects the current state of the GCA compression search which can vary when in continuous sweep. This command is intended to be used with [chan.Single](#) (trigger).

**VB Syntax** *value* = *gca*.SearchSummary

**Variable** [\(Type\)](#) - Description

*value* **(Enum)** Variable to store the returned value.

- **0 - naSearchNotDone** - Acquisition is still in process.
- **1 - naSearchSucceeded** - Acquisition is complete and compression value found for all frequency points.
- **2 - naSearchFailed** - Acquisition is complete and unable to find the compression value at one or more frequency points.

*gca* A [GainCompression](#) (object)

**Return Type** Enum

**Default** Not Applicable

**Examples** `sum = gca.SearchSummary 'Read`

**C++ Syntax** HRESULT get\_SearchSummary(enum naGCASearchSummary\* value)

**Interface** IGainCompression

---

Last Modified:

10-Jun-2009    MX New topic

## SecurityLevel Property

---

**Description** Controls the display of frequency information on the PNA screen and printouts.

**VB Syntax** *app.SecurityLevel value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (enum **NASecurityLevel**) -Choose from:

**0 - naNoSecurity** ALL frequency information is displayed.

**1 - naLowSecurity** NO frequency information is displayed. Frequency information can be redisplayed using the Security Setting dialog box or this command.

**2 - naHighSecurity** LOW setting plus [GPIB console](#) is disabled. Frequency information can be redisplayed ONLY by performing a Preset, recalling an instrument state with None or Low security settings, or using this command.

**3 - naExtraSecurity** HIGH setting plus [ASCII data saving](#) is disabled. Same method to redisplay frequency information as HIGH setting.

**Return Type** Long Integer

**Default** 0 - None

**Examples** `app.SecurityLevel = naLowSecurity 'Write`

`level = app.SecurityLevel 'Read`

**C++ Syntax** HRESULT get\_NASecurityLevel(tagNASecurityLevel \*level);  
HRESULT put\_NASecurityLevel(tagNASecurityLevel level);

**Interface** IApplication4

---

Last Modified:

17-Jul-2007    Add Extra level



## SegmentNumber Property

---

**Description** Returns the number of the current segment, PowerSensorCalFactorSegment or PowerLossSegment object.

**VB Syntax** *seg*.SegmentNumber

**Variable** [\(Type\)](#) - Description

*seg* Any of the following objects:

A [Segment](#) (Object)

A [PowerSensorCalFactorSegment](#) (Object)

A [PowerSensorCalFactorSegmentPMAR Object](#) (Object)

A [PowerLossSegment](#) (Object)

A [PowerLossSegmentPMAR](#) (Object)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `segNum = seg.SegmentNumber` 'returns the segment number -Read

**C++ Syntax** HRESULT get\_SegmentNumber(long \*pVal)

**Interface** ISegment  
IPowerSensorCalFactorSegment  
IPowerSensorCalFactorSegmentPMAR  
IPowerLossSegment  
IPowerLossSegmentPMAR

---

Last Modified:

27-Aug-2009    Added PMAR (9.0)

## SelectPort Property

---

**Description** Sets and returns a port mapping for a single port. If this command creates a conflict with an existing port, the PNA will resolve the conflict.

**Note:** This command is currently not supported for the Z5623AK44.

**VB Syntax** *tset.SelectPort(chNum, portNum) = portValue*

**Variable** [\(Type\) - Description](#)

*tset* A [TestsetControl](#) object.

*chNum* **(Long)** Channel number of the measurement.

*portNum* **(Long)** Physical port number to map.

*portValue* **(Long) Logical** port value to assign

**Return Type** Long

**Default** Not Applicable

**Examples** [See External Testset Program](#)

**C++ Syntax** HRESULT get\_SelectPort(long channelNum, long PortNum long \*outPort);  
HRESULT put\_SelectPort(long channelNum, long PortNum long outPort);

**Interface** ITestsetControl

## ShowStatistics Property

---

**Description** Displays and hides the measurement (Trace) statistics (peak-to-peak, mean, standard deviation) on the screen. To display measurement statistics for a narrower band of the X-axis, use [StatisticsRange](#).

The analyzer will display either measurement statistics or Filter Bandwidth statistics; not both.

**VB Syntax** *meas.ShowStatistics* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**boolean**) - Boolean value:

**True** - Show statistics

**False** - Hide statistics

**Return  
Type** Boolean

**Default** **False**

**Examples** `meas.ShowStatistics = True 'Write`

`showstats = meas.ShowStatistics 'Read`

**C++ Syntax** HRESULT put\_ShowStatistics(VARIANT\_BOOL bState)

**Interface** IMeasurement

## ShowProperties Property

---

**Description** Turns ON and OFF the display of the test set control status bar. This status bar indicates the test set that is being controlled and the current port mappings.

This setting is turned ON and OFF automatically when the test set is enabled or disabled.

**VB Syntax** *tset*.**ShowProperties** = *value*

**Variable** [\(Type\)](#) - Description

*tset* A [TestsetControl](#) object.

OR

An [E5091Testset](#) object.

*value* **(Boolean)**

**True** - Turns display of testset properties ON.

**False** - Turns display of testset properties OFF.

**Return Type** Boolean

**Default** **False** (True when test set control is enabled.)

**Examples** [See E5091A Example Program](#)

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_ShowProperties(VARIANT\_BOOL \*state);  
HRESULT put\_ShowProperties(VARIANT\_BOOL state);

**Interface** IE5091Testsets  
ITestsetControl

## SIDL Property

---

**Description** Allows you to control the PNA via SIDL (standard instrument control library). In this mode, the analyzer can receive SCPI commands from the LAN interface or from a program residing on the PNA itself. This command performs the same function as the [SIDL / GPIB](#) dialog box - **SIDL Enabled** checkbox. [See Configuring the analyzer for SIDL/VISA](#).

With this method you can augment a test program written using SIDL that resides on the PNA so that it will run unattended. An automation script can be written to start the PNA, enable SIDL (using the SIDL property), and then start the SIDL based program.

**VB Syntax** *app.SIDL value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* **(Boolean)** Choose from:

**True** - enable SIDL

**False** - disable SIDL

**Return Type** Boolean

**Default** False

**Examples**

```
Dim Pna as AgilentPNA835x.Application
Dim siclState as Boolean
Set Pna = CreateObject("AgilentPNA835x.Application")
Pna.SIDL = true           'write

siclState = Pna.SIDL      'Read
```

**C++ Syntax** HRESULT get\_SIDL(VARIANT\_BOOL \*pVal)  
 HRESULT put\_SIDL(VARIANT\_BOOL newVal)

**Interface** IApplication5

## SICLAddress Property

---

**Description** Sets and returns the PNA SICL address. This is the address used for SICL over LAN.

**VB Syntax** *app.SICLAddress = value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (Integer) SICL Address of the PNA. Choose a value between 0 and 30.

**Return Type** Short Integer

**Default** 16

**Examples** `address=app.SICLAddress 'Read`

`app.SICLAddress=16 'Write`

**C++ Syntax** HRESULT get\_SICLAddress(short busIndex, short\* address);  
HRESULT put\_SICLAddress(short busIndex,short address);

**Interface** IApplication8

## Simultaneous2PortAcquisition Property

---

<b>Description</b>	<p>Specifies whether a 2-port calibration will be done with a single set of standards (one port at a time) or with two sets of standards (simultaneously).</p> <p>The <a href="#">AcquireCalStandard2</a> command uses the same standard index for each calibration class. To specify the calibration standard gender for each port, you must first ensure that the order of calibration class accurately reflects the configuration of your DUT. For example, for a DUT with a male connector on port 1 and a female connector on port 2, order the devices within the S11 classes (A, B, and C) such that the MALE standards are first in the list. Then order the S22 classes specifying the FEMALE standards as the first in the list.</p>
<b>VB Syntax</b>	<code>cal.Simultaneous2PortAcquisition = state</code>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<code>cal</code>	A Calibrator ( <b>object</b> )
<code>state</code>	<b>(boolean)</b> - Choose from: <b>True</b> - measures 2 ports simultaneously <b>False</b> - measures 1 port at a time
<b>Return Type</b>	Boolean
<b>Default</b>	True
<b>Examples</b>	<code>cal.Simultaneous2PortAcquisition = True</code>
<b>C++ Syntax</b>	HRESULT put_Simultaneous2PortAcquisition( VARIANT_BOOL bTwoSetsOfStandards) HRESULT Simultaneous2PortAcquisition( VARIANT_BOOL *bTwoSetsOfStandards)
<b>Interface</b>	ICalibrator

---

Last modified:

- 9/20/06    Changed default to True
- 9/12/06    Modified for cross-browser

## SmartSweepMaximumIterations Property

---

**Description** Set and read the maximum permitted number of iterations which SMART Sweep may utilize to find the desired compression level, to within the specified tolerance.

**VB Syntax** `gca.SmartSweepMaximumIterations = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (integer) - Maximum number of iterations. Choose a value between 1 and 50.

**Return Type** Integer

**Default** 20

**Examples** `gca.SmartSweepMaximumIterations = 10 'Write`

`iters = gca.SmartSweepMaximumIterations 'Read`

**C++ Syntax** HRESULT get\_SmartSweepMaximumIterations(int\* pVal)  
HRESULT put\_SmartSweepMaximumIterations(int newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic



## SmartSweepSettlingTime Property

---

**Description** Set and read the amount of time SMART Sweep will dwell at the first point where the input power changes by the Backoff or X level.

[Learn more.](#)

**VB Syntax** `gca.SmartSweepSettlingTime = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(double)** - Settling time in seconds. Choose any positive value.

**Return Type** Double

**Default** 0

**Examples**

```
gca.SmartSweepSettlingTime = .01 'Write
```

```
sTime = gca.SmartSweepSettlingTime 'Read
```

**C++ Syntax** HRESULT get\_SmartSweepSettlingTime(double\* pVal)  
HRESULT put\_SmartSweepSettlingTime(double newVal)

**Interface** IGainCompression

---

Last Modified:

21-Nov-2007    MX New topic

## SmartSweepShowIterations Property

---

**Description** Set and read whether to show intermediate results for each iteration in SMART sweep.

**VB Syntax** `gca.SmartSweepShowIterations = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* **(Boolean)** Choose from:

**True** Compression traces are updated after each iteration.

**False** Compression traces are updated after ALL iterations are complete.

**Return Type** Boolean

**Default** False

**Examples** `gca.SmartSweepShowIterations = True 'Write`

`SShow = gca.SmartSweepShowIterations 'Read`

**C++ Syntax** HRESULT get\_SmartSweepShowIterations(VARIANT\_BOOL \*pVal)  
HRESULT put\_SmartSweepShowIterations(VARIANT\_BOOL newVal)

**Interface** IGainCompression

---

Last Modified:

21-Nov-2007    MX New topic

## SmartSweepTolerance Property

---

**Description** Set and read the acceptable range SMART Sweep will allow for the measured compression level.

**VB Syntax** `gca.SmartSweepTolerance = value`

**Variable** [\(Type\)](#) - Description

*gca* A [GainCompression](#) (object)

*value* (**double**) - Tolerance level in dB. Choose a value between .01 and 10

**Return Type** Double

**Default** .05

**Examples** `gca.SmartSweepTolerance = .01 'Write`

`tol = gca.SmartSweepTolerance 'Read`

**C++ Syntax** HRESULT get\_SmartSweepTolerance(double\* pVal)  
HRESULT put\_SmartSweepTolerance(double newVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## SmoothingAperture Property

---

**Description** Specifies or returns the amount of smoothing as a ratio of the number of data points in the measurement trace.

There is no COM command for specifying smoothing by number of aperture points.

**VB Syntax** *meas.SmoothingAperture* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*value* (**double**) - Smoothing Aperture. A ratio of (aperture points / trace points). Choose any number between **.01** and **.25**.

**Return Type** Double

**Default** .25

**Examples** `meas.SmoothingAperture = .10 'Write`

`saperture = meas.SmoothingAperture 'Read`

**C++ Syntax** HRESULT get\_SmoothingAperture(double \*pVal)  
HRESULT put\_SmoothingAperture(double newVal)

**Interface** IMeasurement

---

Last modified:

Oct. 25, 2006    Fixed formula for smoothing

## Smoothing Property

---

**Description** Turns ON and OFF data smoothing.

**VB Syntax** *meas.Smoothing = state*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*state* (**boolean**)

**True** - Turns smoothing ON

**False** - Turns smoothing OFF

**Return Type** Boolean

**Default** **False**

**Examples** `meas.Smoothing = False 'Write`

`smooth = meas.Smoothing 'Read`

**C++ Syntax** HRESULT get\_Smoothing(VARIANT\_BOOL \*pVal)  
HRESULT put\_Smoothing(VARIANT\_BOOL newVal)

**Interface** IMeasurement

## SnPFormat Property

---

**Description** Specifies the format of .SnP files.

Use either app.[Save](#) (saves data to file) or meas.[GetSnPData](#) (reads data into variant array).

**VB Syntax** *pref.SnPFormat = value*

**Variable** [\(Type\)](#) - Description

*pref* A [Preferences](#) (object)

*value* **(string)** - Format of the .S1P, .S2P, .S3P, .S4P data. Choose from:

"MA" - Linear Magnitude / degrees

"DB" - Log Mag / degrees

"RI" - Real / Imaginary

"Auto" - Format in which the trace is already displayed. If other than Log Mag, Linear Magnitude, or Real/Imag, then the format will be in Real/Imag.

**Return  
Type** String

**Default** "Auto"

**Examples** `pref.SnPFormat = "MA" 'Write`

`format = pref.SnPFormat 'Read`

**C++ Syntax** HRESULT get\_SnPFormat(BSTR \*Format)  
HRESULT put\_SnPFormat(BSTR Format)

**Interface** IPreferences

## SoundOnFail Property

---

**Description** Turns ON or OFF the audio indicator for limit failures.

**VB Syntax** *limitst*.SoundOnFail = *state*

**Variable**    [\(Type\)](#) - Description

*limitst*    A LimitTest (**object**)

*state*    (**boolean**)

**False** - Turns the sound OFF

**True** - Turns the sound ON

**Return Type** Long Integer

**Default**    **True**

**Examples**    `Limttest.SoundOnFail = False 'Write`

`sound = Limttest.SoundOnFail 'Read`

**C++ Syntax**    HRESULT get\_SoundOnFail(VARIANT\_BOOL \*pVal)  
                  HRESULT put\_SoundOnFail(VARIANT\_BOOL newVal)

**Interface**    ILimitTest

## SourceCount Property

---

**Description** Returns the number of sources in the remote PNA.

**VB Syntax** *value* = *cap*.SourceCount

**Variable** [\(Type\)](#) - Description

*value* (Long) - Variable to store the returned number of sources.

*cap* A [Capabilities](#) (object)

**Return Type** Long

**Default** Not Applicable

**Examples** `value = cap.SourceCount 'Read`

**C++ Syntax** HRESULT get\_SourceCount(long \* sourceCount );

**Interface** ICapabilities



Read-only

## SourcePort Property

---

**Description** Returns the source port of measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**SourcePort**

**Variable** [\(Type\)](#) - **Description**

*meas* A Measurement (**object**)

*value* (**Long**) - Variable to store the returned value

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `sp = meas.SourcePort`

**C++ Syntax** HRESULT SourcePort( [out, retval] Long\* srcPort);

**Interface** IMeasurement2

## SourcePortCount Property

---

**Description** Returns the number of ports that can output a signal.  
To learn more, see [Remotely Specifying a Source Port](#).

**VB Syntax** *value* = *object*.**SourcePortCount**

**Variable** [\(Type\)](#) - **Description**

*value* (Long) - Variable to store the returned integer value of the number of source ports.

*object* A [Channel](#) (**object**)- always more complete than capabilities object.  
A [Capabilities](#) (**object**) - use when a channel is not available, or to find the common ports across all channels.

**Return Type** Long

**Default** Not Applicable

**Examples** `value = chan.SourcePortCount 'Read`

**C++ Syntax** HRESULT get\_SourcePortCount(long \* count );

**Interface** IChannel13  
ICapabilities4

---

### Last Modified:

23-May-2008 Added channel object

14-Jan-2007 MX New topic

## SourcePortMode Property

---

**Description** Sets the state of the PNA source for the specified port.

**VB Syntax** *chan.SourcePortMode (sourcePort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*chan* **(object)** - A [Channel](#) object

*sourcePort* **(long integer)** - The source port for which to make this setting.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* **(enum)** - State of the source. Choose from:

0 - naSourcePortAuto Port power is turned on when required for a measurement.

1 - naSourcePortOn Port power is always ON, regardless of the measurement.

2 - naSourcePortOff Port power is always OFF, regardless of the measurement.

**Note:** naSourcePortOn and naSourcePortOff are valid only on [PNA models with two sources](#).

**Return Type** Enum

**Default** 0 - naSourcePortAuto

**Examples**

```
chan.SourcePortMode(1) = naSourcePortOn 'Write
state = chan.SourcePortMode(4) 'Read
```

**C++ Syntax** HRESULT get\_SourcePortMode(long sourcePort, enum NASourcePortMode\*);  
HRESULT put\_SourcePortMode(long sourcePort, enum NASourcePortMode);

**Interface** IChannel9

---

Last modified:

24-Apr-2008 Added note for strings

30-Apr-2007 Edited for src strings

10/18/06 MQQ New topic

## SourcePortNames Property

---

**Description** Returns the string names of ports that can output a signal.

The following is a list of string names for the PNA-X. Your PNA will NOT have all of these ports. Use [GetPortNumber Method](#) to return the correct port number for the specified port name.

- "Port 1"
- "Port 2"
- "Port 3"
- "Port 4"
- "Src2 Out1"
- "Src2 Out2"
- "Port 1 Src2"

For [iTMSA \(Opt 460\)](#)

- "Bal Port 1"
- "Bal Port 2"
- "SE Port1"
- "SE Port 2"

This command also lists the [External Sources](#) that are currently configured and selected.

To learn more, see [Remotely Specifying a Source Port](#).

**VB Syntax** *value* = *object*.SourcePortNames

**Variable** [\(Type\)](#) - **Description**

*value* (Variant array) - Variable to store the returned integer value of the number of source ports.

*object* A [Channel \(object\)](#) - always more complete than capabilities object.

A [Capabilities \(object\)](#) - use when a channel is not available, or to find the common ports across all channels.

**Return Type** Variant array of string names.

**Default** Not Applicable

**Examples** `value = chan.SourcePortNames 'Read`

**C++ Syntax** HRESULT get\_SourcePortNames(VARIANT \*names) ;

**Interface** IChannel13  
ICapabilities4

---

Last Modified:

3-Mar-2009 Added SE Port 2

23-May-2008 Added iTMSA sources and channel object

23-Jul-2007 Clarification

14-Jan-2007 MX New topic

## Source Property

---

**Description** Sets or returns the source of triggering in the PNA.

**VB Syntax** *trigSetup.Source = value*

**Variable** [\(Type\)](#) - Description

*trigSetup* A [TriggerSetup](#) (object)

*value* (enum **NATriggerSource**) - Choose from:

**0 - naTriggerSourceInternal** - free run

**1 - naTriggerSourceManual** - manual trigger source; use app.[ManualTrigger](#) to send a trigger signal.

**2 - naTriggerSourceExternal** - a trigger signal is generated when a trigger signal is sensed on the [external trigger pin](#) of the Aux IO connector. Use [ExternalTriggerConnectionBehavior](#) to configure the characteristics of the external trigger signal.

This setting has implications on Calibration. [Learn more.](#)

**Return Type** Long Integer

**Default** naTriggerSourceInternal

**Examples** `trigSetup.Source = naTriggerSourceInternal 'Write`

`trigsource = trigSetup.Source 'Read`

**C++ Syntax** HRESULT get\_Source(tagNATriggerSource \*pTrigger);  
 HRESULT put\_Source(tagNATriggerSource trigger);

**Interface** ITriggerSetup

## SourcePowerCalPowerOffset Property

---

**Description** Sets or returns a power level offset from the PNA test port power. This can be a gain or loss value (in dB) to account for components you connect between the source and the reference plane of your measurement. For example, specify 10 dB to account for a 10 dB amplifier at the input of your DUT.

Cal power is the sum of the test port power setting and this offset value. Following the calibration, the PNA power readouts are adjusted to the cal power.

This property performs the same function as the power offset argument on [SetCalInfoEx Method](#), except that this property can read the offset value.

**VB Syntax** *chan.SourcePowerCalPowerOffset (sourcePort) = value*

**Variable** [\(Type\)](#) - **Description**

*chan* **(object)** - A [Channel](#) object

*sourcePort* **(long integer)** - The source port for which to set this power offset value.

*value* **(double)** - Gain or loss value in dB. Choose a value between -200 and 200.

**Return Type** Double

**Default** 0 dB

**Examples**

```
chan.SourcePowerCalPowerOffset(1) = 10 'Write  
offset = chan.SourcePowerCalPowerOffset(2) 'Read
```

**C++ Syntax** HRESULT get\_SourcePowerCalPowerOffset(long sourcePort, double \*pVal);  
HRESULT put\_SourcePowerCalPowerOffset(long sourcePort, double newVal);

**Interface** IChannel4

---

Last Modified:

1-May-2007    Modified link to EX method.

## SourcePowerCorrection Property

---

**Description** Sets source power correction ON or OFF for a specific source port on this channel, or returns the current ON or OFF state of correction for that source port.

**VB Syntax** *chan*.**SourcePowerCorrection** (*srcPort*) = *value*

**Variable** [\(Type\)](#) - Description

*chan* **(object)** – A Channel object

*srcPort* **(long integer)** – Source port for which to set or return the ON or OFF state of source power correction.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* **(boolean)**

False – Turns source power correction OFF for the source port.

True – Turns source power correction ON for the source port.

**Return Type** Boolean

**Default** False - Source power correction will turn correction ON

**Examples** `chan.SourcePowerCorrection(1) = False 'Write`  
`calOnPort2 = chan.SourcePowerCorrection(2) 'Read`

**C++ Syntax** HRESULT put\_SourcePowerCorrection(VARIANT\_BOOL bState);  
HRESULT get\_SourcePowerCorrection(VARIANT\_BOOL \*bState);

**Interface** IChannel

---

Last Modified:

24-Apr-2008 Added note for string names

30-Apr-2007 Modified for src strings



## SourcePowerOption Property

---

**Description** Enables the source power to be set on individual sweep segments. This property must be set True **before** seg.TestPortPower = *value* is sent. Otherwise, the test port power command will be ignored.

**VB Syntax** *segs.SourcePowerOption* = *state*

**Variable** [\(Type\)](#) - [Description](#)

*segs* A Segments collection (**object**)

*state* (**boolean**)

**True** - Enables variable TestPortPower to be set segment sweep

**False** - Disables variable TestPortPower to be set segment sweep

**Return Type** Boolean

**Default** False

**Examples** `segs.SourcePowerOption = True 'Write`

`powerOption = SourcePowerOption 'Read`

**C++ Syntax** HRESULT get\_SourcePowerOption(VARIANT\_BOOL \*pVal)  
HRESULT put\_SourcePowerOption(VARIANT\_BOOL newVal)

**Interface** ISegments

## SourcePowerState Property

---

**Description** Turns Source Power ON and OFF.

[See note about source power state with instrument state save and recall.](#)

**VB Syntax** *app.SourcePowerState = state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (boolean)

**False** - Turns Source Power OFF

**True** - Turns Source Power ON

**Return Type** Boolean

**Default** True

**Examples** `app.SourcePowerState = True 'Write`

`pwr = app.SourcePowerState 'Read`

**C++ Syntax** HRESULT get\_SourcePowerState(VARIANT\_BOOL \*pVal)  
HRESULT put\_SourcePowerState(VARIANT\_BOOL newVal)

**Interface** IApplication

## Span Property

---

<b>Description</b>	Sets or returns the Span time of either Gating or Time Domain transform windows
<b>VB Syntax</b>	<i>object</i> . <b>Span</b> = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>object</i>	<b>(object)</b> As Gating or <b>(object)</b> As Transform
<i>value</i>	<b>(double)</b> - Span time in seconds. Choose any number between: $2 * [(number\ of\ points - 1) / frequency\ span]$ and 0
<b>Return Type</b>	Double
<b>Default</b>	20ns
<b>Examples</b>	<pre>Trans.Span = 4.5e-9 'sets the time span of a transform window - Write Gate.Span = 4.5e-9 'sets the Span time of a gating window -Write span = Trans.Span 'Read</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT get_Span(double *pVal) HRESULT put_Span(double newVal)
<b>Interface</b>	ITransform IGating

## Span Property

---

**Description** Returns the stimulus span of the measurement (stop-start data points). To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**Span**

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** - Variable to store the returned value.

*meas* A Measurement **(object)**

**Return Type** Double

**Default** Not Applicable

**Examples** `Print meas.Span 'prints the span of the measurement`

**C++ Syntax** HRESULT get\_Span(double \* Val);

**Interface** IMeasurement2

## SpectrumCenterFrequency Property

---

**Description** Sets and returns the receiver Center frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when [Sweep Type](#) = Linear. Otherwise, this setting is ignored.

**VB Syntax** *ims.SpectrumCenterFrequency = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Double) Center frequency in Hz. Choose a frequency within the range of the PNA.

**Return Type** Double

**Default** 1.0 GHz

**Examples** `ims.SpectrumCenterFrequency = 10e9 'Write`

`value = ims.SpectrumCenterFrequency 'Read`

**C++ Syntax** HRESULT get\_SpectrumCenterFrequency(double \*pVal)  
HRESULT put\_SpectrumCenterFrequency(double newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## SpectrumSpanFrequency Property

---

**Description** Sets and returns the Span of receiver frequencies for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when [Sweep Type](#) = Linear. Otherwise, this setting is ignored.

**VB Syntax** *ims.SpectrumSpanFrequency = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Double) Frequency span in Hz. Choose a frequency within the range of the PNA.

**Return Type** Double

**Default** 100 MHz

**Examples** `ims.SpectrumSpanFrequency = 10e9 'Write`

`value = ims.SpectrumSpanFrequency 'Read`

**C++ Syntax** HRESULT get\_SpectrumSpanFrequency(double \*pVal)  
HRESULT put\_SpectrumSpanFrequency(double newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## SpectrumStartFrequency Property

---

**Description** Sets and returns the receiver Start frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

**VB Syntax** *ims.SpectrumStartFrequency = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Double) Start frequency in Hz. Choose a frequency within the range of the PNA.

**Return  
Type** Double

**Default** 950 MHz

**Examples** `ims.SpectrumStartFrequency = 10e9 'Write`

`value = ims.SpectrumStartFrequency 'Read`

**C++ Syntax** HRESULT get\_SpectrumStartFrequency(double \*pVal)  
HRESULT put\_SpectrumStartFrequency(double newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## SpectrumStopFrequency Property

---

**Description** Sets and returns the receiver Stop frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

**VB Syntax** *ims.SpectrumStopFrequency = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Double) Stop frequency in Hz. Choose a frequency within the range of the PNA.

**Return  
Type** Double

**Default** 950 MHz

**Examples** `ims.SpectrumStopFrequency = 10e9 'Write`

`value = ims.SpectrumStopFrequency 'Read`

**C++ Syntax** HRESULT get\_SpectrumStopFrequency(double \*pVal)  
HRESULT put\_SpectrumStopFrequency(double newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic



## SSB\_BalPortNegative Property

---

**Description** With a Single-ended - Single-ended - Balanced topology, returns the PNA port number that is connected to the Negative side of the DUT's Balanced Port.

Use [SetSSBPorts Method](#) to set the port mapping for a Single-Ended - Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SSB_BalPortNegative`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTopology.SSB_BalPortNegative` 'Read

**C++ Syntax** HRESULT get\_SSB\_BalPortNegative(long \*bVal)

**Interface** IBalancedTopology

## SSB\_BalPortPositive Property

---

**Description** With a Single-ended - Single-ended - Balanced topology, returns the PNA port number that is connected to the Positive side of the DUT's Balanced Port.

Use [SetSSBPorts Method](#) to set the port mapping for a Single-Ended - Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SSB_BalPortPositive`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTopology.SSB_BalPortPositive` 'Read

**C++ Syntax** HRESULT get\_SSB\_BalPortPositive(long \*bVal)

**Interface** IBalancedTopology

## SSB\_SEPort1 Property

---

**Description** With a Single-ended - Single-ended - Balanced topology, returns the PNA port number that is connected to the DUT's Logical Port 1.

Use [SetSSBPorts Method](#) to set the port mapping for a Single-Ended - Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SSB_SEPort1`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTopology.SSB_SEPort1` 'Read

**C++ Syntax** HRESULT get\_SSB\_SEPort1(long \*bVal)

**Interface** IBalancedTopology

## SSB\_SEPort2 Property

---

**Description** With a Single-ended - Single-ended - Balanced topology, returns the PNA port number that is connected to the DUT's Logical Port 2.

Use [SetSSBPorts Method](#) to set the port mapping for a Single-Ended - Single-Ended - Balanced topology.

**VB Syntax** `var = balTopology.SSB_SEPort2`

**Variable** [\(Type\)](#) - Description

`var` (Long Integer) Variable to store the returned value.

`balTopology` A [BalancedTopology](#) (**object**)

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `variable = balTopology.SSB_SEPort2` 'Read

**C++ Syntax** HRESULT get\_SSB\_SEPort2(long \*bVal)

**Interface** IBalancedTopology

## SSBMeasurement Property

---

**Description** Sets and returns the measurement for the Single-Ended - Single-Ended - Balanced topology.

**VB Syntax** *balMeas.SSBMeasurement = value*

**Variable** [\(Type\)](#) - Description

*balMeas* A [BalancedMeasurement](#) (object)

*value* **(String)** - Single-ended - Single-ended - Balanced Measurement parameter. Not case sensitive. Choose from:

Sss11	Sss12	Ssd13	Scs13
Sss21	Sss22	Ssd23	Scs23
Sds31	Sds32	Sdd33	Sdc33
Scs31	Scs32	Scd33	Scs33
lmb1	lmb2	CMRR1 (Sds31/Scs31)	CMRR2 (Sds32/Scs32)

**Return Type** String

**Default** Sss11

**Examples**

```
balMeas.SSBMeasurement = "Sss11" 'Write
```

```
variable = balMeas.SSBMeasurement 'Read
```

**C++ Syntax** HRESULT get\_SSBMeasurement(BSTR \*pVal)  
HRESULT put\_SSBMeasurement(BSTR p newVal)

**Interface** IBalancedMeasurement

## Stage1Coefficients Property

---

**Description** Sets and returns the digital filter coefficients of stage1.

**VB Syntax** `spm4.Stage1Coefficients = value`

**Variable** [\(Type\)](#) - Description

`spm4` A [SignalProcessingModuleFour](#) (object)

`value` **(Variant Array)** Coefficients. An array of real values.

**Return Type** Variant

**Default** Stage dependent.

**Examples** `spm4.Stage1Coefficients = 0,0.1,0.7,0.7,0.1 'Write`  
`mode = spm4.Stage1Coefficients 'Read`

**C++ Syntax** HRESULT get\_Stage1Coefficients(VARIANT\* pCoefs);  
HRESULT put\_Stage1Coefficients(VARIANT pCoefs);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

18-Jun-2007    MX New topic

## Stage1Frequency Property

---

**Description** Sets and returns the Numerically Controlled Oscillator (NCO) frequency of the Stage 1 filter. This command is only used when [FilterMode Property](#) is set to Manual.

**VB Syntax** `spm4.Stage1Frequency = value`

**Variable** [\(Type\)](#) - Description

*spm4* A [SignalProcessingModuleFour](#) (object)

*value* **(Double)** Stage 1 Frequency. Min value= 0 Hz, Max value= 15 MHz. Or programmatically use [MinimumIFFrequency Property](#) and [MaximumIFFrequency Property](#) to determine the range of settable values.

**Return Type** Double

**Default** 9 MHz

**Examples** `spm4.Stage1Frequency = 9E6 'Write`

`mode = spm4.Stage1Frequency 'Read`

**C++ Syntax** HRESULT get\_Stage1Frequency(double \*val);  
HRESULT put\_Stage1Frequency(double val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

18-Jun-2007  
18-Jun-2007  
2007 MX New topic

## Stage1MaximumCoefficient Property

---

**Description** Returns the maximum value of any single coefficient.

**VB Syntax** `value = spm4.Stage1MaximumCoefficient`

**Variable** [\(Type\)](#) - Description

`value` **(Long)** Variable to store the returned Max coefficient.

`spm4` A [SignalProcessingModuleFour](#) **(object)**

**Default** Not Applicable

Examples 

```
mode = spm4.Stage1MaximumCoefficient
'Read
```

**C++ Syntax** HRESULT get\_Stage1MaximumCoefficient(long\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic



## Stage1MaximumCoefficientCount Property

---

**Description** Returns the maximum number of coefficients for Stage1.

**VB Syntax** *value* = *spm4*.**Stage1MaximumCoefficientCount**

**Variable** [\(Type\)](#) - Description

*value* **(Long)** Variable to store the returned Max coefficient count.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Default** Not Applicable

Examples 

```
mode = spm4.Stage1MaximumCoefficientCount
'Read
```

**C++ Syntax** HRESULT get\_Stage1MaximumCoefficientCount(long\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage1MaximumCoefficientSum Property

---

**Description** Returns the maximum sum of all Stage1 coefficients.

**VB Syntax** *value* = *spm4*.**Stage1MaximumCoefficientSum**

**Variable** [\(Type\)](#) - Description

*value* (**\_\_int64\* val**) Variable to store the returned Max sum of all coefficients.

*spm4* A [SignalProcessingModuleFour](#) (**object**)

**Default** Not Applicable

Examples `mode = spm4.Stage1MaximumCoefficientSum 'Read`

**C++ Syntax** HRESULT get\_Stage1MaximumCoefficientSum(\_\_int64\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage1MinimumCoefficientCount Property

---

<b>Description</b>	Returns the minimum number of coefficients for Stage1
<b>VB Syntax</b>	<i>value</i> = <i>spm4</i> . <b>Stage1MinimumCoefficientCount</b>
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>value</i>	<b>(Long)</b> Variable to store the returned Min coefficient count.
<i>spm4</i>	A <a href="#">SignalProcessingModuleFour</a> <b>(object)</b>
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>mode = spm4.Stage1MinimumCoefficientCount 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_Stage1MinimumCoefficientCount(long* val);
<b>Interface</b>	ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage2Coefficients Property

---

**Description** Sets and returns Stage2Coefficients.

**VB Syntax** `spm4.Stage2Coefficients = value`

**Variable** [\(Type\)](#) - Description

`spm4` A [SignalProcessingModuleFour](#) (object)

`value` (Variant) An array of real numbers. Filter coefficients

**Return Type** Variant

**Default** Not Applicable

**Examples** `spm4.Stage2Coefficients = 'Write`

`mode = spm4.Stage2Coefficients 'Read`

**C++ Syntax** HRESULT get\_Stage2Coefficients(VARIANT\* pCoefs);  
HRESULT put\_Stage2Coefficients(VARIANT pCoefs);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage2MaximumCoefficient Property

---

**Description** Returns the maximum value of any single coefficient.

**VB Syntax** *value* = *spm4*.**Stage2MaximumCoefficient**

**Variable** [\(Type\)](#) - Description

*value* **(Long)** Variable to store the returned Max coefficient.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Default** Not Applicable

Examples 

```
mode = spm4.Stage2MaximumCoefficient
'Read
```

**C++ Syntax** HRESULT get\_Stage2MaximumCoefficient(long\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage2MaximumCoefficientCount Property

---

**Description** Returns the maximum number of coefficients for Stage2

**VB Syntax** *value* = *spm4*.**Stage2MaximumCoefficientCount**

**Variable** [\(Type\)](#) - Description

*value* **(Long)** Variable to store the returned Max coefficient count.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Default** Not Applicable

Examples 

```
mode = spm4.Stage2MaximumCoefficientCount
'Read
```

**C++ Syntax** HRESULT get\_Stage2MaximumCoefficientCount(long\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage2MaximumCoefficientSum Property

---

**Description** Returns the maximum sum of all Stage2 coefficients.

**VB Syntax** *value* = *spm4*.**Stage2MaximumCoefficientSum**

**Variable** [\(Type\)](#) - Description

*value* (**\_\_int64\* val**) Variable to store the returned Max sum of all coefficients.

*spm4* A [SignalProcessingModuleFour](#) (**object**)

**Default** Not Applicable

Examples `mode = spm4.Stage2MaximumCoefficientSum 'Read`

**C++ Syntax** HRESULT get\_Stage2MaximumCoefficientSum(\_\_int64\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage2MinimumCoefficientCount Property

---

**Description** Returns the minimum number of coefficients for Stage2

**VB Syntax** *value* = *spm4*.**Stage2MinimumCoefficientCount**

**Variable** [\(Type\)](#) - Description

*value* **(Long)** Variable to store the returned Min coefficient count.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Default** Not Applicable

**Examples**

```
mode = spm4.Stage2MinimumCoefficientCount
'Read
```

**C++ Syntax** HRESULT get\_Stage2MinimumCoefficientCount(long\* val);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic



## Stage3FilterType Property

---

**Description** Sets and returns the Stage 3 filter type. This command is only used when [FilterMode](#) is set to Manual.

**VB Syntax** `spm4.Stage3FilterType = value`

**Variable** [\(Type\)](#) - Description

*spm4* A [SignalProcessingModuleFour](#) (object)

*value* (String) Filter type. Chose from:

- "RECT" Rectangular Window Filter
- "TUKEY" Tukey Filter
- "PWIN" Pulse window filter

**Default** TUKEY

**Examples** `spm4.Stage3FilterType = "PWIN"`

```
mode = spm4.Stage3FilterType 'Read
```

**C++ Syntax** HRESULT get\_Stage3FilterType(BSTR\* pFType);  
HRESULT put\_Stage3FilterType(BSTR GType);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage3FilterTypes Property

---

**Description** Returns a list of strings for the currently supported filter types that can be used for the stage 3 filter. This command is only used when [FilterMode](#) is set to False (Manual). See [Stage3FilterType](#) for a list of currently supported filter types.

**VB Syntax** *values* = *spm4*.Stage3FilterTypes

**Variable** [\(Type\)](#) - Description

*value* **(Variant)** Variable to store the returned filter types.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Return Type** Variant Array

**Default** Not Applicable

**Examples** `mode = spm4.Stage3FilterTypes 'Read`

**C++ Syntax** HRESULT get\_Stage3FilterTypes(VARIANT\* pTypes);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage3Parameter Property

---

**Description** Sets and returns the Stage 3 filter parameters.

Must first select the filter type using [Stage3FilterType](#) before setting these parameters

Use [Stage3Parameters](#) to return a list of the available parameters for the currently selected filter type.

**VB Syntax** `spm4.Stage3Parameter(param) = value`

**Variable** [\(Type\)](#) - Description

*spm4* A [SignalProcessingModuleFour](#) (object)

*param* **(String)** Filter parameter. Choose from:

"C" - Tap count (Tukey, RECT, PWIN)

"P" - Period (PWIN ONLY)

"D" - Delay (PWIN ONLY)

"W" - Width (PWIN ONLY)

"R" - Ramp Count (PWIN ONLY)

*value* **(String)** Parameter Value for the specified stage 3 parameter. Use [Stage3ParameterMaximum](#) and [Stage3ParameterMinimum](#) to return a range of values for the specified parameter.

**Default** **RECT:** C = 1

**PWIN:** C=1E6, P=10ms, D=50us, W=50us, R=7

**TUKEY:** C=1

**Examples** `spm4.Stage3Parameter("C") = 2`

`mode = spm4.Stage3Parameter("pwin") 'Read`

**C++ Syntax** `HRESULT get_Stage3Parameter(BSTR pName, double* pVal);`

`HRESULT put_Stage3Parameter(BSTR pName, double pVal);`

**Interface** `ISignalProcessingModuleFour`

Last Modified:

1-Jan-2007    MX New topic

## Stage3ParameterMaximum Property

---

**Description** Returns maximum parameter value for the current filter type.

**VB Syntax** *values=* `spm4.Stage3ParameterMaximum (parameter)`

**Variable** [\(Type\)](#) - Description

*value* **(Variant)** Variable to store the maximum parameter value.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

*parameter* **(String)** Parameter name. See [Stage3Parameter Property](#) for a list of parameters.

**Return Type** Double

**Default** Not Applicable

Examples `mode = spm4.Stage3ParameterMaximum ("c") 'Read`

**C++ Syntax** HRESULT get\_Stage3ParameterMaximum(BSTR pName, double\* pVal);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage3ParameterMinimum Property

---

**Description** Returns minimum parameter value for the current filter type.

**VB Syntax** *values=* *spm4.Stage3ParameterMinimum* (*parameter*)

**Variable** [\(Type\)](#) - Description

*value* **(Variant)** Variable to store the minimum parameter value.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

*parameter* **(String)** Parameter name. See [Stage3Parameter Property](#) for a list of parameters.

**Return Type** Double

**Default** Not applicable

Examples `mode = spm4.Stage3ParameterMinimum ("c") 'Read`

**C++ Syntax** HRESULT get\_Stage3ParameterMinimum(BSTR pName, double\* pVal);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

1-Jan-2007    MX New topic

## Stage3Parameters Property

---

**Description** Returns the names of parameters for the current filter type. Use [Stage3FilterType Property](#) to set the filter type.

**VB Syntax** *values*= *spm4*.**Stage3Parameters**

**Variable** [\(Type\)](#) - **Description**

*value* **(Variant)** Variable to store the returned parameter names.

*spm4* A [SignalProcessingModuleFour](#) **(object)**

**Return Type** Variant

**Default** Not Applicable

**Examples** `mode = spm4.Stage3Parameters 'Read`

**C++ Syntax** HRESULT get\_Stage3Parameters(VARIANT\* pNames);

**Interface** ISignalProcessingModuleFour

---

Last Modified:

18-Jan-2007    MX New topic

## StandardDeviation Property

---

**Description** Returns the standard deviation of the measurement.  
To retrieve all 3 statistics value at the same time, use [meas.GetTraceStatistics](#)

**VB Syntax** *stdev* = *meas*.**StandardDeviation**

**Variable** [\(Type\)](#) - **Description**

*stdev* **(single)** - Variable to store standard deviation value

*meas* A Measurement **(object)**

**Return Type** Single

**Default** Not applicable

**Examples** `stdev = meas.StandardDeviation 'Read`

**C++ Syntax** HRESULT get\_StandardDeviation(float\* stdDeviation)

**Interface** IMeasurement

**StandardForClass Property - Superseded**

---

**Description** **Superseded** This command sets a **single** standard to a calibration class. Does NOT set or dictate the order for measuring the standards.

Use [GetStandardForClass](#) and [SetStandardForClass](#). These commands allow up to seven standards to be assigned to a cal class.

**VB Syntax** *calKit.StandardForClass(class, portNum) = value*

**Variable** [\(Type\)](#) - Description

*calKit* A CalKit (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*class* (**enum NACalClass**) Standard. Choose from:

1 - naClassA

2 - naClassB

3 - naClassC

4 - naClassD

5 - naClassE

6 - naReferenceRatioLine

7 - naReferenceRatioThru

**SOLT Standards**

1 - naSOLT\_Open

2 - naSOLT\_Short

3 - naSOLT\_Load

4 - naSOLT\_Thru

5 - naSOLT\_Isolation

**TRL Standards**

1 - naTRL\_Reflection

2 - naTRL\_Line\_Reflection

3 - naTRL\_Line\_Tracking

4 - naTRL\_Thru

5 - naTRL\_Isolation



*portNum* **(long)** - The port number the standard will be connected to. For example, you may have a 3.5mm connector designated for port 1, and Type N designated for port 2.

*value* **(long)** - Calibration class number. Choose a number between **1** and **8**. The *<value>* numbers are associated with the following calibration classes:

<i>&lt;value&gt;</i>	<b>Class</b>	<b>Description</b>
1	S11A	Reflection standard
2	S11B	Reflection standard
3	S11C	Reflection standard
4	S21T	Thru standard
5	S22A	Reflection standard
6	S22B	Reflection standard
7	S22C	Reflection standard
8	S21T	Thru standard

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `calKit.StandardForClass(naSOLT_Short, 1) = 1`

`Kclass = calKit.StandardForClass(naSOLT_Short, 1)`

**C++ Syntax** HRESULT put\_StandardForClass (NACalClass item, long pNum);  
HRESULT get\_StandardForClass (NACalClass\* item, long \*pNum);

**Interface** ICalKit

## StartFrequency (Cal Set) Property

---

**Description** Returns the start frequency that is stored in the Cal Set.

**VB Syntax** *value* = *CalSet*.**StartFrequency** (*range*)

**Variable** [\(Type\)](#) - **Description**

*value* **(double)** - returned Start frequency in Hertz.

*CalSet* [CalSet](#) **(object)**

*range* **(Long)** Choose from:  
0 - Source and receiver frequency

**Return Type** Double

**Default** Not Applicable

**Examples** `start = calset.StartFrequency(1) 'Reads the start frequency stored in the cal set.`

**C++ Syntax** HRESULT get\_StartFrequency(double \*pVal)

**Interface** |CalSet3

---

Last modified:

Nov. 1, 2006    New command - split from ch.StartFreq

## StartFrequency Property

---

**Description** Sets or returns the start frequency of the channel.  
**or**  
Sets or returns the start frequency of the segment.  
**or**  
Sets or returns the start frequency of the FOM Range.  
See also [Measurement2](#) interface

**VB Syntax** *object*.**StartFrequency** = *value*

**Variable** [\(Type\)](#) - **Description**

*object* Any of the following:

[Channel](#) (**object**)

[Segment](#) (**object**)

[FOMRange](#) (**object**)

*value* (**double**) - Start frequency in Hertz. Choose any number between the **minimum** and **maximum** frequencies of the analyzer.

**Return  
Type** Double

**Default** Channel - Minimum frequency of the analyzer  
Segment - 0  
FOMRange - Minimum frequency of the analyzer

**Examples** `chan.StartFrequency = 4.5e9 'sets the start frequency of a linear sweep  
for the channel object -Write`

`startfreq = Chan.StartFrequency 'Read`

**C++ Syntax** HRESULT get\_StartFrequency(double \*pVal)  
HRESULT put\_StartFrequency(double newVal)

**Interface** IChannel  
ISegment  
IFOMRange

---

Last modified:

8-Mar-2007 Added FOMRange

Nov. 1, 2006 Removed Cal Set object. There is now a new [cs.StartFreq](#)

## StartPower Property

---

**Description** Sets the start power of the analyzer when [sweep type](#) is set to Power Sweep. Frequency of the measurement is set with chan.[CWFrequency](#).

**VB Syntax** *object.StartPower* = *value*

**Variable** [\(Type\)](#) - **Description**

*object* One of the following:

- [Channel](#) (object)
- [CalSet](#) (object) - Read-only property

*value* **(double)** - Start Power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, use [cap.MaximumSourceALCPower](#) and [cap.MinimumSourceALCPower](#)

Auto attenuation is not allowed in Power Sweep.

**Return Type** Double

**Default** 0

**Examples** `Chan.StartPower = -10 'Write`

`strtpwr = Chan.StartPower 'Read`

**C++ Syntax** HRESULT get\_StartPower(double \*pVal)  
HRESULT put\_StartPower(double newVal)

**Interface** IChannel  
|CalSet3

---

Last Modified:

7-Jan-2008    Removed FOMRange

8-Mar-2007    Added FOMRange

## Start Property

---

**Description** Sets or returns the start time of either Gating or Time Domain transform windows

**VB Syntax** *object*.**Start** = *value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** As Gating  
or  
**(object)** As Transform

*value* **(double)** - Start time in seconds. Choose any number between:  
 $\pm (\text{number of points}-1) / \text{frequency span}$

**Return Type** Double

**Default** -10ns

**Examples**

```
Trans.Start = 4.5e-9 'sets the start time of a transform window
-Write
Gate.Start = 4.5e-9 'sets the start time of a gating window -
Write
```

```
strt = Trans.Start 'Read
```

**C++ Syntax** HRESULT get\_Start(double \*pVal)  
HRESULT put\_Start(double newVal)

**Interface** ITransform  
IGating

Read-only

## Start Property

---

**Description** Returns the stimulus value of the first data point for the measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**Start**

**Variable** [\(Type\)](#) - [Description](#)

*value* **(Double)** - Variable to store the returned value

*meas* A Measurement **(object)**

**Return Type** Double

**Default** Not Applicable

**Examples** `Print meas.Start` 'prints the stimulus value of the first data point

**C++ Syntax** HRESULT get\_Start (double \* Val);

**Interface** IMeasurement2

## StartPowerEx Property

---

**Description** Sets and reads the power sweep start power value for a specific port. This allows uncoupled forward and reverse power sweep ranges. Must also set [SweepType](#) = naPowerSweep, [Coupled](#) = False (Off), and [StopPowerEx](#).

**VB Syntax** *chan.StartPowerEx (srcPort) = value*

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (object)

*srcPort* **(long integer)** – Source port for which to set the Start power value.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* **(double)** - Start Power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, use [cap.MaximumSourceALCPower](#) and [cap.MinimumSourceALCPower](#)

Auto attenuation is not allowed in Power Sweep.

**Return Type** Double

**Default** -10 dBm

**Examples** `Chan.StartPowerEx 1 = -10 'Write`

`strtpwr = Chan.StartPowerEx 2 'Read`

**C++ Syntax** HRESULT get\_StartPowerEx(long port, double \*pVal)  
HRESULT put\_StartPowerEx(long port, double newVal)

**Interface** IChannel13

---

Last Modified:

23-May-2008    MX New topic



## State Property

---

**Description** Turns an Object ON and OFF.

**VB Syntax** *object.State* = *value*

**Variable** [\(Type\)](#) - **Description**

*object* Applies to any of the following objects:

[FOM](#)

[Gating](#)

[InterfaceControl](#)

[LimitTest](#)

Port Extension- Superseded [\(See Fixturing Object\)](#)

[Segment](#)

[Transform](#)

[Equation](#)

[FIFO](#)

### Notes:

- **LimitTest.State** - If using Global Pass/Fail status, trigger the PNA AFTER turning Limit testing ON.
- **Segment.State** - At least ONE segment must be ON or [Sweep Type](#) is automatically set to **Linear**.

*value* **(boolean)** -

**False** - Turns *obj* OFF

**True** - Turns *obj* ON

**Return Type** Boolean

**Default** Depends on the object:

**0** - FOM

**0** - Gating

**0** - InterfaceControl

**0** - LimitTest

**0** - Port Extension

**1** - Segment

**0** - Transform

0 - Equation

0 - FIFO

**Examples**

```
Seg.State = 1 'Turns the segment object ON -Write
```

```
tran = Trans.State 'returns the state of Transform -Read
```

**C++ Syntax**

```
HRESULT get_State(VARIANT_BOOL *pVal)  
HRESULT put_State(VARIANT_BOOL newVal)
```

**Interface**

ISegment  
IInterfaceControl  
ITransform  
IGating  
ILimitTest  
IPortExtension  
IFOM  
IEquation  
IEmbeddedLO  
IFIFO

---

Last Modified:

10-Oct-2008    Added FIFO

13-Apr-2007    Added EmbeddedLO

## State (Rx Leveling) Property

---

**Description** Sets and reads the state of Receiver Leveling for a specific source port.

**VB Syntax** *RxLevel.State(srcPort) = value*

**Variable** [\(Type\)](#) - **Description**

*RxLevel* A [ReceiverLeveling](#) Object

*value* (Boolean) Choose from:

**True** - Receiver leveling ON

**False** - Receiver leveling OFF

*srcPort* (Long Integer) Source port for which to set the state of Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

**Return Type** Variant Boolean

**Default** False

**Examples** `rxLevel.State (1) = True ' Write`

`value = rxLevel.State 2 ' Read`

**C++ Syntax** HRESULT get\_State(long port, VARIANT\_BOOL\* pLevelingState);  
HRESULT put\_State(long port, VARIANT\_BOOL LevelingState);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

12-Feb-2009    MX New topic

## State Property

---

**Description** Turns the specified pulse generator ON and OFF.

**VB Syntax** *pulse.State (n) = value*

**Variable**    **(Type) - Description**

*pulse*    A [PulseGenerator](#) (object)

*n*    **(Integer)** Pulse generator number. Choose from 0 to 4.  
0 is the generator that pulses the ADC.

*value*    **True** - turns pulse output ON.  
**False** - turns pulse output OFF.

**Return Type** Boolean

**Default** False

**Examples** `pulse.State(1) = True 'Write`

`value = pulse.State(4) 'Read`

**C++ Syntax** HRESULT get\_State(VARIANT\_BOOL \*pVal);  
HRESULT put\_State(VARIANT\_BOOL newVal);

**Interface** IPulseGenerator

---

Last Modified:

2-Jan-2007    MX New topic

## Statistics Range Property

---

**Description** Sets the User Range number for calculating measurement statistics. Set the start and stop values for a User Range with [chan.UserRangeMin](#) and [chan.UserRangeMax](#).

There are 16 User Ranges per channel. User ranges are applied independently to any measurement.

**VB Syntax** *meas.StatisticsRange* = *value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*value* (**long integer**) - Range Number. Choose any number between 0 and 16

**0** is Full Span

**1 - 16** are user-defined ranges

**Return Type** Long Integer

**Default** 0

**Examples** `meas.StatisticsRange = 2 'Write`

`statrange = meas.StatisticsRange 'Read`

**C++ Syntax** HRESULT get\_StatisticsRange(long\* rangeNumber)  
HRESULT put\_StatisticsRange(long rangeNumber)

**Interface** IMeasurement

## StatusAsString Property

---

**Description** Returns a string that describes the result of the last tuning sweeps.

**VB Syntax** *value* = *embedLODiag*.**StatusAsString**

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) **(object)**

**Return Type** **(String)**

**Default** Not Applicable

**Examples** `data= embedLO.StatusAsString 'read`

**C++ Syntax** HRESULT get\_StatusAsString(BSTR\* status);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

12-Apr-2007    MX New topic

## StepRiseTime Property

---

**Description** Sets or returns the Rise time of the stimulus in Low Pass Step Mode.

**VB Syntax** *trans*.**StepRiseTime** = *value*

**Variable** [\(Type\)](#) - Description

*trans* A Transform (**object**)

*value* (**double**) - Rise time in seconds. Choose any number between **5.0e-13** and **1.63e-14**.

**Return Type** Double

**Default** 0

**Examples** `trans.StepRiseTime = 1.0e-14 'sets the step rise time to 100 psec. -Write`

`rt = trans.StepRiseTime 'Read`

**C++ Syntax** HRESULT get\_StepRiseTime(double \*pVal)  
HRESULT put\_StepRiseTime(double newVal)

**Interface** ITransform

## StepData Property

---

**Description** Returns an array of data from the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**StepData** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Variant Array)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.StepData 3 'read`

**C++ Syntax** HRESULT StepData(long sweep,VARIANT\* pArray);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

12-Apr-2007    MX New topic



## StepTitle Property

---

**Description** Returns the title of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.**StepTitle** (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.StepTitle 3 'read`

**C++ Syntax** HRESULT StepTitle (long sweep, BSTR \* title);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## StimulusValues Property

---

**Description** Returns the specified X-axis FOM frequency range. The array contains one frequency value for each data point.

**VB Syntax** *value* = *calSet*.**StimulusValues** (*range*)

**Variable** [\(Type\)](#) - [Description](#)

*calSet* [CalSet](#) (**object**) - Read-only property

*range* (**Long**) FOM frequency range to read.

**0** - returns source frequencies.

**1** - returns response frequencies.

**2** - returns primary frequencies.

**Return Type** 1-dimensional variant array

**Default** Not Applicable

**Examples** `array = CalSet.StimulusValues 'Read`

**C++ Syntax** HRESULT get\_StimulusValues (long range, VARIANT\* vals)

**Interface** ICalSet3

---

Last Modified:

16-Sep-2009    Added 2 - primary

19-Oct-2007    Fixed title and syntax

## StopFrequency Property

---

**Description** Sets or returns the stop frequency of the channel  
**or**  
Sets or returns the stop frequency of the segment.  
**or**  
Sets or returns the stop frequency of the FOMRange.  
see also [Measurement2](#) interface

**VB Syntax** *object*.**StopFrequency** = *value*

**Variable** [\(Type\)](#) - Description

*object* Any of the following:

[Channel](#) (**object**)

[Segment](#) (**object**)

[FOMRange](#) (**object**)

*value* (**double**) - Stop frequency in Hertz. Choose any number between the **minimum** and **maximum** frequencies of the analyzer.

**Return  
Type** Double

**Default** Channel - Maximum frequency of the analyzer.  
Segment - 0  
FOMRange - Maximum frequency of the analyzer.

**Examples** `chan.StopFrequency = 4.5e9 'sets the stop frequency for the channel  
object -Write`

`stopfreq = Chan.StopFrequency 'Read`

**C++ Syntax** HRESULT get\_StopFrequency(double \*pVal)  
HRESULT put\_StopFrequency(double newVal)

**Interface** IChannel  
ISegment  
IFOMRange

---

Last modified:

8-Mar-2007 Added FOMRange

Nov. 1, 2006 Removed Cal Set object - created cs.stopfreq

## StopFrequency (Cal Set) Property

---

**Description** Returns the stop frequency that is stored in the Cal Set.

**VB Syntax** *value* = *CalSet*.**StopFrequency** (*range*)

**Variable** [\(Type\)](#) - **Description**

*value* **(double)** - returned Stop frequency in Hertz.

*CalSet* [CalSet](#) **(object)**

*range* **(Long)** Choose from:  
0 - Source and receiver frequency

**Return Type** Double

**Default** Not Applicable

**Examples** `stop = calset.StopFrequency(1)` 'Reads the stop frequency stored in the cal set.

**C++ Syntax** HRESULT get\_StopFrequency(double \*pVal)

**Interface** |CalSet3

---

Last modified:

Nov. 1, 2006    New command - split from ch.StopFreq

## StopPower Property

---

**Description** Sets the Stop Power of the analyzer when [sweep type](#) is set to Power Sweep. Frequency of the measurement is set with chan.[CWFrequency](#).

**VB Syntax** *object*.**StopPower** = *value*

**Variable** [\(Type\)](#) - Description

*object* One of the following:

- [Channel](#) (object)
- [CalSet](#) (object) - Read-only property

*value* **(double)** - Stop Power in dB.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, use [cap.MaximumSourceALCPower](#) and [cap.MinimumSourceALCPower](#)

Auto attenuation is not allowed in Power Sweep.

**Return Type** Double

**Default** 0

**Examples** `Chan.StopPower = -10 'Write`

`stppwr = Chan.StopPower 'Read`

**C++ Syntax** HRESULT get\_StopPower(double \*pVal)  
HRESULT put\_StopPower(double newVal)

**Interface** IChannel  
|CalSet3

---

Last Modified:

7-Jan-2008    Removed FOMRange

8-Mar-2007    Added FOMRange

## Stop Property

---

**Description** Sets or returns the Stop time of either Gating or Time Domain transform windows

**VB Syntax** *object*.**Stop** = *value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** As Gating  
or  
**(object)** As Transform

*value* **(double)** - Start time in seconds. Choose any number between:  
**± (number of points-1) / frequency span**

**Return Type** Double

**Default** 10 ns

**Examples**

```
Trans.Stop = 4.5e-9 'sets the stop time of a transform window -
Write
Gate.Stop = 4.5e-9 'sets the stop time of a gating window -Write

stp = Trans.Stop 'Read
```

**C++ Syntax** HRESULT get\_Stop(double \*pVal)  
HRESULT put\_Stop(double newVal)

**Interface** ITransform  
IGating

## Stop Property

---

**Description** Returns the stimulus value of the last data point for the measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *value* = *meas*.**Stop**

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** Variable to store the returned value

*meas* A Measurement **(object)**

**Return Type** Double

**Default** Not Applicable

**Examples** `Print meas.Stop 'prints the stimulus value of the last data point`

**C++ Syntax** HRESULT get\_Stop(double \* Val);

**Interface** IMeasurement2



## StopPowerEx Property

---

**Description** Sets and reads the power sweep start power value for a specific port. This allows uncoupled forward and reverse power sweep ranges. Must also set [SweepType](#) = naPowerSweep, [Coupled](#) = False (Off), and [StartPowerEx](#).

**VB Syntax** *chan*.StopPowerEx (*srcPort*) = *value*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*srcPort* (long integer) – Source port for which to set the Stop power value.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* (double) - Stop Power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, use [cap.MaximumSourceALCPower](#) and [cap.MinimumSourceALCPower](#)

Auto attenuation is not allowed in Power Sweep.

**Return Type** Double

**Default** 0 dBm

**Examples** `Chan.StopPowerEx 1 = -10 'Write`

`stopPwr = Chan.StopPowerEx 2 'Read`

**C++ Syntax** HRESULT get\_StopPowerEx(long port, double \*pVal)  
HRESULT put\_StopPowerEx(long port, double newVal)

**Interface** IChannel13

---

Last Modified:

3-Jun-2010    MX New topic

## strPort2Pdeembed\_S2PFile Property

---

**Description** Sets and returns the 2 port De-embedding .S2P file name for the specified port number. Model is applied when both the file name is specified and **User** is specified using [Port2PdeembedCktModel Property](#).

[Learn more about S2P files.](#)

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.strPort2Pdeembed\_S2PFile(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive circuit model.

*value* **(String)** Full path, file name, and extension (.s2P) of the de-embedding circuit.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents

**Return  
Type** String

**Default** Not Applicable

**Examples** `fixture.strPort2Pdeembed_S2PFile(2) = "C:/Program Files/Agilent/Network Analyzer/Documents/myFile.s2p" 'Write`

`value = fixture.strPort2Pdeembed_S2PFile(1) 'Read`

**C++ Syntax** HRESULT get\_strPort2Pdeembed\_S2PFile(short port BSTR \*bstrFile)  
HRESULT put\_strPort2Pdeembed\_S2PFile(short port BSTR bstrFile)

**Interface** IFixturing

## strPortMatch\_S2PFile Property

---

**Description** Sets and returns the Port Matching 'S2P' file name for the specified port number. Model is applied when both the file name is specified and **User** is specified using [PortMatchingCktModel Property](#).

[Learn more about S2P files.](#)

**Note:** This command affects ALL measurements on the channel.

**VB Syntax** *fixture.strPort2PMatch\_S2PFile(port) = value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*port* **(Integer)** Port number to receive circuit model.

*value* **(String)** Full path, file name, and extension (.s2P) of the matching circuit.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents".

**Return Type** String

**Default** Not Applicable

**Examples** `fixture.strPort2PMatch_S2PFile(2) = "C:/Program Files/Agilent/Network Analyzer/Documents/myFile.s2p" 'Write`

`value = fixture.strPort2PMatch_S2PFile(1) 'Read`

**C++ Syntax** HRESULT get\_strPort2PMatch\_S2PFile(short port BSTR \*bstrFile)  
HRESULT put\_strPort2PMatch\_S2PFile(short port BSTR bstrFile)

**Interface** IFixturing

## SweepEndMode Property

---

**Description** Sets and reads the event that will cause the Sweep End line to go to a low state. The line will return to a high state after the appropriate calculations are complete.

**Note:** This line is connected to the following pins on the [HANDLER IO connector](#) and [AUX IO connector](#) in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *object.SweepEndMode = value*

**Variable** [\(Type\)](#) - Description

*object* **(object)** - A HandlerIO or AuxIO object

*value* (enum as NASweepEndMode ) Choose from:

- 0 - naSweep - the line goes low when each sweep is complete
- 1 - naChannelSweep - the line goes low when all the sweeps for each channel is complete.
- 2 - naGlobalSweep - the line goes low when all sweeps for all [triggerable](#) channels are complete.

**Return Type** Long Integer

**Default** 0 - naSweep

**Examples** `HWAuxIO.PassFailMode = naSweep 'Write`

`value = HWAuxIO.PassFailMode 'Read`

**C++ Syntax** HRESULT put\_SweepEndMode ( tagNASweepEndMode Mode );  
 HRESULT get\_SweepEndMode ( tagNASweepEndMode\* Mode );

**Interface** IHWAuxIO  
 IHWMaterialHandlerIO

## SweepHoldOff Property

---

**Description** Returns a boolean that represents the state of SweepHoldoff line (pin2) of the External Test Set connector.

**VB Syntax** *value* = *ExtIO*.**SweepHoldOff**

**Variable** [\(Type\)](#) - Description

*value* **(boolean)** - Variable to store the returned data

*ExtIO* **(object)** - An External IO object

**Return Type** Boolean  
**False** - indicates the line is being held at a TTL Low  
**True** - indicates the line is being held at a TTL High

**Default** Not Applicable

**Examples** `value = ExtIO.SweepHoldOff`

**C++ Syntax** HRESULT get\_SweepHoldOff( VARIANT\_BOOL\* bValue);

**Interface** IHWExternaTestSetIO

## SweepOrder Property

---

**Description** Sets and returns the order number of IM products to view when [SweepType](#) = NTH is specified. This actually sets the frequency span to [DeltaF](#) \* N (this command value).

**VB Syntax** *ims.SweepOrder = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Integer) Order number of IM product.

**Return Type** Long Integer

**Default** 9

**Examples** `ims.SweepOrder = 5 'Write`

`value = ims.SweepOrder 'Read`

**C++ Syntax** HRESULT get\_SweepOrder(long \*pVal)  
HRESULT put\_SweepOrder(long newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## SweepSpeedMode Property

---

**Description** Sets and returns the sweep speed mode: FastSweep or Normal.

**VB Syntax** *chan.SweepSpeedMode = value*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*value* (enum as **NASweepSpeedMode**) - Choose from:

**0 - naSweepSpeedModeNormal** - Standard PNA sweep mode

**1 - naSweepSpeedModeFast** - Fast sweep mode

**Return Type** Enum

**Default** 0 - naSweepSpeedModeNormal

**Examples** `chan.SweepSpeedMode = naSweepSpeedModeNormal 'Write`

`swpSpeed = chan.SweepSpeedMode 'Read`

**C++ Syntax** HRESULT get\_SweepSpeedMode(tagNASweepSpeedModes\* pVal)  
HRESULT put\_SweepSpeedMode(tagNASweepSpeedModes newVal)

**Interface** IChannel14

---

Last Modified:

26-Aug-2008    MX New topic

## SweepGenerationMode Property

---

**Description** Sets the method used to generate a sweep: continuous ramp (analog) or discrete steps (stepped).

**VB Syntax** *object*.SweepGenerationMode = *value*

**Variable** [\(Type\)](#) - Description

*object* Channel (**object**)

or

CalSet (**object**) - Read-only property

*value* (**enum NASweepGenerationModes**) - Choose either:

**0 - naSteppedSweep** - source frequency is CONSTANT during measurement of each displayed point. More accurate than Analog. Dwell time can be set in this mode.

**1 - naAnalogSweep** - source frequency is continuously RAMPING during measurement of each displayed point. Faster than Stepped. Sweep time (not dwell time) can be set in this mode.

**Return Type** Long Integer

**Default** Analog

**Examples** `Chan.SweepGenerationMode = naAnalogSweep 'Write`

`swpgen = Chan.SweepGenerationMode 'Read`

**C++ Syntax** HRESULT get\_SweepGenerationMode(tagNASweepGenerationModes\* pVal)  
HRESULT put\_SweepGenerationMode(tagNASweepGenerationModes newVal)

**Interface** IChannel  
|CalSet3



## SweepTime Property

---

**Description** Sets the Sweep time of the analyzer. If sweep time accuracy is critical, use ONLY the values that are attained using the up and down arrows next to the sweep time entry box. [See Sweep Time.](#)

**VB Syntax** *object*.SweepTime = *value*

**Variable** [\(Type\)](#) - Description

*object* [Channel](#) (**object**)

or

[Segment](#) (**object**) first set [SweepTimeOption](#) to true.

or

[CalSet](#) (**object**) - Read-only property

*value* (**double**) - Sweep time in seconds. The maximum sweep time of the PNA is 86400 seconds (1 day).

To set the fastest sweep speed possible, set this value to 0.

**Return  
Type** Double

**Default** 0

**Examples** `chan.SweepTime = 3e-3 'Write`

`swptme = chan.SweepTime 'Read`

**C++ Syntax** HRESULT get\_SweepTime(double \*pVal)  
HRESULT put\_SweepTime(double newVal)

**Interface** IChannel  
CalSet3  
ISegment2

## SweepTimeOption Property

---

**Description** Enables the Sweep time or Dwell time to be set on individual sweep segments. This property must be set True **before** the sweep or dwell time commands are sent. Otherwise, those commands will be ignored.

**VB Syntax** *segs.SweepTimeOption = state*

**Variable** [\(Type\)](#) - [Description](#)

*segs* A [Segments](#) collection (**object**)

*state* (**boolean**)

**True** - Enables Sweep or Dwell time to be set independently.

**False** - Disables Sweep or Dwell time from being set independently.

**Return  
Type** Boolean

**Default** False

**Examples** `segs.SweepTimeOption = True 'Write`

`timeOption = SweepTimeOption 'Read`

**C++ Syntax** HRESULT get\_SweepTimeOption(VARIANT\_BOOL \*pVal)  
HRESULT put\_SweepTimeOption(VARIANT\_BOOL newVal)

**Interface** ISegments3

---

Last modified:

9/29/06    MQQ New command

## SweepType Property (IMD Opt 087)

---

**Description** Sets and returns the type of sweep for a Swept IMD measurement.

See a [list of commands](#) that are relevant for each sweep type.

**VB Syntax** `imd.SweepType = value`

**Variable** [\(Type\)](#) - Description

*imd* A [SweptIMD](#) Object

*value* (Enum as naSweepTypes) - Choose from:

- **0 - naIMDToneCWSweep** The main tone frequencies (F1 and F2) and power levels (P1 and P2) are held constant. Measurements are taken for the specified number of points.
- **1 - naIMDTonePowerSweep** The main tone frequencies are specified as either F1 and F2, or as FC and DeltaF. These frequencies are held constant while the power of each tone is varied from the Start Power to Stop Power
- **2 - naIMDToneCenterFreqSweep** Maintaining a constant tone spacing (DeltaF) and tone powers (P1 and P2), the center frequency (FC) is swept from Start to Stop, or can also be specified as Center and Span.
- **3 - naIMDDeltaFrequencySweep** The center frequency (FC) is held constant. The tone spacing is increased from Start DeltaF to Stop DeltaF.
- **4 - naIMDToneSegmentSweep** Same as FCenter sweep, except that the center frequencies for the sweep are constructed using the standard [segment sweep commands](#). (NOT valid for IMDx)
- **5 - naLOPowerSweep** All frequencies are fixed while the LO power is swept. (IMDx ONLY)

**Return Type** Enum

**Default** 2 - naIMDToneCenterFreqSweep

**Examples** `imd.SweepType = naIMDToneCWSweep 'Write`

```
swtyp = imd.SweepType 'Read
```

**C++ Syntax** HRESULT get\_SweepType(tagNASweepTypes\* pVal)  
HRESULT put\_SweepType(tagNASweepTypes newVal)

**Interface** ISweptIMD

---

Last Modified:

27-Mar-2009 Edited for IMDx

19-Aug-2008 MX New topic

## SweepType Property (IMSpectrum Opt 087)

---

**Description** Sets and returns the type of sweep for an IMSpectrum measurement.

**VB Syntax** *ims.SweepType = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Enum as NAIMSSweepType) - Choose from:

- **0 - naIMSLinearSpan** When Tracking is enabled, allows tuning the Response Settings (receiver) to any values within the frequency range of the PNA. When Tracking is NOT enabled also allows setting the Stimulus (sources) to any values within the frequency range or the PNA.
- **1 - naIMSSecondOrderSpan** The receiver is tuned to view the 2nd order products (f2-f1 and f1+f2) of the main tones that are currently specified in Stimulus Settings. When Tracking is enabled, the main tones are specified in the Swept IMD channel.
- **2 - naIMSThirdOrderSpan** The receiver is tuned to view the 3rd order products (2f1 -f2 and 2f2-f1) of the main tones that are currently specified in Stimulus Settings. When Tracking is enabled, the main tones are specified in the Swept IMD channel.
- **3 - naIMSNthOrderSpan** The frequency range is set to N \* DeltaF. This algorithm will NOT tune the receivers to see the EVEN order products.

**Return Type** Enum

**Default** 3 - naIMSNthOrderSpan

**Examples** `ims.SweepType = naIMSNthOrderSpan 'Write`

`swptyp = ims.SweepType 'Read`

**C++ Syntax** HRESULT get\_SweepType(tagNAIMSSweepType\* pVal)  
HRESULT put\_SweepType(tagNAIMSSweepType newVal)

**Interface** IMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## SweepType Property

---

**Description** Sets and returns the type of sweep. First set SweepType, then set sweep parameters such as frequency or power settings.

**VB Syntax** *object.SweepType* = *value*

**Variable** [\(Type\)](#) - [Description](#)

*object* One of the following:

- [Channel](#) (object)
- [FOMRange](#) (object) Must be an [UNCOUPLD](#) range.
- [CalSet](#) (object) - Read-only property

*value* **(enum NASweepTypes)** - Choose from:

**0 - naLinearSweep**

**1 - naLogSweep**

**2 - naPowerSweep**

**3 - naCWTimeSweep**

**4 - naSegmentSweep**

**Note:** Sweep type cannot be set to Segment sweep if there are no segments turned ON. A segment is automatically turned ON when a application is created.

**Return Type** Long Integer

**Default** naLinearSweep

**Examples** `chan.SweepType = naPowerSweep 'Write`

`swptyp = chan.SweepType 'Read`

**C++ Syntax** HRESULT get\_SweepType(tagNASweepTypes\* pVal)  
HRESULT put\_SweepType(tagNASweepTypes newVal)

**Interface** IChannel  
|CalSet3  
IFOMRange

---

Last Modified:

6-May-2010 Added sweep type note.

8-Mar-2007 Added FOMRange

## SystemImpedanceZ0 Property

---

**Description** Sets and returns the impedance for the analyzer.

**VB Syntax** `app.SystemImpedanceZ0 = value`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (double) Analyzer Impedance. Choose any number between 0 and 1000 ohms.

**Return Type** Double

**Default** 50

**Examples** `app.SystemImpedanceZ0 = 75 'Write`

`z0 = app.SystemImpedanceZ0 'Read`

**C++ Syntax** HRESULT get\_SystemImpedanceZ0(double dSystemZ0)  
HRESULT put\_SystemImpedanceZ0(double \*pdSystemZ0)

**Interface** IApplication



## SystemName Property

---

**Description** Returns the computer name of the PNA.

**VB Syntax** *name* = *app*.**SystemName**

**Variable**    [\(Type\)](#) - Description

*name*        (String) Variable to store the returned computer name.

*app*         An [Application](#) (**object**)

**Return Type**    String

**Default**        Not Applicable

**Examples**       `name = app.SystemName`

**[C++ Syntax](#)**    HRESULT SystemName(BSTR\* computerName)

**Interface**      IApplication

## TargetValue Property

---

**Description** Sets the target value for the marker when doing Target Searches ([SearchTargetLeft](#), [SearchTarget](#), [SearchTargetRight](#)).

**VB Syntax** *mark.TargetValue* = *value*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*value* (**single**) - Target value. Choose any number between: **-500** and **500**

**Return Type** Single

**Default** 0

**Examples**

```
mark.TargetValue = 10.5 'Write
```

```
target = mark.TargetValue 'Read
```

**C++ Syntax** HRESULT get\_TargetValue(float \*pVal)  
HRESULT put\_TargetValue(float newVal)

**Interface** IMarker

## TestPortPower Property

---

**Description** Sets or returns the RF power level for the channel  
**or**  
 Sets or returns the RF power level of the segment.

**VB Syntax** *object*.TestPortPower(*srcPort*) = *value*

### Variable (Type) - Description

*object* A [Channel](#) (object) - to set coupled power, use chan.[CouplePorts](#). If CouplePorts = False, then each port power can be set independently. Otherwise, chanTestPortPower (1) = value sets power level at both ports.

**or**

A [CalSet](#) (object)

**or**

A [Segment](#) (object)

*srcPort* (long integer) - Source Port number.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*value* (double) - RF Power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, use [cap.MaximumSourceALCPower](#) and [cap.MinimumSourceALCPower](#)

Actual achievable leveled power depends on frequency.

**Return Type** Double

**Default** 0

**Examples** `chan.TestPortPower(1) = 5 'sets the port 1 RF power level for the channel object -Write`

`powerlev = Chan.TestPortPower(1) 'Read`

**C++ Syntax** HRESULT get\_TestPortPower(long port, double \*pVal)  
 HRESULT put\_TestPortPower(long port, double newVal)

**Interface** IChannel  
 ICalSet3  
 ISegment

---

Last Modified:

24-Apr-2008 Added note for string names

30-Apr-2007 Edited for src strings

## TestSetType Property

---

**Description** Returns the Test Set Type (model) that was used for the Cal Set.

**VB Syntax** *TSType* = *calset*.**TestSetType**

**Variable** [\(Type\)](#) - **Description**

*TSType* (String) Variable to store the returned test set model.

*calset* A [Cal Set](#) object.

**Return Type** String

**Default** Depends on the test set.

**Example** `TSType = calset.TestSetType`

**C++ Syntax** HRESULT get\_OutputPorts(BSTR \*mapping);

**Interface** ICals5

---

Last modified:

9/18/06 MQ Added for multiport

## Text Property

---

**Description** Specifies an equation or expression to be used on the measurement.

**VB Syntax** *eq*.**Text** = *eqText*

**Variable**    [\(Type\)](#) - Description

*eq*    [Equation](#) (object)

*eqText*    **(String)** - Any valid equation or expression.

**Return Type** String

**Default** Not Applicable

**Examples**    `eq.Text = "foo=S11/S21"`

`equation = eq.Text 'Read`

**C++ Syntax** HRESULT get\_Text(BSTR \*equation)  
HRESULT put\_Text(BSTR equation)

**Interface** IEquation

## ThruCalMethod Property

---

**Description** Sets and returns the method for performing the thru portion of the calibration.

**VB Syntax** *obj*.**ThruCalMethod** = *value*

**Variable** [\(Type\)](#) - **Description**

*obj* [SMCType](#) (object)

or

[VMCType](#) (object)

*value* (String) Specifies the Thru method. Case insensitive - include spaces.  
Choose from:

"Default"

"Flush Thru" **or** "FLUSH"

"Unknown Thru" **or** "UNKN"

"Adapter Removal" **or** "ADAP"

**Return Type** String

**Default** **Default**

**Examples** `SMC.ThruCalMethod = "UNKN"`

**C++ Syntax** HRESULT put\_ThruCalMethod(enum NThruCalMethod thruMethod);  
HRESULT get\_ThruCalMethod(enum NThruCalMethod \*thruMethod);

**Interface** SMCType  
VMCType

## ThruCalMethod Property **Superseded**

---

<b>Description</b>	<p>This command is replaced by PathThruMethod Property .</p> <p>Sets and returns the method for performing the Cal Method and the THRU portion of the calibration.</p>
<b>VB Syntax</b>	<i>guidedCal</i> . <b>ThruCalMethod</b> = <i>value</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>guidedCal</i>	GuidedCalibration (object)
<i>value</i>	<p>(Enum as NATHruCalMethod) Choose from:</p> <p><b>0 - naDefaultCalMethod</b> - allow the PNA to choose the best possible method (from the following) depending on whether the device or ECal module is insertable or non-insertable and given the model number of the PNA. (default selection if omitted.)</p> <p><b>1 - naAdapterRemoval</b> - Perform Adapter removal calibration.</p> <p><b>2 - naFlushThru</b> - Perform Flush Thru calibration.</p> <p><b>3 - naDefinedThru</b> - Perform Defined Thru calibration. If performing an ECal, this is the Thru standard in the ECal module.</p> <p><b>4 - naUnknownThru</b> - Perform Unknown Thru calibration.</p> <p><b>5 - naSOLT</b> - Perform SOLT calibration</p> <p><b>6 - naTRL</b> - Perform TRL calibration</p> <p><b>7 - naQSOLT</b> - Perform QSOLT calibration.</p> <p>Learn more about Cal Methods</p>
<b>Return Type</b>	Enum
<b>Default</b>	0 - naDefaultCalMethod
<b>Examples</b>	<code>guided.ThruCalMethod = naDefinedThru</code>
<b>C++ Syntax</b>	<pre>HRESULT get_ThruCalMethod(enum NATHruCalMethod *thruMethod); HRESULT put_ThruCalMethod(enum NATHruCalMethod thruMethod);</pre>
<b>Interface</b>	IGuidedCalibration

---

Last Modified:

30-Apr-2007    MX Superseded



## ThruPortList Property

---

**Description** **Note:** Available only on PNA releases 5.0 and greater.

Sets and returns the thru connection port pairs for the calibration.

- For 3-port cals, specify at least two pairs.
- For 4-port cals, specify at least three pairs.
- For highest accuracy, specify more than the minimum pairs.
- For a 2-port cal, there is only one port pair. It is the only pair in the list and it is required.

Learn more about [Thru method and port pairings](#).

See an example of a [4-port guided calibration using COM](#).

**VB Syntax** *guidedCal*.ThruPortList = t1a, t1b, t2a, t2b, t3a, t3b

**Variable** [\(Type\) - Description](#)

*guidedCal* [GuidedCalibration](#) (object)

*t1a, t1b...* **(Variant)** Port numbers in pairs - a one-dimensional array of Long integers.

t1a, t1b (Thru1 - port A and port B)

t2a, t2b (Thru2 - port A and port B)

t3a, t3b (Thru3 - port A and port B)

**Return Type** **Variant** - a one-dimensional array of Long integers.

**Default** Not Applicable

**Example** `thruList = Array(1,2,1,3,1,4)`

`guided.ThruPortList = thruList`

'Sets the following three thru connections for a 4-port calibration:

Thru 1 - ports 1 and 2

Thru 2 - ports 1 and 3

Thru 3 - ports 1 and 4

**C++ Syntax** HRESULT get\_ThruPortList(VARIANT\* portList);  
HRESULT put\_ThruPortList(VARIANT portList);

**Interface** IGuidedCalibration



## Title Property

---

**Description** Writes or reads a custom title for the window. Newer entries replace (not append) older entries. Turn the title ON and OFF with [TitleState](#)

**VB Syntax** *win.Title = string*

**Variable** [\(Type\)](#) - Description

*win* A NaWindow (**object**)

*string* (**long**) - Title limited to 50 characters.

**Return  
Type** String

**Default** Null

**Examples** `win.Title = "Hello World" 'Write`

`titl = win.Title 'Read`

**[C++](#) Syntax** HRESULT get\_Title(BSTR \*title)  
HRESULT put\_Title(BSTR title)

**Interface** INAWindow

## TitleState Property

---

**Description** Turns ON and OFF the window title. Write a window title with [Title](#)

**VB Syntax** *win.TitleState = state*

**Variable** [\(Type\)](#) - Description

*win* A NaWindow (**object**)

*state* (**boolean**)

**True** - Title ON

**False** - Title OFF

**Return Type** Boolean

**Default** False

**Examples** `win.TitleState = True 'Write`

`titlestate = win.TitleState 'Read`

**C++ Syntax** HRESULT get\_TitleState(VARIANT\_BOOL\* bState)  
HRESULT put\_TitleState(VARIANT\_BOOL bState)

**Interface** INAWindow

## Tolerance Property

---

**Description** Sets and returns the tolerance value for leveling sweeps.

**VB Syntax** *RxLevel.Tolerance(srcPort) = value*

**Variable** [\(Type\)](#) - [Description](#)

*RxLevel* A [ReceiverLeveling](#) Object

*srcPort* (Long Integer) Source port for which to set the tolerance value for Receiver Leveling.

**Note:** If the source port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-X model, then use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#)

*value* (Double) Tolerance level in dB.

**Return Type** (Double)

**Default** .1 dB

**Examples** `rxLevel.Tolerance (1) = .5 ' Write`

`value = rxLevel.Tolerance 2 ' Read`

**C++ Syntax** HRESULT get\_Tolerance(long port, double\* pVal);  
HRESULT put\_Tolerance(long port, double newVal);

**Interface** IReceiverLevelingConfiguration

---

Last Modified:

13-Feb-2009    MX New topic

## TonePower Property

---

**Description** Sets and returns the power level of the Main Tones. Use with IMD sweep types:

- **naIMDToneCWSweep**,
- **naIMDToneCenterFreqSweep**
- **naIMDDeltaFrequencySweep**.

When tone power is coupled, setting either F1 or F2 power sets both.

**VB Syntax** *object.TonePower (tone) = value*

**Variable** [\(Type\)](#) - [Description](#)

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*tone* (Enum as NAIMDTonePowerID) Choose from:

**0 - naIMDF1Power** - F1 tone

**1 - naIMDF2Power** - F2 tone

*value* (Double) Tone power level in dBm. Choose a value between +30 dBm and -30 dBm.

**Return Type** Double

**Default** -20 dBm

**Examples** `imd.TonePower(naIMDF1Power) = 0 'Write`

`value = imd.TonePower(naIMDF2Power) 'Read`

**C++ Syntax** HRESULT get\_TonePower(tagNAIMDTonePowerID naIMDF1, double \* pVal);  
HRESULT put\_TonePower(tagNAIMDTonePowerID naIMDF1, double newVal);

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TonePowerStart Property

---

**Description** Sets and returns the start power level of the Main tones. Use with IMD sweep type=**naIMDTonePowerSweep**.

When tone power is coupled, setting either F1 or F2 power sets both.

**VB Syntax** *object.TonePowerStart (tone) = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*tone* (Enum as NAIMDTonePowerID) Choose from:

**0 - naIMDF1Power** - F1 tone

**1 - naIMDF2Power** - F2 tone

*value* (Double) Start power in dBm. Choose a value between +30 dBm and -30 dBm.

**Return Type** Double

**Default** -20 dBm

**Examples** `imd.TonePowerStart (naIMDF1Power) = 0 'Write`

`value = imd.TonePowerStart (naIMDF2Power) 'Read`

**C++ Syntax** HRESULT get\_TonePowerStart(tagNAIMDTonePowerID naIMDF1, double \* pVal);  
HRESULT put\_TonePowerStart(tagNAIMDTonePowerID naIMDF1, double newVal);

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TonePowerStop Property

---

**Description** Sets and returns the stop power level of the Main tones. Use with IMD sweep type=**naIMDTonePowerSweep**.

When tone power is coupled, setting either F1 or F2 power sets both.

**VB Syntax** *object.TonePowerStop (tone) = value*

**Variable** [\(Type\)](#) - Description

*object* A [SweptIMD](#) or [IIMSpectrum](#) Object

*tone* (Enum as NAIMDTonePowerID) Choose from:

**0 - naIMDF1Power** - F1 tone

**1 - naIMDF2Power** - F2 tone

*value* (Double) Stop power in dBm. Choose a value between +30 dBm and -30 dBm.

**Return Type** Double

**Default** -20 dBm

**Examples** `imd.TonePowerStop (naIMDF1Power) = 0 'Write`

`value = imd.TonePowerStop (naIMDF2Power) 'Read`

**C++ Syntax** HRESULT get\_TonePowerStop (tagNAIMDTonePowerID naIMDF1, double \* pVal);  
HRESULT put\_TonePowerStop (tagNAIMDTonePowerID naIMDF1, double newVal);

**Interface** ISweptIMD  
IIMSpectrum

---

Last Modified:

19-Aug-2008 MX New topic



## TotalIterations Property

---

**Description** Returns the total number of iterations required by the last SMART sweep. Returns number of power points for a 2D sweep.

**VB Syntax** *iter* = *gca*.**TotalIterations**

**Variable** [\(Type\)](#) - **Description**

*iter* (Integer) Variable to store the returned number of iterations.

*gca* A [GainCompression](#) (**object**)

**Return Type** Integer

**Default** Not Applicable

**Examples** `data = gca.TotalIterations`

**C++ Syntax** HRESULT get\_TotalIterations();

**Interface** IGainCompression2

---

Last Modified:

9-May-2008    MX New topic

## TotalNumberOfPoints Property

---

**Description** Read the total number of points a complete GCA measurement will generate.

- For 2D modes, this is Frequency \* Power points
- For SMART Sweep, this is Frequency points.

The total can NOT exceed the [PNA maximum](#).

See [Frequency](#) and [Power](#) points.

**VB Syntax** *value* = *gca*.TotalNumberOfPoints

**Variable** [\(Type\)](#) - Description

*value* (**integer**) Variable to store the returned total number of points

*gca* A [GainCompression](#) (**object**)

**Return Type** Integer

**Default** 5226 (201 \* 26)

**Example** `totPoints = gca.TotalNumberOfPoints 'Read`

**C++ Syntax** HRESULT get\_TotalNumberOfPoints(int\* pVal)

**Interface** IGainCompression

---

Last Modified:

11-Sep-2007    MX New topic

## Touchscreen Property

---

**Description** Sets and reads the state of the PNA-X Touchscreen (ON and OFF).  
This setting remains until changed again from the front-panel or remote command.

**VB Syntax** *app.Touchscreen = state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (boolean)

**False (0)** - Disables use of Touchscreen

**True (1)** - Enables use of Touchscreen

**Return Type** Boolean

**False** - OFF

**True** - ON

**Default** **TRUE** when shipped from factory.

**Examples** `app.Touchscreen = True 'Write`

`coupl = app.Touchscreen 'Read`

**C++ Syntax** HRESULT put\_Touchscreen(VARIANT\_BOOL bState)  
HRESULT get\_Touchscreen(VARIANT\_BOOL \*bState)

**Interface** IApplication12

---

Last Modified:

23-Feb-2007    MX New topic

## TraceMath Property

---

**Description** Performs math operations on the measurement object and the trace stored in memory. (There MUST be a trace stored in Memory to perform math. See [Meas.DataToMemory](#) method.)

**VB Syntax** *meas.TraceMath = value*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*value* (enum **NAMathOperation**) - Choose from:

- 0** - naDataNormal
- 1** - naDataMinusMemory
- 2** - naDataPlusMemory
- 3** - naDataDivMemory
- 4** - naDataTimesMemory

**Return Type** NAMathOperation

**Default** Normal (0)

**Examples** `meas.TraceMath = naDataMinusMemory 'Write`

`mathOperation = meas.TraceMath 'Read`

**C++ Syntax** HRESULT get\_TraceMath(tagNAMathOperation\* pMathOp)  
HRESULT put\_TraceMath(tagNAMathOperation mathOp)

**Interface** IMeasurement

## TraceMax Property

---

**Description** Maximizes (isolates) or restores the active trace in the active window. When TraceMax is ON, the active trace is the ONLY trace on the display. All other traces are hidden.

**VB Syntax** *meas.TraceMax = state*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*state* (boolean) - Choose from:

**True** - Maximizes / isolates the active trace in the window.

**False** - Restores other traces to be viewed in the window.

**Return  
Type** Boolean

**Default** False

**Examples** `meas.TraceMax = True`

`state = meas.TraceMax`

**C++ Syntax** HRESULT get\_TraceMax(VARIANT\_BOOL bState)  
HRESULT put\_TraceMax(VARIANT\_BOOL\* bState)

**Interface** IMeasurement10

---

Last Modified:

26-Aug-2008    MX New topic

## TraceTitle Property

---

**Description** Writes and reads data for the trace title area.  
The trace title is embedded in the [trace status field](#).  
The title is turned ON and OFF using [TraceTitleState](#).

**VB Syntax** *meas.TraceTitle = value*

**Variable** [\(Type\)](#) - **Description**

*meas* A [Measurement](#) (**object**)

*value* (**string**) - Title to be displayed. Any characters (no spaces), enclosed with quotes.

**Return Type** String

**Default** Not Applicable

**Examples** `meas.TraceTitle = "My new s11 measurement"`

`title = TraceTitle 'Read`

**C++ Syntax** HRESULT get\_TraceTitle(BSTR \*title );  
HRESULT put\_TraceTitle(BSTR title );

**Interface** IMeasurement8

---

Last Modified:

16-Jan-2007    MX New topic

## TraceTitleState Property

---

**Description** Turns display of the Trace Title ON or OFF. When turned OFF, the previous trace title returns. Create a trace title using [TraceTitle Property](#)

**VB Syntax** *meas.TraceTitleState = value*

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*value* (**boolean**) - Choose from:

**True** - Turns the trace title ON

**False** - Turns the trace title OFF

**Return  
Type** Boolean

**Default** False

**Examples** `meas.TraceTitleState = False`

`title = TraceTitleState 'Read`

**C++ Syntax** HRESULT get\_TraceTitleState(VARIANT\_BOOL \*isTitleON);  
HRESULT put\_TraceTitleState(VARIANT\_BOOL isTitleON);

**Interface** IMeasurement8

---

Last Modified:

18-Jun-2007    MX New topic

## Tracking Property

---

**Description** This property, when on, executes the search function ([marker.SearchFunction](#)) every sweep. In effect, turning Tracking ON is the same as executing one of the immediate, one-time, "Search..." methods (such as [SearchMin](#), [SearchMax](#)) for every sweep.

**VB Syntax** *mark.Tracking = state*

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

*state* (**boolean**) - Tracking state. Choose from:

**False** - Tracking OFF

**True** - Tracking ON

**Return Type** Boolean

**Default** False

**Examples** `mark.Tracking = False 'Write`

`markTracking = mark.Type 'Read`

**C++ Syntax** HRESULT put\_Tracking(VARIANT\_BOOL bOn)  
HRESULT get\_Tracking(VARIANT\_BOOL \* pbOn)

**Interface** IMarker



## TrackingChannel Property

---

**Description** Sets and returns the IMD channel number to which the IM Spectrum channel is coupled. Use [TrackingEnable](#) to enable tracking.

**VB Syntax** *ims*.TrackingChannel = *value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Integer) Existing IMD channel number to which frequency and power settings are coupled.

**Return Type** Long Integer

**Default** First existing IMD channel

**Examples** `ims.TrackingChannel = 1 'Write`

`value = ims.TrackingChannel 'Read`

**C++ Syntax** HRESULT get\_TrackingChannel(long \*pVal)  
HRESULT put\_TrackingChannel(long newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TrackingEnable Property

---

**Description** When an IMD channel exists, allows the IM Spectrum frequency and power setting to track (couple with) the IMD channel settings. Use [TrackingChannel](#) to set the channel number to track.

**VB Syntax** *ims.TrackingEnable = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Boolean) Tracking state. Choose from:

**True** - IM Spectrum frequency and power settings track the IMD channel settings.

**False** - IM Spectrum frequency and power settings are specified in the IMS channel.

**Return Type** Boolean

**Default** False

**Examples** `ims.TrackingEnable = True 'Write`

`value = ims.TrackingEnable 'Read`

**C++ Syntax** HRESULT get\_TrackingEnable(VARIANT\_BOOL\* bValue)  
HRESULT put\_TrackingEnable(VARIANT\_BOOL newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TrackingManualStepEnable Property

---

**Description** Sets and returns the step sweep mode for the IM Spectrum channel.

**VB Syntax** *ims.TrackingManualStepEnable = value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Boolean) Choose from:

**False** - Automatic Step

**True** - Manual Step

**Return Type** Boolean

**Default** **False** - Automatic Step

**Examples** `ims.TrackingManualStepEnable = True 'Write`

`value = ims.TrackingManualStepEnable 'Read`

**C++ Syntax** HRESULT get\_TrackingManualStepEnable(VARIANT\_BOOL \*bVal)  
HRESULT put\_TrackingManualStepEnable(VARIANT\_BOOL newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TrackingStepIndex Property

---

**Description** When [TrackingManualStepEnable](#) = True (Manual step), sets and returns the data point number at which the IM spectrum measurement occurs.

**VB Syntax** *ims*.TrackingStepIndex = *value*

**Variable** [\(Type\)](#) - Description

*ims* An [IMSpectrum](#) Object

*value* (Integer) Existing IMD channel number to which frequency and power settings are coupled.

**Return Type** Long Integer

**Default** 1

**Examples** `ims.TrackingStepIndex = 50 'Write`

`value = ims.TrackingStepIndex 'Read`

**C++ Syntax** HRESULT get\_TrackingStepIndex(long \*pVal)  
HRESULT put\_TrackingStepIndex(long newVal)

**Interface** IIMSpectrum

---

Last Modified:

19-Aug-2008    MX New topic

## TriggerDelay Property

---

**Description** Sets and reads the trigger delay for all measurements (GLOBAL). This delay is only applied while in [app.Source = naTriggerSourceExternal](#) and [trigsetup.Scope = naGlobalTrigger](#) . After an external trigger is applied, the start of the sweep is delayed for the specified delay value plus any inherent latency.

To apply a trigger delay for a channel only, use [ExternalTriggerDelay Property](#).

**VB Syntax** `app.TriggerDelay = value`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* **Double**- Trigger delay value in seconds. Range is from 0 to 107

**Return Type** Double

**Default** 0

**Examples** `app.TriggerDelay = .003 'Write`

`delay = app.TriggerDelay 'Read`

**C++ Syntax** HRESULT get\_TriggerDelay(double \*delay);  
HRESULT put\_TriggerDelay(double delay)

**Interface** IApplication

## TriggerInPolarity Property

---

**Description** Specifies the polarity of the trigger IN signal.

- **AuxTrigger Object** - Sets the polarity to which the rear-panel AuxTrig IN responds.
- **PulseGenerator Object** - Sets the polarity of trigger to which the internal pulse generators will respond when being externally triggered. **Note:** This feature requires DSP version: **4.0 FPGA:34** or higher. [Learn more](#).

**Note:** Used on PNA-X ONLY.

**VB Syntax** *object.TriggerInPolarity = value*

**Variable** [\(Type\)](#) - Description

*object* An [AuxTrigger](#) (object) or  
A [PulseGenerator](#) (object)

*value* (enum NATriggerPolarity) - Choose from:  
**naTriggerPositive** PNA responds to rising edge or HIGH level  
**naTriggerNegative** PNA responds to falling edge or LOW level.  
Set Edge or Level triggering using [TriggerInType Property](#)

**Return Type** Enum

**Default** AuxTriggerIn Object - naTriggerNegative  
PulseGenerator Object - naTriggerPositive. Also the polarity used when the PNA-X does not have the required DSP hardware

**Exaamples** `obj.TriggerInPolarity = naTriggerPositive 'Write`

`value = obj.TriggerInPolarity 'Read the value`

**C++ Syntax** HRESULT get\_TriggerInPolarity(enum NATriggerPolarity \*val);  
HRESULT put\_TriggerInPolarity(enum NATriggerPolarity val);

**Interface** IAuxTrigger  
IPulseGenerator3

Last Modified:

9-Dec-2009	Added PulseGenerator Object
5-Sep-2008	Added Note
14-Dec-2006	MX New topic

## TriggerInType Property

---

**Description** Specifies the type of trigger input being supplied to the PNA.

- **AuxTrigger Object** - Sets the type to which the rear-panel AuxTrig IN responds.
- **PulseGenerator Object** - Sets the type of trigger to which the internal pulse generators will respond when being externally triggered. **Note:** This feature requires DSP version: **4.0 FPGA:34** or higher. [Learn more](#).

**Note:** Use on PNA-X ONLY.

**VB Syntax** *obj.TriggerInType = value*

**Variable** [\(Type\)](#) - Description

*obj* An [AuxTrigger](#) (object) **or**  
A [PulseGenerator](#) (object)

*value* (enum NATriggerSignalType) Choose from:  
**naTriggerEdge** PNA responds to the edge (rising or falling) of a signal  
**naTriggerLevel** PNA responds to the level (HIGH or LOW) of a signal  
Use [TriggerInPolarity](#) to set Positive or Negative polarity.

**Return Type** Enum

**Default** **naTriggerLevel** - Also the type used for the PulseGenerator Object when the PNA-X does not have the required DSP hardware

**Exaamples** `obj.TriggerInType = naTriggerEdge 'Write`

`value = obj.TriggerInType 'Read the value`

**C++ Syntax** HRESULT get\_TriggerInType(enum NATriggerSignalType \*val);  
HRESULT put\_TriggerInType(enum NATriggerSignalType val);

**Interface** IAuxTrigger  
IPulseGenerator3

---

Last Modified:



9-Dec-2009	Added PulseGenerator obj.
5-Sep-2008	Added Note
14-Dec-2006	MX New topic

## TriggerOutDuration Property

---

**Description** Specifies the width of the output pulse, which is the time that the Aux trigger output will be asserted.

**VB Syntax** *auxTrig.TriggerOutDuration* = *value*

**Variable** [\(Type\)](#) - **Description**

*auxTrig* An [AuxTrigger](#) (**object**)

*value* (**single**) - Duration value in seconds. Choose a value between 1E-6 and 1.

**Return Type** Double

**Default** 1E-6 sec

**Exaamples** `auxTrig.TriggerOutDuration = 1e-3 'Write`

`value = auxTrig.TriggerOutDuration 'Read the value`

**C++ Syntax** HRESULT get\_TriggerOutDuration(double \*val);  
HRESULT put\_TriggerOutDuration(double val);

**Interface** IAuxTrigger

---

Last Modified:

14-Dec-2006    MX New topic

## TriggerOutInterval Property

---

**Description** Specifies how often a trigger output signal is sent.

**VB Syntax** *auxTrig.TriggerOutInterval = value*

**Variable** [\(Type\)](#) - Description

*auxTrig* An [AuxTrigger](#) (object)

*value* **(single)** - Choose from:

**0 - naTriggerModePoint** - a single data point is measured with each trigger signal the channel receives. Subsequent trigger signals continue to go to the channel in Point mode until the channel measurements are complete. This is effectively the same as [trigger point mode](#).

**1 - naTriggerModeMeasurement** - entire traces are swept with a trigger signal. which and how many traces depends on the Scope setting.

**Return Type** Enum

**Default** 1 - naTriggerModeMeasurement

**Exaamples** `auxTrig.TriggerOutInterval = naTriggerModeMeasurement 'Write`

`value = auxTrig.TriggerOutInterval 'Read the value`

**C++ Syntax** HRESULT get\_TriggerOutInterval(enum NATriggerMode \*val);  
HRESULT put\_TriggerOutInterval(enum NATriggerMode val);

**Interface** IAuxTrigger

---

Last Modified:

14-Dec-2006    MX New topic

## TriggerOutPolarity Property

---

**Description** Specifies the polarity of the trigger output signal being supplied by the PNA.

**VB Syntax** *auxTrig.TriggerOutPolarity = value*

**Variable** [\(Type\)](#) - Description

*auxTrig* An [AuxTrigger](#) (object)

*value* (enum NATriggerPolarity) - Choose from:

**naTriggerPositive** PNA sends positive going (active HIGH) pulse.

**naTriggerNegative** PNA sends negative going (active LOW) pulse.

**Return Type** Enum

**Default**

**Exaamples** `auxTrig.TriggerOutPolarity = naTriggerPositive 'Write`

`value = auxTrig.TriggerOutPolarity 'Read the value`

**C++ Syntax** HRESULT get\_TriggerOutPolarity(enum NATriggerPolarity \*val);  
HRESULT put\_TriggerOutPolarity(enum NATriggerPolarity val);

**Interface** IAuxTrigger

---

Last Modified:

14-Dec-2006    MX New topic

## TriggerOutPosition Property

---

**Description** Specifies whether the Aux trigger out signal is sent Before or After the acquisition.

**VB Syntax** *auxTrig.TriggerOutPosition = value*

**Variable** [\(Type\)](#) - Description

*auxTrig* An [AuxTrigger](#) (object)

*value* (enum NATriggerPosition) Choose from:

**naTriggerOutBeforeAcquire** Use if the external device needs to be triggered before the data is acquired, such as a power meter.

**naTriggerOutAfterAcquire** Use if the external device needs to be triggered just after data has been acquired, such as an external source. This could be more efficient since it allows the external device to get ready for the next acquisition at the same time as the PNA.

**Return  
Type** Enum

**Default** **naTriggerOutAfterAcquire**

**Exaamples** `auxTrig.TriggerOutPosition = naTriggerOutAfterAcquire 'Write`

`value = auxTrig.TriggerOutPosition 'Read the value`

**[C++ Syntax](#)** HRESULT get\_TriggerOutPosition(enum NATriggerPosition \*val);  
HRESULT put\_TriggerOutPosition(NATriggerPosition val);

**Interface** IAuxTrigger

---

Last Modified:

14-Dec-2006    MX New topic

## TriggerOutputEnabled Property

---

**Description** Enables the PNA to send trigger signals out the [rear-panel TRIGGER OUT](#) connector. For more information, see [External triggering](#).

**VB Syntax** *trigsetup.TriggerOutputEnabled = boolean*

**Variable** [\(Type\)](#) - Description

*trigsetup* A [TriggerSetup2](#) (object)

*boolean* Choose from:

**False** - PNA does NOT send output trigger signals.

**True** - PNA sends output trigger signals.

**Return  
Type** Boolean

**Default** False

**Examples** `trigsetup.TriggerOutputEnabled = True 'Write`

`atba = trigsetup.TriggerOutputEnabled 'Read`

**C++ Syntax** HRESULT get\_TriggerOutputEnabled( BOOL \*pVal);  
HRESULT put\_TriggerOutputEnabled( BOOL newVal);

**Interface** ITriggerSetup2

## TuningIFBW Property

---

**Description** Set the IF Bandwidth for Broadband and Precise tuning sweeps.

**VB Syntax** *obj.TuningIFBW = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* (Double) IF Bandwidth

**Return Type** (Double)

**Default** 30 kHz

**Examples** `embedLO.TuningIFBW = 10e3 'write`

`value = embedLO.TuningIFBW 'read`

**C++ Syntax** HRESULT get\_TuningIFBW(double\* ifbw);  
HRESULT put\_TuningIFBW(double ifbw);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## TuningMode Property

---

**Description** Sets and returns the method used to determine the embedded LO Frequency.

**VB Syntax** *obj.TuningMode = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* **(Enum as NAEmbeddedLOTuningMode)**

Tuning mode. Choose from:

0 - naEmbeddedLOTuningMode\_Broadband\_And\_Precise

1 - naEmbeddedLOTuningMode\_Precise\_Only

2 - naEmbeddedLOTuningMode\_None

**Return Type** **(Enum)**

**Default** 0 - naEmbeddedLOTuningMode\_Broadband\_And\_Precise

**Examples** `embedLO.TuningMode = naEmbeddedLOTuningMode_None 'write`

```
value = embedLO.TuningMode 'read
```

**C++ Syntax** HRESULT get\_TuningMode(enum NAEmbeddedLOTuningMode\* mode);  
HRESULT put\_TuningMode(enum NAEmbeddedLOTuningMode mode);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic



## TuningSweepInterval Property

---

**Description** Set how often a tuning sweep is performed.

**VB Syntax** *obj.TuningSweepInterval = value*

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (object) or  
A [ConverterEmbeddedLO](#) (object)

*value* (Long) Tuning sweep interval.

**Return Type** (Long)

**Default** 1

**Examples** `embedLO.TuningSweepInterval = 3 'write .. tuning is performed  
every third measurement sweep`

`value = embedLO.TuningSweepInterval 'read`

**C++ Syntax** HRESULT get\_TuningSweepInterval(long\* interval);  
HRESULT put\_TuningSweepInterval(long interval);

**Interface** IEmbeddedLO

---

Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## TriggerMode Property

---

**Description** These settings determine what EACH signal will trigger.

**Note:** Setting Point and EverySweep mode forces [Trigger.Scope](#) = naChannelTrigger.

**VB Syntax** *chan.TriggerMode = value*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*value* (enum NATriggerMode) - Choose from:

**0 - naTriggerModePoint** - Each Manual or External trigger signal causes one data point to be measured.

**1 - naTriggerModeMeasurement** (superseded - still works but replaced with a more descriptive enum)

**1 - naTriggerModeChannel** - Each trigger signal causes **ALL traces** in that channel to be swept.

**2 - naTriggerModeEverySweep** - Each Manual or External trigger signal causes **ALL traces that share a source port** to be swept.

**3 - naTriggerModeTrace** - Allowed ONLY when [PointSweepState](#) is enabled. Each trigger signal causes two identical measurements to be triggered separately - one trigger signal is required for each measurement. Other trigger mode settings cause two identical parameters to be measured simultaneously.

**Return Type** Long Integer

**Default** 1 - naTriggerModeChannel

**Examples** `chan.TriggerMode = naTriggerModePoint 'Write`

`trigtyp = chan.TriggerMode 'Read`

**C++ Syntax** HRESULT get\_TriggerMode (tagNATriggerMode \*pMode)  
HRESULT put\_TriggerMode (tagNATriggerMode newMode)

**Interface** IChannel

---

Last Modified:

6-Nov-2007    Added new sweep mode

## TriggerSignal Property - Superseded

---

**Description** **Note:** This command has been replaced by [Source](#) Property

Sets or returns the trigger source.

**VB Syntax** `app.TriggerSignal = value`

**Variable** [\(Type\)](#) - Description

*app* An Application (**object**)

*value* (**enum NATriggerSignal**) - Choose from:

**0 - naTriggerInternal** - free run

**1 - naTriggerExternalPositive** - a trigger signal is generated when a TTL high is sensed on the [external trigger pin](#) of the Aux IO connector

**2 - naTriggerExternalNegative** - a trigger signal is generated when a TTL low is sensed on the [external trigger pin](#) of the Aux IO connector.

**3 - naTriggerManual** - manual trigger source; use `app.ManualTrigger` to send a trigger signal.

**4 - naTriggerExternalHigh** - a trigger signal is generated when a TTL high is sensed on the [external trigger pin](#) of the Aux IO connector

**5 - naTriggerExternalLow** - a trigger signal is generated when a TTL low is sensed on the [external trigger pin](#) of the Aux IO connector

**Return Type** Long Integer

**Default** naTriggerInternal

**Examples** `app.TriggerSignal = naTriggerExternalPositive 'Write`

`trigsign = app.TriggerSignal 'Read`

**C++ Syntax** HRESULT get\_TriggerSignal(tagNATriggerSignal \*pSignal)  
HRESULT put\_TriggerSignal(tagNATriggerSignal signal)

**Interface** IApplication

## TriggerType Property - Superseded

---

**Description** **Note:** This property has been replaced with [Scope](#) Property.

Sets or returns the trigger type which determines the scope of a trigger signal.

**VB Syntax** `app.TriggerType = value`

**Variable** [\(Type\)](#) - Description

*app* An Application (**object**)

*value* (**enum NATriggerType**) - Trigger type. Choose from:

**0 - naGlobalTrigger** - a trigger signal is applied to all triggerable channels

**1 - naChannelTrigger** - a trigger signal is applied to the current channel. The next trigger signal will be applied to the next channel; not necessarily channel 1-2-3-4.

**Return Type** Long Integer

**Default** naGlobalTrigger

**Examples** `app.TriggerType = naGlobalTrigger 'Write`

`trigtyp = app.TriggerType 'Read`

**C++ Syntax** HRESULT get\_TriggerType(tagNATriggerType \*pTrigger)  
HRESULT put\_TriggerType(tagNATriggerType trigger)

**Interface** IApplication

## Type (calstd) Property

---

**Description** Sets and Returns the type of calibration standard.

**VB Syntax** *calstd.Type* = *value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**enum** NACalStandardType) -Choose from:  
**0 - naOpen**  
**1 - naShort**  
**2 - naLoad**  
**3 - naThru**

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `calstd.Type = naOpen 'Write`

`standardtype = calstd.Type 'Read`

**C++ Syntax** HRESULT get\_Type(tagNACalStandardType \*pVal)  
HRESULT put\_Type(tagNACalStandardType newVal)

**Interface** ICalStandard

## Type (testset) Property

---

**Description** Returns the testset model number.

**VB Syntax** *tset.Type* *model*

**Variable** [\(Type\)](#) - Description

*model* **(String)** Variable to return the Test set model

*tset* A [TestsetControl](#) object.  
Obtained from the [ExternalTestsets](#) collection.

**Return  
Type** String

**Default** None

**Examples** `testset.type` `model`

[See External Testset Program](#)

**C++ Syntax** HRESULT get\_Type(BSTR \*pType);

**Interface** ITestsetControl

## TZImag Property

---

**Description** Sets and Returns the TZImag value (the Imaginary Terminal Impedance value) for the calibration standard. Only applicable when "[Type](#)" is set to **naArbitraryImpedance**.

To set the other resistance values, use [TZReal](#)

**VB Syntax** *calstd.TZImag = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for TZImag in Ohms

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.TZImag = 15 'Write the value of TZImag to 15 Ohms`

`imp0 = calstd.TZImag 'Read the value of TZImag`

**C++ Syntax** HRESULT get\_TZImag(float \*pVal);  
HRESULT put\_TZImag(float newVal);

**Interface** ICalStandard2

## TZReal Property

---

**Description** Sets and Returns the TZReal value (the real Terminal Impedance value) for the calibration standard. Only applicable when "[Type](#)" is set to **naArbitraryImpedance**.

To set the other resistance values, use [TZImag](#)

**VB Syntax** *calstd.TZReal = value*

**Variable** [\(Type\)](#) - Description

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) - Value for TZReal in Ohms

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.TZReal = 15 'Write the value of TZReal to 15 Ohms`

`imp0 = calstd.TZReal 'Read the value of TZReal`

**C++ Syntax** HRESULT get\_TZReal(float \*pVal);  
HRESULT put\_TZReal(float newVal);

**Interface** ICalStandard2



Read-only

## UnusedChannelNumbers Property

---

**Description** Returns an array of channel numbers that are NOT in use. An unused channel has NO measurements subscribed to it.

**VB Syntax** *chanNumbers = chans.UnusedChannelNumbers (NumberOfChannels)*

**Variable** [\(Type\)](#) - **Description**

*chanNumbers* Variable array to store the returned channel numbers

*chans* A [Channel collection](#) (**object**)

*NumberOfChannels* (**Long Integer**) Number of channels that you are requesting.

**Return Type** One-dimensional array of long integers. The size of the array is specified by the *NumberOfChannels* parameter.

**Default** Not Applicable

**Examples** `chanNumbers = chans.UnusedChannelNumbers (5)`

**C++ Syntax** HRESULT get\_UnusedChannelNumbers(long numberRequested,VARIANT\* channelNumbers);

**Interface** IChannels2

## USBPowerMeterCatalog Property

---

**Description** Returns the ID string of power meters / sensors that are connected to the PNA USB. Use the list to select a power sensor for a source power cal.

**VB Syntax** *list* = *pwrCal*.**USBPowerMeterCatalog**

**Variable** [\(Type\)](#) - Description

*list* **(String)** Variable to store the returned list of USB power meters.

*pwrCal* **(object)** – [A SourcePowerCalibrator](#) (object)

**Return Type** Comma-delimited strings. Two sensor strings are separated by a semicolon.

**Default** Not Applicable

**Examples**

```
Set pwrCal = pna.SourcePowerCalibrator
list = pwrCal.USBPowerMeterCatalog'Read
```

**C++ Syntax** HRESULT get\_USBPowerMeterCatalog(BSTR \*pUSBList);

**Interface** ISourcePowerCalibrator6

---

### Last Modified:

24-Feb-2009 Updated with semicolon

11-Jul-2007 MX New topic

Read/Write

## UseCalWindow Property

---

**Description** Turns Calibration window ON or OFF during a calibration. Learn more about this preference.

**VB Syntax** *guidedCal.UseCalWindow = value*

**Variable** [\(Type\) - Description](#)

*guidedCal* [GuidedCalibration](#) (object)

*value* **(Boolean)**

**True** Show calibration window

**False** Hide calibration window

**Return Type** **Boolean**

**Default** True

**Example** `guided.UseCalWindow = True`

**C++ Syntax** HRESULT get\_UseCalWindow(VARIANT\_BOOL\* val);  
HRESULT put\_UseCalWindow(VARIANT\_BOOL newVal);

**Interface** IGuidedCalibration

---

Last Modified:

12-Sep-2007    MX New topic

Read-only

## UsedChannelNumbers Property

---

**Description** Returns an array of channel numbers that are in use. A used channel has at least one measurement subscribed to it

**VB Syntax** *chanNumbers* = *chans*.UsedChannelNumbers

**Variable** [\(Type\)](#) - **Description**

*chanNumbers* Variable array to store the returned channel numbers

*chans* A [Channel collection](#) (**object**)

**Return Type** One-dimensional array of long integers

**Default** Not Applicable

**Examples** `chanNumbers = chans.UsedChannelNumbers`

**C++ Syntax** HRESULT get\_UsedChannelNumbers(VARIANT\* channelNumbers);

**Interface** IChannels2

## UsePowerLossSegments Property

---

**Description** Specifies if subsequent power readings will use of the loss table.  
(PowerLossSegments).

**VB Syntax** *pwrCal*.**UsePowerLossSegments** = *value*

**Variable** [\(Type\)](#) - Description

*pwrCal* A [SourcePowerCalibrator](#) (object)

A [PowerSensorAsReceiver](#) (Object)

*value* **(boolean)**

**False** – Do not use loss table

**True** – Use loss table

**Return Type** Boolean

**Default** False

**Examples**

```
pwrSens.UsePowerLossSegments = True 'Write  
lossTableState = pwrSens.UsePowerLossSegments 'Read
```

**C++ Syntax** HRESULT put\_UsePowerLossSegments(VARIANT\_BOOL bState);  
HRESULT get\_UsePowerLossSegments(VARIANT\_BOOL \*bState);

**Interface** ISourcePowerCalibrator  
IPowerSensorAsReceiver

---

Last Modified:

25-Aug-2009    MX New topic

## UsePowerSensorFrequencyLimits Property

---

**Description** Specifies if subsequent calls to the [AcquirePowerReadings](#) method will observe frequency values of the [MinimumFrequency](#) and [MaximumFrequency](#) properties.

**VB Syntax** *pwrCal.UsePowerSensorFrequencyLimits = value*

**Variable** [\(Type\)](#) - **Description**

*pwrCal* **(object)** – A SourcePowerCalibrator (object)

*value* **(boolean)** -

**False** – Do not use power sensor frequency limits. An acquisition will use just one power sensor for the entire sweep, regardless of frequency.

**True** – Use power sensor frequency limits. A requested acquisition will only succeed for those frequency points which fall between the MinimumFrequency and MaximumFrequency values of that PowerSensor. An acquisition will pause in mid-sweep if the frequency is about to exceed the MaximumFrequency value. When the sweep is paused in this manner, a sensor connected to the other channel input of the power meter can be connected to the measurement port in place of the previous sensor, and then the sweep completed by another call to AcquirePowerReadings. However, the MaximumFrequency specified for the second sensor would need to be sufficient for the sweep to complete.

**Return Type** Boolean

**Default** False

**Examples**

```
Set powerCalibrator = pna.SourcePowerCalibrator
powerCalibrator.UsePowerSensorFrequencyLimits = True 'Write
FreqCheck = powerCalibrator.UsePowerSensorFrequencyLimits 'Read
```

**C++ Syntax** HRESULT put\_UsePowerSensorFrequencyLimits(VARIANT\_BOOL bState);  
 HRESULT get\_UsePowerSensorFrequencyLimits(VARIANT\_BOOL \*bState);

**Interface** ISourcePowerCalibrator

## UserRange Property

---

**Description** Assigns the marker to the specified User Range. This restricts the marker's x-axis travel to the User Range span, specified with [Start](#) and [Stop](#) values.

- Each channel has **16** user ranges.
- Markers and trace statistics can be restricted to any user range.
- More than one marker can occupy a user range.
- User ranges can overlap. For example:
  - User range 1: 3 GHz to 5 GHz
  - User range 2: 4 GHz to 6 GHz

**Note:** User ranges are especially useful in restricting marker searches to specific areas of the measurement.

**VB Syntax** *mark*.**UserRange** = *value*

**Variable** [\(Type\)](#) - **Description**

*mark* A Marker (**object**)

*value* (**long integer**) - User Range. Choose any number between 0 and 16 (0=Full Span)

**Return Type** Long Integer

**Default** 0 - Full Span

**Examples** `mark.UserRange = 1 'Write`

`UseRnge = mark.UserRange 'Read`

**C++ Syntax** HRESULT get\_UserRange(long \*pRangeNumber)  
HRESULT put\_UserRange(long IRangeNumber)

**Interface** IMarker

## UserRangeMax Property

---

**Description** Sets the stimulus stop value for the specified User Range.  
This property uses different arguments for the channel and marker objects.

**VB Syntax** *chan.UserRangeMax(domainType,Mnum) = value*  
**or**  
*mark.UserRangeMax(mnum) = value*

**Variable** [\(Type\) - Description](#)

*chan* A Channel (**object**)

*mark* A Marker (**object**)  
To assign a marker to a User Range, use the [UserRange Property](#).

**Note:** The Marker object does not require the "domainType" argument.

*domainType* (**enum NADomainType**) - Choose from:

**0 - naDomainFrequency**

**1 - naDomainTime**

**2 - naDomainPower**

*Mnum* (**long integer**) - User Range number. Choose any number between 1 and 16 (0=Full Span)

*value* (**double**) - Stop value. Choose any number within the full span of the channel

**Return Type** Double

**Default** The current stimulus setting for the channel

**Examples**

```
mark.UserRangeMax(1) = 3e9 'Write
chan.UserRangeMax(naDomainFrequency,1) = 3e9 'Write
```

```
UserRngeMax = mark.UserRangeMax 'Read
UserRngeMax = chan.UserRangeMax 'Read
```

**C++ Syntax** HRESULT put\_UserRangeMax(tagNADomainType domain, long rangeNumber, double maxValu  
HRESULT get\_UserRangeMax(tagNADomainType domain, long rangeNumber, double  
\*maxValue)

**Interface** IChannel



## UserRangeMin Property

---

**Description** Sets the stimulus start value for the specified User Range.  
This property uses different arguments for the channel and marker objects.

**VB Syntax** *chan.UserRangeMin(domainType,range) = value*  
**or**  
*mark.UserRangeMin(range) = value*

**Variable** [\(Type\) - Description](#)

*chan* A Channel (**object**)

*mark* A Marker (**object**)  
To assign a marker to a User Range, use the [UserRange Property](#).

**Note:** The Marker object does not require the domainType argument

*domainType* (**enum NADomainType**) Type of sweep currently implemented on the channel - Choose from:  
**0 - naDomainFrequency**  
**1 - naDomainTime**  
**2 - naDomainPower**

*range* (**long**) - User Range number. Choose any number between **1** and **16** (0=Full Span)

*value* (**double**) - Start value. Choose any number within the full span of the analyzer

**Return Type** Double

**Default** The current stimulus setting for the channel

**Examples**

```
mark.UserRangeMin(1) = 3e9 'Write
chan.UserRangeMin(naDomainFrequency,1) = 3e9 'Write

UseRngeMin = mark.UserRangeMin 'Read
UseRngeMin = chan.UserRangeMin 'Read
```

**C++ Syntax** HRESULT put\_UserRangeMin(tagNADomainType domain, long rangeNumber, double minValue)  
HRESULT get\_UserRangeMin(tagNADomainType domain, long rangeNumber, double \*minValue)

**Interface** IChannel

## UserDescriptionOfPNA Property

---

**Description** Sets and reads a description of the PNA used to perform the User Characterization. This description is stored with the characterization in the ECal module.

Set this description before sending [Initialize](#) or the default (empty string) will be used.

**VB Syntax** *userChar*.**UserDescriptionOfPNA** = *value*

**Variable** [\(Type\)](#) - **Description**

*userChar* An [ECalUserCharacterizer](#) Object

*value* (String) Descriptive text, limited to 14 characters maximum.

**Return Type** String

**Default** ""(Empty String)

**Examples** `userChar.UserDescriptionOfPNA = "My PNA"`

**C++ Syntax** HRESULT get\_UserDescriptionOfPNA(BSTR \*info);  
HRESULT put\_UserDescriptionOfPNA(BSTR info);

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## UserName Property

---

**Description** Sets and reads the description of the person and/or company who is producing the ECal user characterization. This description is stored with the characterization in the ECal module.

Set this description before sending [Initialize](#) or the default (empty string) will be used.

**VB Syntax** *userChar.UserName = value*

**Variable** [\(Type\)](#) - Description

*userChar* An [ECalUserCharacterizer](#) Object

*value* (String) Descriptive text, limited to 19 characters maximum.

**Return Type** String

**Default** "" (Empty String)

**Examples** `userChar.UserName = "John Doe, Acme Inc."`

**C++ Syntax** HRESULT get\_UserName(BSTR \*name);  
HRESULT put\_UserName(BSTR name);

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## UserPresetEnable Property

---

**Description** 'Checks' and 'clears' the enable box on the User Preset dialog box. This only affects subsequent Presets from the front panel user interface.

Regardless of the state of the User Preset Enable checkbox, the [app.Preset](#) command will always preset the PNA to the factory preset settings, and [app.UserPreset](#) will always perform a User Preset.

**VB Syntax** `app.UserPresetEnable = state`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (**boolean**) Front Panel User Preset State. Choose from:  
False – User Preset OFF  
True – User Preset ON

**Return Type** Boolean

**Default** False

**Examples** `app.UserPresetEnable = True 'Write`

`upreset = app.UserPresetEnable 'Read`

**C++ Syntax** HRESULT get\_UserPresetEnable(VARIANT\_BOOL \*pVal)  
HRESULT put\_UserPresetEnable(VARIANT\_BOOL newVal)

**Interface** IApplication6

## Valid Property

---

**Description** Returns a boolean value to indicate if the current equation on the measurement is valid. For equation processing to occur, the equation must be valid and ON.

**VB Syntax** *IsValid* = *eq.Valid*

**Variable** [\(Type\)](#) - Description

*good* **(Boolean)** Variable to store the returned value.

**True** (1) - equation is valid

**False** (0) - equation is NOT valid

*eq* [Equation](#) **(object)**

**Return Type** Boolean

**Default** Not Applicable

**Examples** `IsValid = eq.Valid 'Read`

**C++ Syntax** HRESULT get\_Valid(Boolean \*equation)

**Interface** IEquation

---

Last Modified:

18-Jun-2007    New topic

Read only

## ValidConnectorTypes Property

---

**Description** Returns a list of all connector types for which there are calibration kits. Looks for connector types in mechanical cal kits, within PNA disk memory, and within the attached ECal Module memory.

**VB Syntax** *value* = *obj*.ValidConnectorTypes

**Variable** [\(Type\)](#) - [Description](#)

*value* (Variant) List of connector types

*obj* Any of the following:  
[ECalUserCharacterizer](#) (object)  
[GuidedCalibration](#) (object)  
[SMCType](#) (object)  
[VMCType](#) (object)

**Return Type** Variant

**Default** Not Applicable

**Examples** `value = SMC.ValidConnectorTypes`

**C++ Syntax** HRESULT get\_ValidConnectorTypes(VARIANT\* connectorTypes);

**Interface** IGuidedCalibration  
SMCType  
VMCType

---

Last Modified:

1-Sep-2009 Added ECal User and within disk memory (A.09.00)

## Value Property

---

**Description** Write or read a value (setting) for the current element.

See a [list of configurable elements and settings](#) for various PNA models.

**VB Syntax** *pathElement.Value = value*

**Variable** [\(Type\)](#) - Description

*pathElement* A [PathElement](#) (object)

*value* **(String)** Value for the element. Use pathElement.Values to return a list of valid settings for this element.

**Return Type** String

**Default** Not Applicable

**Examples** `` Set the "Combiner" element to value "Reversed"`

```
chan.PathConfiguration.Element("Combiner").Value = "Reversed"
```

```
setting=pathElement.Value
```

**C++ Syntax** HRESULT get\_Value( BSTR\* pValue );  
HRESULT put\_Value( BSTR value );

**Interface** IPathElement

---

Last Modified:

14-Dec-2006    MX New topic

## Values Property

---

**Description** Returns an array of valid settings that can be used with the element object.

See a [list of configurable elements and settings](#) for various PNA models.

**VB Syntax** *values* = *pathElement*.**Values**

**Variable** [\(Type\)](#) - **Description**

*values* **(Variant array)** Variable to store the array of valid settings for the element.

*pathElement* A [PathElement](#) **(object)**

**Return Type** Variant array

**Default** Not Applicable

**Examples** `settings=pathElement.Values`

**C++ Syntax** HRESULT Values( VARIANT\* pValues );

**Interface** IPathElement

---

Last Modified:

14-Dec-2006    MX New topic



## VelocityFactor Property

---

**Description** Sets the velocity factor to be used with Electrical Delay, Port Extensions, and Time Domain marker distance calculations.

**VB Syntax** *app.VelocityFactor = value*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*value* (**double**) - Velocity factor. Choose a number between: **0** and **10**  
(.66 polyethylene dielectric; .7 teflon dielectric)

**Return  
Type** Double

**Default** 1

**Examples** `app.VelocityFactor = .66 'Write`

`RelVel = app.VelocityFactor 'Read`

**C++ Syntax** HRESULT get\_VelocityFactor(double \*pVal)  
HRESULT put\_VelocityFactor(double newVal)

**Interface** IApplication

## View Property

---

**Description** Sets (or returns) the type of trace displayed on the screen.

**VB Syntax** *meas.View = value*

**Variable** [\(Type\)](#) - Description

*meas* A measurement (**object**)

*value* (**enum NAVView**) - Type of trace. Choose from:

**0** - naData

**1** - naDataAndMemory

**2** - naMemory

**3** - naNoTrace

**Note:** The **naData** trace may reflect the result of a [TraceMath](#) operation.

**Return Type** NAVView

**Default** naData

**Examples** `meas.View = naData 'Write`

`trceview = meas.View 'Read`

**C++ Syntax** HRESULT get\_View(tagNAView\* pView)  
HRESULT put\_View(tagNAView newView)

**Interface** IMeasurement

## Visible Property

---

**Description** Makes the Network Analyzer application visible or not visible. In the Not Visible state, the analyzer cycle time for making measurements can be significantly faster because the display does not process data.

**VB Syntax** `app.Visible = state`

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

*state* (boolean)

**False** - Network Analyzer application **NOT** visible

**True** - Network Analyzer application **IS** visible

**Return  
Type** Boolean

**Default** True

**Examples** `app.Visible = False 'Write`

`vis = app.Visible 'Read`

**C++ Syntax** HRESULT get\_Visible(VARIANT\_BOOL \* bVisible)  
HRESULT put\_Visible(VARIANT\_BOOL bVisible)

**Interface** IApplication

## WGCutoffFreq Property

---

**Description** Sets or returns the value of the waveguide cut off frequency.

**VB Syntax** *meas.WGCutoffFreq = value*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement **(object)**

*value* **(double)** - Frequency in Hertz.

**Return Type** Double

**Default** Not Applicable

**Examples** `Print meas.WGCutoffFreq 'prints the value of the waveguide cut off frequency`

**C++ Syntax** HRESULT get\_WGCutoffFreq(double \*pVal);  
HRESULT put\_WGCutoffFreq(double newVal);

**Interface** IMeasurement2

## Width Property

---

**Description** Sets the pulse width - the amount of time that the pulse is ON.

**VB Syntax** *pulse.Width (n) = value*

**Variable** [\(Type\)](#) - Description

*pulse* A [PulseGenerator](#) (object)

*n* **(Integer)** Pulse generator number. Choose from 0 to 4.  
0 is the generator that pulses the ADC.

*value* **(Double)** Pulse width in seconds. Choose a value from about 33ns to about 70 seconds.

**Return Type** Double

**Default** 1e-4 sec

**Examples**

```
pulse.Width = 1ms 'Write
```

```
value = pulse.Width 'Read
```

**C++ Syntax** HRESULT get\_Width (integer pulse, double\* width);  
HRESULT put\_Width (integer pulse, double width);

**Interface** IPulseGenerator

---

Last Modified:

2-Jan-2007    MX New topic

Read-only

## WindowNumber Property

---

**Description** Returns the window number. You might use this property to identify a particular window so that you can create a new Measurement in that window.

**VB Syntax** *value* = *win*.**WindowNumber**

**Variable** [\(Type\)](#) - Description

*win* A NAWindow (object)

*value* **(long integer)** - Variable to store the returned window number

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `value = app.ActiveNAWindow.WindowNumber`

**C++ Syntax** HRESULT (long\* windowNumber);

**Interface** INAWindow

## WindowState Property

---

**Description** Sets or returns the window setting of Maximized, Minimized, or Normal. To arrange all of the windows, use app.[ArrangeWindows](#).

**VB Syntax** *object*.WindowState = *value*

**Variable** [\(Type\)](#) - Description

*object* An [Application](#) (object) - main window  
or  
A NaWindow (**object**) - data windows

*value* (**enum NAWindowStates**) - The window state. Choose from:  
**0 - naMinimized** - Minimizes the window to an Icon on the lower toolbar  
**1 - naMaximized** - Maximizes the window  
**2 - naNormal** - changes the window size to the user defined setting (between Max and Min).

**Return Type** Long Integer

**Default** naMaximized

**Examples**

```
app.WindowState = naMinimized 'changes the Network Analyzer application window to an icon. -Write
win.WindowState = naNormal 'changes the window defined by the win object variable to user defined settings. -Write

winstat = app.WindowState 'Read
```

**C++ Syntax** HRESULT get\_WindowState(tagNAWindowStates \*pVal)  
HRESULT put\_WindowState(tagNAWindowStates newVal)

**Interface** INAWindow  
IApplication

## XAxisAnnotation Property

---

**Description** Returns the X-Axis annotation of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag.XAxisAnnotation* (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.XAxisAnnotation 3 'read`

**C++ Syntax** HRESULT XAxisAnnotation (long sweep, BSTR\* annotation);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic



## XAxisPointSpacing Property

---

**Description** Sets X-axis Point Spacing for the displaytraces measured with segment sweeps on the active channel.

**VB Syntax** *chan.XAxisPointSpacing = value*

**Variable** [\(Type\)](#) - Description

*chan* A Channel (**object**)

*value* (**Enum as naStates**) - Choose from:

**0 - naOFF** - Turns X-axis Point Spacing OFF

**1 - naON** - Turns X-axis Point Spacing ON

**Return Type** Enum

**Default** 0 - naOFF

**Examples** `chan.XAxisPointSpacing = naOFF 'Write`

`xspac = chan.XAxisPointSpacing 'Read`

**C++ Syntax** HRESULT get\_XAxisPointSpacing (tagNAStates \*pState);  
HRESULT put\_XAxisPointSpacing (tagNAStates newState);

**Interface** IChannel2

## XAxisStart Property

---

**Description** Returns the X-Axis start value of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag.XAxisStart* (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.XAxisStart 3 'read`

**C++ Syntax** HRESULT XAxisStart (long sweep, double\* start);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## XAxisStop Property

---

**Description** Returns the X-Axis stop value of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag.XAxisStop* (*n*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Double)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.XAxisStop 3 'read`

**C++ Syntax** HRESULT XAxisStop (long sweep, double\* start);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## YAxisAnnotation Property

---

**Description** Returns the Y-Axis annotation of the specified tuning sweep.

**VB Syntax** *value* = *embedLODiag*.YAxisAnnotation (*n*)

**Variable** [\(Type\)](#) - Description

*value* **(String)** Variable to store the returned data.

*embedLODiag* An [EmbeddedLODiagnostic](#) (**object**)

*n* (Long) Tuning sweep number. Use [NumberOfSweeps](#) to find the number of sweeps taken.

**Default** Not Applicable

**Examples** `data= embedLO.YAxisAnnotation 3 'read`

**C++ Syntax** HRESULT YAxisAnnotation (long sweep, BSTR\* annotation);

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

13-Apr-2007    MX New topic

## YScale Property

---

**Description** Sets or returns the Y-axis Per-Division value of the active trace.

**VB Syntax** *trace*.**YScale** = *value*

**Variable**    [\(Type\)](#) - Description

*trace*    A Trace (**object**)

*value*    (**double**) - Scale /division number. Units and range depend on the current data format.

**Return Type** Double

**Default** 10 (db)

**Examples** `trac.YScale = 5 'Write`

`yscl = trac.YScale 'Read`

**C++ Syntax** HRESULT get\_YScale(double \*pVal)  
HRESULT put\_YScale(double newVal)

**Interface** ITrace

## Z0 Property

---

**Description** Sets and Returns the characteristic impedance for the calibration standard.

**VB Syntax** *calstd.Z0* = *value*

**Variable** [\(Type\)](#) - **Description**

*calstd* A CalStandard (**object**). Use calKit.[GetCalStandard](#) to get a handle to the standard.

*value* (**single**) -Impedance in Ohms

**Return Type** Single

**Default** Not Applicable

**Examples** `calstd.Z0 = 50 'Write`

`impedance = calstd.Z0 'Read`

**C++ Syntax** HRESULT get\_Z0(float \*pVal)  
HRESULT put\_Z0(float newVal)

**Interface** ICalStandard

## Abort Method

---

**Description** Ends the current measurement sweep on the channel.

**VB Syntax** *chan*.**Abort** [*sync*]

**Variable** [\(Type\)](#) - Description

*chan* **(object)** - A [Channel](#) object

*sync* **(boolean)** - wait (or not) for the analyzer to stop before processing subsequent commands. Optional argument; if unspecified, value is set to False. Choose from:  
**True** - synchronize - the analyzer will not process subsequent commands until the current measurement is aborted.  
**False** - continue processing commands immediately

**Return Type** None

**Default** None

**Examples**

```
chan.abort True  
chan.abort
```

**C++ Syntax** HRESULT Abort(VARIANT\_BOOL bSynchronize);

**Interface** IChannel

## AbortPowerAcquisition Method

---

**Description** Aborts a source power cal acquisition sweep that is currently in progress.

**VB Syntax** *powerCalibrator*.**AbortPowerAcquisition**

**Variable** [\(Type\)](#) - Description

*powerCalibrator* **(object)** - A [SourcePowerCalibrator](#) object

**Return Type** None

**Default** Not Applicable

**Examples** `powerCalibrator.AbortPowerAcquisition`

**C++ Syntax** HRESULT AbortPowerAcquisition();

**Interface** ISourcePowerCalibrator



**AcquireCalStandard Method - Superseded**

**Description** **Note:** This command has been replaced by [AcquireCalStandard2 Method](#), which provides for acquisition of sliding load standards. All other functionality is identical.

**VB Syntax** `cal.AcquireCalStandard std[,index]`

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

*std* (**enum NACalClass**) Standard to be measured. Choose from:

- 1 - naClassA
- 2 - naClassB
- 3 - naClassC
- 4 - naClassD
- 5 - naClassE
- 6 - naReferenceRatioLine
- 7 - naReferenceRatioThru

**SOLT Standards**

- 1 - naSOLT\_Open
- 2 - naSOLT\_Short
- 3 - naSOLT\_Load
- 4 - naSOLT\_Thru
- 5 - naSOLT\_Isolation

**TRL Standards**

- 1 - naTRL\_Reflection
- 2 - naTRL\_Line\_Reflection
- 3 - naTRL\_Line\_Tracking
- 4 - naTRL\_Thru
- 5 - naTRL\_Isolation

*index* (**long integer**) number of the standard. Optional argument - Used if there is more than one standard required to cover the necessary frequency range. If unspecified, value is set to 1.

**Note** The behavior has changed with PNA revisions as follows:

- Before 6.01: Accepted 0 and changed it to 1
- 6.01 to 6.04: Did NOT accept 0
- 6.04.11 and higher: Accepts 0 and changes it to 1

**Return Type** None

**Default** Not Applicable

---

**Examples** `Cal.AcquireCalStandard naSOLT_Thru 'Write`

---

**C++ Syntax** HRESULT AcquireCalStandard(tagNACalClass enumClass, short standardNumber)

**Interface** ICalibrator

---

Last modified:

10/05/06    Modified Index argument.

## AcquireCalStandard2 Method

---

**Description** Measures the specified standard from the selected calibration kit. The calibration kit is selected using [app.CalKitType](#).

For 2-port calibration, it is also necessary to specify direction with [AcquisitionDirection](#).

To omit Isolation from a 2-port calibration, do not Acquire a cal standard for naSOLT\_Isolation.

For using two sets of standards, see [Simultaneous2PortAcquisition Property](#).

**Note:** This command replaces [AcquireCalStandard](#). This command provides for the acquisition of a sliding load cal. All other functionality is identical.

**VB Syntax** `cal.AcquireCalStandard2 std[,index][,slide]`

**Variable** [\(Type\)](#) - Description

*cal* A [Calibrator](#) (object)

*std* (**enum NACalClass**) Standard to be measured. Choose from:

1 - naClassA

2 - naClassB

3 - naClassC

4 - naClassD

5 - naClassE

6 - naReferenceRatioLine

7 - naReferenceRatioThru

### SOLT Standards

1 - naSOLT\_Open

2 - naSOLT\_Short

3 - naSOLT\_Load

4 - naSOLT\_Thru

5 - naSOLT\_Isolation

### TRL Standards

1 - naTRL\_Reflection

2 - naTRL\_Line\_Reflection

3 - naTRL\_Line\_Tracking

4 - naTRL\_Thru

5 - naTRL\_Isolation

*[index]* **(long integer)** Number of the standard. Optional argument - Used if there is more than one standard required to cover the necessary frequency range. If unspecified, value is set to 1.

**Note** The behavior has changed with PNA revisions as follows:

- Before 6.01: Accepted 0 and changed it to 1
- 6.01 to 6.04: Did NOT accept 0
- 6.04.11 and higher: Accepts 0 and changes it to 1

*[slide]* **(enum as NACalStandardSlidingState)** Optional argument. State of the sliding load. The slide should be set a minimum of five times. Seven is the maximum that can be stored. Choose from:

0 - **naNotSlidingStd** - not using a sliding load - Default if not specified.

1 - **naSlidelsSet** - slide is set for acquisition

2 - **naSlidelsDone** - this next acquisition will be the last. Calculations will then be performed.

**Return Type** None

**Default** Not Applicable

**Examples**

```
Cal.AcquireCalStandard2 naSOLT_Thru
Cal.AcquireCalStandard2 naSOLT_Thru,2,naNotSlidingStd
'measures the second standard listed in the class of naSOLT_Thru
```

**C++ Syntax** HRESULT AcquireCalStandard2(NACalClass enumClass, long standardPosition, NACalStandardSlidingState slidingStandardState)

**Interface** ICalibrator

---

Last modified:

10/05/06    Modified Index argument.

## AcquireCalConfidenceCheckECALEX Method

---

<b>Description</b>	<p>This method replaces <a href="#">AcquireCalConfidenceCheckECAL</a></p> <p>Transfers confidence data from the specified ECal module into the measurement's memory trace. The data is transferred to the specified S-parameter on the same channel as this Calibrator object.</p> <p>The characterization within the ECal module that the confidence data will be read from is specified by <a href="#">ECALCharacterizationEx</a>. The default value is 0.</p>
<b>VB Syntax</b>	<i>cal.AcquireCalConfidenceCheckECALEX Sparam [,ecalModule]</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>cal</i>	A Calibrator ( <b>object</b> )
<i>Sparam</i>	<b>(String)</b> S-parameter to transfer confidence data to. This parameter must be present on the same channel as the calibrator object.
<i>ecalModule</i>	<p><b>(Integer)</b> – Optional argument. ECal module.</p> <p>Choose from modules <b>1</b> through <b>8</b></p> <p>Use <a href="#">IsECALModuleFoundEx</a> to determine the number of modules connected to the PNA</p> <p>Use <a href="#">GetECALModuleInfoEx</a> to return the model and serial number of each module.</p>
<b>Return Type</b>	None
<b>Default</b>	Not applicable
<b>Examples</b>	<code>Cal.AcquireCalConfidenceCheckECALEX "S11", 2</code>
<b>C++ Syntax</b>	<code>HRESULT AcquireCalConfidenceCheckECALEX(BSTR strParameter, long moduleNumber = 1);</code>
<b>Interface</b>	ICalibrator4

## AcquirePowerReadingsEx Method

**Description** This command replaces [AcquirePowerReadings Method](#)

Initiates a source power cal acquisition.

**VB Syntax** *powerCalibrator.AcquirePowerReadingsEx calMethod, acqdevice [,sync]*

**Variable** [\(Type\) - Description](#)

*powerCalibrator* **(object)** - A [SourcePowerCalibrator](#) object

*calMethod* **(enum NASourcePowerCalMethod)** Selects the calibration method to be used for the source power cal acquisition.

**0 – naPowerMeter** Use power meter for all readings.

**1 - naPowerMeterAndReceiver** Power meter for the first iteration; then use the reference receiver for remaining readings if necessary.

**2 - naReceiver** Use PNA measurement receiver for all readings.

*acqdevice* **(String)** The specific acquisition device to be used. NOT case sensitive. Choose from:

If *calMethod* = **naPowerMeter** or **naPowerMeterAndReceiver**, choose from:

"ASEN" -- Sensor on power meter channel A.

"BSEN" -- Sensor on power meter channel B.

To use the sensor that currently corresponds to the frequency of interest, use the value from the [PowerAcquisitionDevice](#) property.

If *calMethod* = **naReceiver**, choose from the receiver names for your specific PNA using either physical receiver notation or [logical receiver notation](#). For example, "a1" or "A".

*[sync]* **(boolean)** Optional argument. If not specified, value is set to False. Choose from:

**True (1)** – The method does not return until this acquisition has completed (the program calling this method is halted while waiting for the method to return).

**False (0)** – The method initiates an acquisition then returns immediately (while the acquisition still proceeds). The program calling this method can then perform other operations during the acquisition.

**Return Type** None

**Default** Not Applicable

**Examples**

```
powerCalibrator.AcquirePowerReadingsEx naPowerMeter, "asen",
True
powerCalibrator.AcquirePowerReadingsEx naReceiver, "b2"
```

**C++ Syntax** HRESULT AcquirePowerReadingsEx (tagNASourcePowerCalMethod enumCalMethod, BSTR bstrAcqDevice, VARIANT\_BOOL bSync);

Last modified:

9/12/06 MQ New command to accommodate receiver only SPC

Write-only

## AcquireStep Method

---

**Description** Acquire the measurement data for the specified step in the calibration process. For an ECal User characterization this measures the ECal module.

**Note:** Guided Cal allows you to measure standards in any order. [See an example.](#)

**VB Syntax** *obj.AcquireStep (n)*

**Variable** [\(Type\)](#) - **Description**

*obj* Any of the following:

[GuidedCalibration](#) (object)

[SMCType](#) (object)

[VMCType](#) (object)

[ECalUserCharacterizer](#) (object) - Currently, only ONE step is required to measure the ECal module.

*n* Step number in the calibration process.

Use [GenerateSteps](#) to determine the total number of steps.

Use [GetStepDescription](#) to read the description of each step.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `VMC.AcquireStep (3)`

**C++ Syntax** HRESULT put\_AcquireStep(long step);

**Interface** SMCType  
VMCType  
IGuidedCalibration  
IECalUserCharacterizer

---

Last Modified:

4-Nov-2008 Added ECalUserChar object

20-Jan-2007 Added any order note.



## Activate Method

---

<b>Description</b>	<p>Makes an object the Active Object. When making a measurement active, the channel and window the measurement is contained in becomes the active channel and active window.</p> <p>In order to change properties on any of the active objects, you must first have a "handle" to the active object using the <b>Set</b> command. For more information, See <a href="#">Getting a Handle to an Object</a>.</p> <p>You do not have to make an object "Active" to set or read its properties remotely. But an object must be "Active" to change its values from the front panel.</p>
<b>VB Syntax</b>	<i>object</i> . <b>Activate</b>
<b>Variable</b>	<b>(Type) - Description</b>
<i>object</i>	<a href="#">Measurement</a> ( <b>object</b> ) or <a href="#">Marker</a> ( <b>object</b> )
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>meas.Activate mark.Activate</pre>
<b>C++ Syntax</b>	HRESULT Activate()
<b>Interface</b>	IMeasurement IMarker

## ActivateMarker Method

---

**Description** Makes a marker the Active Marker. Use meas.[ActiveMarker](#) to read the number of the active marker.

**VB Syntax** *meas*.**ActivateMarker**(*Mnum*)

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*Mnum* **(long integer)** - the number of the marker to make active. Choose any marker number from **1** to **9**.

**Return Type** None

**Default** Not Applicable

**Examples** `meas.ActivateMarker(1) 'Write`

**C++ Syntax** HRESULT ActivateMarker(long IMarkerNumber)

**Interface** IMeasurement

**Remarks** Use [ReferenceMarkerState](#) to control the Reference marker.

## ActivateWindow Method

---

**Description** Makes a window object the Active Window.

In order to change properties on any of the active objects, you must first have a "handle" to the active object using the **Set** command. For more information, See [Programming the Analyzer Object Model](#).

You do not have to make an object "Active" to set or read its properties remotely. But an object must be "Active" to change its values from the front panel.

**VB Syntax** `app.ActivateWindow n`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*n* (**long**) Number of the window to make active

**Return Type** Window Object

**Default** Not Applicable

**Examples** `app.ActivateWindow 4`

**C++ Syntax** HRESULT ActivateWindow(long WindowNumber)

**Interface** IApplication

[See the PNA Object Model](#)

## Add (channels) Method

---

**Description** Creates a channel and returns a handle to it. If the channel already exists, it returns the handle to the existing channel.

**VB Syntax** *chans.Add (item)*

**Variable** [\(Type\)](#) - Description

*chans* A [Channel collection](#) (**object**)

*item* (**variant**) - Channel number.

**Return Type** Channel

**Default** Not Applicable

**Examples** `chans.Add 3 'Creates channel 3`

**C++ Syntax** HRESULT Add(VARIANT numVal, IChannel\*\* pChannel)

**Interface** IChannels

## Add (measurement) Method

---

**Description** Adds a Measurement to the collection.

**VB Syntax** *meas.Add channel,param,source[,window]*

*meas* A [Measurements](#) collection (**object**)

*channel* (**long**) - Channel number of the new measurement.

*param* (**string**) - New parameter. Case insensitive.

### For S-parameters:

[Any S-parameter that can be measured by your PNA.](#)

Single-digit port numbers can be separated by "\_" (underscore). For example: "S21" or "S2\_1"

Double-digit port numbers MUST be separated by underscore. For example: "S10\_1"

### For Ratioed measurements:

Any two receivers in your PNA separated by "/". For example: "A/R1"

See the [block diagram](#) showing the receivers in YOUR PNA.

### For Unratioed (absolute power) measurements:

Any receiver in the PNA. For example: "A"

See the [block diagram](#) showing the receivers in YOUR PNA

With PNA Rev 6.2, **Ratioed** and **Unratioed** measurements can also use **logical receiver notation** to refer to receivers. This notation makes it easy to refer to receivers with an [external test set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port. [Learn more.](#)

### For ADC measurements

Any ADC receiver in the PNA followed by a comma, then the source port.

For example: "AI1\_2" indicates the Analog Input1 with source port of 2.

[Learn more about ADC receiver measurements.](#)

### For Balanced S-parameter measurements:

"topology:Sabxy"

*topology* - Choose from:

- **sbal** - single-ended to balanced
- **ssb** - single-ended / single-ended to balanced
- **bbal** - balanced to balanced

**Sabxy -**

**Where**

**a** - device output (receive) mode

**b** - device input (source) mode

(choose from the following for both a and b:)

- **d** - differential
- **c** - common
- **s** - single ended

**x** - device output (receive) logical port number

**y** - device input (source) logical port number

For example: "**sbal:sdd42**"

[See an example program](#)

**For Imbalance and Common Mode Rejection measurements:**

"*topology:parameter*" Choose from:

Choose this:	To get this:	
	Topology	Parameter
"SBAL:IMBSB"	single-ended to balanced	imbalance
"SBAL:CMRRSB1"	single-ended to balanced	common mode rejection (Sds21/Scs21)
"SBAL:CMRRSB2"	single-ended to balanced	common mode rejection (Ssd12/Scs12)
"SSB:IMB1SSB"	single-ended / single-ended to balanced	imbalance 1
"SSB:IMB2SSB"	single-ended / single-ended to balanced	imbalance 2
"SSB:CMRRSSB1"	single-ended / single-ended to balanced	common mode rejection (Sds31/Scs31)
"SSB:CMRRSSB2"	single-ended / single-ended to balanced	common mode rejection (Sds32/Scs32)
"BBAL:IMB1BB"	balanced to balanced	imbalance 1

"BBAL:IMB2BB"	balanced to balanced	imbalance 2
"BBAL:CMRRBB"	balanced to balanced	common mode rejection (Sdd21/Sc21)

*source* **(long integer)** - Source port number; if unspecified, value is set to 1. Only used for non-s-parameter measurements; ignored if s-parameter.

*window* **(long integer)** - Optional argument. Window number of the new measurement. If unspecified, the S-Parameter will be created in the Active Window. Choose between 1 and the [maximum number of windows allowed on the PNA](#). If unspecified, the measurement will be created in the Active Window.

See also [Traces, Channels, and Windows on the PNA](#)

**Return Type** None

**Default** None

**Examples** `meass.Add 3, "A/R1", 1, 1` 'Adds A/R1 measurement to channel 3 in window 1

**C++ Syntax** HRESULT Add(long ChannelNum, BSTR strParameter, long srcPort, VARIANT\_BOOL bNewWindow)

**Interface** IMeasurements

---

Last modified:

9/12/06 MQ Added logical receiver notation and number of windows.

## Add (NAWindows) Method

---

**Description** Add a window to the display. Does not add a measurement. The window number must not already exist.

**VB Syntax** *wins.Add [item]*

**Variable** [\(Type\)](#) - Description

*wins* A [NAWindow](#) collection (**object**)

*item* (**variant**) - Window number. Choose between 1 and the [maximum number of windows allowed on the PNA](#).

See also [Traces, Channels, and Windows on the PNA](#)

**Return Type** Object

**Default** Not Applicable

**Examples** `wins.Add 3 'Creates a window number 3`

**C++ Syntax** HRESULT Add(long windowNumber )

**Interface** INAWindows

---

Last modified:

4-Mar-2009    Removed optional argument

9/12/06    Modified for number of windows



## Add (PowerLossSegment) Method

---

<b>Description</b>	Adds a PowerLossSegment to the PowerLossSegments collection. Also, Adds a PowerLossSegmentPMAR to the PowerLossSegmentsPMAR collection.  To ensure predictable results, it is best to remove all segments before defining a new list of segments. For each segment in the collection, do a seg. <a href="#">Remove</a> .
<b>VB Syntax</b>	<i>segs.Add (item [ size])</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>segs</i>	A <a href="#">PowerLossSegments</a> collection A <a href="#">PowerLossSegmentsPMAR</a> collection.
<i>item</i>	<b>(variant)</b> - Number of the new segment. If it already exists, a new segment is inserted at the requested position.
<i>size</i>	<b>(long integer)</b> - Optional argument. The number of segments to add, starting with item. If unspecified, value is set to 1.
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>segs.Add 1, 4 'Adds segments 1,2,3 and 4</code>
<b>C++ Syntax</b>	HRESULT Add(VARIANT index, long size);
<b>Interface</b>	IPowerLossSegments

---

Last Modified:

25-Aug-2009    Added PMAR

## Add (PowerSensorCalFactorSegment) Method

---

<b>Description</b>	Adds a PowerSensorCalFactorSegment to the CalFactorSegments collection. Also Adds a PowerSensorCalFactorSegmentPMAR to the CalFactorSegmentsPMAR collection.  To ensure predictable results, it is best to remove all segments before defining a new list of segments. For each segment in the collection, do a seg. <a href="#">Remove</a> .
<b>VB Syntax</b>	<i>segs.Add (item [ size])</i>
<b>Variable</b>	<b><a href="#">(Type)</a> - Description</b>
<i>segs</i>	A <a href="#">CalFactorSegments</a> (collection) or A <a href="#">CalFactorSegmentsPMAR</a> (collection)
<i>item</i>	<b>(variant)</b> - Number of the new segment. If it already exists, a new segment is inserted at the requested position.
<i>size</i>	<b>(long integer)</b> - Optional argument. The number of segments to add, starting with item. If unspecified, value is set to 1.
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>segs.Add 1, 4 'Adds segments 1,2,3 and 4</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT Add(VARIANT index, long size);
<b>Interface</b>	ICalFactorSegments ICalFactorSegmentsPMAR

---

Last Modified:

25-Aug-2009    Added PMAR

## Add (segment) Method

---

**Description** Adds segments to the Segments collection, but does not turn the segments ON.

**VB Syntax** *segs.Add (item, [size])*

*segs* A [segments](#) collection (**object**)

*item* (**variant**) Number of the new segment. If it already exists, a new segment is inserted at the requested position.

*size* (**long integer**) Optional argument. The number of segments to add, starting with *item*. If unspecified, value is set to 1.

**Return Type** None

**Default** None

**Examples** `Segs.Add 1, 4 'Adds segments 1,2,3,and 4. (does NOT automatically turn segments ON)`

**C++ Syntax** HRESULT Add(VARIANT index, long size);

**Interface** ISegments

**Remarks** To ensure predictable results, it is best to remove all segments before defining a segment list. For each segment in the collection, do a seg.[Remove](#).

## Add (Testset) Method

---

**Description** Adds a testset to the ExternalTestsets Collection and loads the configuration file.

**VB Syntax** *testsets.Add (model,address)*

**Variable** [\(Type\) - Description](#)

*testsets* An [ExternalTestsets](#) (collection)

*model* **(String)** Model of the testset to be added, NOT case-sensitive.

There is no COM command to read a list of currently-supported test sets. However, the following SCPI command can be used with the following format:

```
string = SCPIStringParser.Execute (" SENSe:MUlTIpLExer:CATalog? ")
```

*address* **(Integer)** Address of the testset to be added.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
testsets.Add("Z5623AK66",12) ' add Z5623AK66 test at address 12
to testsets collection
```

[See an example program](#)

**C++ Syntax** HRESULT Add(BSTR typename, long address)

**Interface** [IExternalTestsets](#)

## AllowAllEvents Method

---

**Description** Sets event filtering to monitor all events in the analyzer. This is the default setting when subscribing to events. This could slow the measurement speed of the analyzer significantly.

**VB Syntax** *app*.**AllowAllEvents**

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.AllowAllEvents`

**C++ Syntax** HRESULT AllowAllEvents()

**Interface** IApplication

## AllowChannelToSweepDuringCalAcquisition Method

---

**Description** Specifies the channel to sweep during a Calibration.

When this command is sent, the *SwpChan* channel is 'flagged' to be swept during calibration. The flag is cleared when the channel is deleted, if the Measurement Class is changed, or if all measurements are deleted from the channel. If the same channel number is recreated, this command must be sent again to sweep the channel during a calibration. The flag is NOT saved with an instrument state.

A Preset or Instrument State Recall deletes the channel.

**VB Syntax** *calMgr*.**AllowChannelToSweepDuringCalAcquisition** (*CalChan*, *SwpChan*, *State*)

**Variable** [\(Type\)](#) - **Description**

*calMgr* **(object)** - A [CalManager](#) object

*CalChan* **(long)** - Channel to be calibrated.

*SwpChan* **(long)** - The channel to sweep when waiting to measure a standard.

This channel must already exist with at least one measurement in the channel. If this channel is in continuous sweep mode, it must have the same attenuator settings and path configuration (PNA-X only).

*state* **(Boolean)** - Channel sweep state. Choose from:

**True** - Sweep the channel during calibration.

**False** - Do NOT sweep the channel during calibration.

**Return Type** Not Applicable

**Default** Not Applicable

**Example** `calMgr.AllowChannelToSweepDuringCalAcquisition 2,1,True`

[See example using this command](#)

**C++ Syntax** HRESULT AllowChannelToSweepDuringCalAcquisition ( long CalChannel, long SwpChannel, VARIANT\_BOOL bVal);

**Interface** ICalManager5

---

Last Modified:

23-Jan-2009 Added 'if measurements are deleted'.

8-Nov-2007 MX New topic



## AllowEventCategory Method

---

**Description** Sets event filtering to monitor a category of event.

**VB Syntax** *app.AllowEventCategory, category, state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*category* Category to monitor. Choose from list in [Working with the Analyzer's Events](#)

*state* **(boolean)**  
**True** - monitor  
**False** - do not monitor

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.AllowEventCategory`

**[C++ Syntax](#)** HRESULT AllowEventCategory(tagNAEventCategory category, VARIANT\_BOOL bAllow )

**Interface** IApplication



## AllowEventMessage Method

---

**Description** Sets event filtering to monitor specific events.

**VB Syntax** *app.AllowEventMessage event*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*event* Event to monitor. Refer to list in [Working with the Analyzer's Events](#)

*state* **(boolean)**  
**True** - monitor  
**False** - do not monitor

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.AllowEventMessage`

**C++ Syntax** HRESULT AllowEventMessage( tagNAEventID eventID, VARIANT\_BOOL bAllow)

**Interface** IApplication

## AllowEventSeverity Method

---

**Description** Sets event filtering to monitor levels of severity.

**VB Syntax** *app.AllowEventSeverity severity,state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*severity* (**enum naEventSeverity**) Choose from:naEventSeverityERROR  
naEventSeverityINFORMATIONAL  
naEventSeveritySUCCESS  
naEventSeverityWARNING

*state* (**boolean**)  
**True** - monitor  
**False** - do not monitor

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.AllowEventSeverity`

**C++ Syntax** HRESULT AllowEventSeverity( tagNAEventSeverity severity, VARIANT\_BOOL bAllow)

**Interface** IApplication

Write only

## Apply Method

---

**Description** Applies the mixer setup and turns the channel ON. (Performs the same function as the Apply button on the [mixer setup dialog box](#).)

**VB Syntax** *obj*.**Apply**

**Variable** [\(Type\)](#) - **Description**

*obj* A Mixer Interface pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mxr.Apply`

**C++ Syntax** HRESULT Apply()

**Interface** IMixer3  
IConverter

---

Last Modified:

2-Feb-2009    Added converter object

## ApplyDeltaMatchFromCalSet Method

---

**Description** Specifies a Cal Set as a source of delta match correction.

If 'GUID' is not supplied then the Global Delta Match Cal Set is assumed. An error is returned if the specified Cal Set does not meet the following Delta Match criteria. The Global Delta Match Cal can ALWAYS be applied.

- Must have been performed using ECal or as a guided mechanical cal (not Unguided).
- Must have the same start freq, stop freq, and number of points as the channel being calibrated.
- Must calibrate the ports that are required by the TRL or Unknown Thru Cal as indicated by [PortsNeedingDeltaMatch Property](#).

[Learn more about Delta match calibration.](#)

See example of a complete Delta Match calibration.

**VB Syntax** *guided*.**ApplyDeltaMatchFromCalSet** [*GUID*]

**Variable** [\(Type\)](#) - **Description**

*guided* [GuidedCalibration](#) (object)

*GUID* Optional Argument. GUID of the Cal Set to use. If unspecified, the Global Delta Match Cal Set is used.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `guided.ApplyDeltaMatchFromCalSet "{2B893E7A-971A-11d5-8D6C-00108334AE96}"`

**C++ Syntax** HRESULT ApplyDeltaMatchFromCalSet(BSTR calsetGUID);

**Interface** IGuidedCalibration2

## ApplyPowerCorrectionValuesEx Method

---

<b>Description</b>	<p>This command replaces <a href="#">ApplyPowerCorrectionValues Method</a>.</p> <p>Applies the array of power correction values to the channel memory and turns correction ON. Perform after completing a source power cal acquisition sweep.</p> <p>This command does NOT save the correction values. To save correction values, save an <a href="#">instrument / calibration state (*.cst file)</a> after performing a source power cal.</p> <p>Optionally, as part of the source power calibration, perform calibration of the reference receiver used in the power calibration. <a href="#">Learn more</a>.</p>
<b>VB Syntax</b>	<code>powerCalibrator.<b>ApplyPowerCorrectionValuesEX</b> [<i>rRec</i>]</code>
<b>Variable</b>	<b>(Type) - Description</b>
<code>powerCalibrator</code>	<b>(object)</b> - A <a href="#">SourcePowerCalibrator</a> object
<code>rRec</code>	(Enum as NASourcePowerApplyCorrectionOption) Optional argument. Choose from: <b>0 - naSourcePowerApplyCorrectionDefault</b> Do NOT perform and save a calibration of the reference receiver. (Default if not specified). <b>1 - naIncludeReferenceReceiverPowerCal</b> Perform and save a calibration of the reference receiver. The Cal Set, which includes only the reference receiver cal, is saved to the destination specified by <a href="#">RemoteCalStoragePreference</a> .
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>powerCalibrator.ApplyPowerCorrectionValuesEX powerCalibrator.ApplyPowerCorrectionValuesEX (naIncludeReferenceReceiverPowerCal)</pre>
<b>C++ Syntax</b>	HRESULT ApplyPowerCorrectionValuesEx(enum NASourcePowerApplyCorrectionOption option);
<b>Interface</b>	ISourcePowerCalibrator5

---

Last Modified:

23-Apr-2007    MX New topic

## ApplySourcePowerCorrectionTo Method

---

**Description** Copies and applies an existing Source Power Calibration to another channel.

**VB Syntax** *chan*.**ApplySourcePowerCorrectionTo** (*fromPortNum*, *targetChan*, *targetPortNum*);

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (object)

*fromPortNum* (Long) Port number of the existing source power correction.

*targetChan* (Long) Channel number to which the source power correction will be copied.

*targetPortNum* (Long) Port number to which the source power correction will be applied.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `chan.ApplySourcePowerCorrectionTo 1,2,1`

**C++ Syntax** HRESULT ApplySourcePowerCalibrationTo (long fromPortNumber, long otherChannelNumber, long portNumber);

**Interface** IChannel11

---

Last Modified:

20-Jul-2007    MX New topic

Write only

## AssignSourceToRole Method

---

**Description** Assigns a configured source to the specified role.

**VB Syntax** *conv.AssignSourceToRole role, source*

**Variable** [\(Type\)](#) - Description

*conv* A [Converter Object](#)

*role* (String) Role to which the source is assigned. Choose from:  
"RF2"  
"LO1"

*source* (String) Source name from [Source Configuration dialog](#).

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `conv.AssignSourceToRole "LO1", "LO1Name"`

**C++ Syntax** HRESULT AssignSourceToRole(BSTR roleID, BSTR deviceName);

**Interface** IConverter

---

Last Modified:

2-Feb-2009    New topic

## AutoPortExtMeasure Method

---

**Description** Measures either an OPEN or SHORT standard. When this command is sent, the PNA acquires the measurement with which to set automatic port extensions. [Learn more about choosing which standard to measure.](#)

**VB Syntax** *fixture*.AutoPortExtMeasure *value*

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

*value* (Enum as NAAutoPortExtMeasure)  
0 - naAPEM\_OPEN - Measure OPEN  
1 - naAPEM\_SHORT - Measure SHORT

**Return Type** ENUM

**Default** Not Applicable

**Examples** `fixture.AutoPortExtMeasure naAPEM_OPEN`

**C++ Syntax** HRESULT get\_AutoPortExtMeasure(tagNAAutoPortExtMeasure \*pVal );

**Interface** IFixturing2



## AutoPortExtReset Method

---

**Description** Clears old port extension delay and loss data in preparation for acquiring new data. Send this command prior to sending a new series of measurements using [AutoPortExtMeasure Method](#). If acquiring both OPEN and SHORT standards, do not send this command between those acquisitions.

**VB Syntax** *fixture*.AutoPortExtReset

**Variable** [\(Type\)](#) - Description

*fixture* A [Fixturing](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `fixture.AutoPortExtReset`

**C++ Syntax** HRESULT AutoPortExtReset();

**Interface** IFixturing2

## Autoscale Method

---

**Description** Trace Object - Autoscales only the **ONE** trace on which Autoscale is being called.  
NAWindow Object - Autoscales **ALL** of the traces in the Window on which Autoscale is being called.

**VB Syntax** *object*.Autoscale

**Variable** [\(Type\)](#) - Description

*object* [Trace](#) (**object**)  
or  
[NAWindow](#) (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** Trac.Autoscale 'Autoscales the trace  
Win.Autoscale 'Autoscales all the traces in the window -Write

**[C++ Syntax](#)** HRESULT AutoScale()

**Interface** INAWindow  
ITrace

## AveragingRestart Method

---

**Description** Clears and restarts averaging of the measurement data.

**VB Syntax** *chan*.**AveragingRestart**

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `chan.AveragingRestart`

**[C++ Syntax](#)** HRESULT AveragingRestart()

**Interface** IChannel

## BuildHybridKit Method

---

**Description** Use this method when you have different port connectors. This is a convenient way to combine two kits that match the connectors on your DUT.

**VB Syntax** *app.BuildHybridKit port1Kit,p1sex,port2Kit,p2sex,adapter,user kit*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

*port1Kit* **(enum NACalKit)** - Specifies the two kits to be used to build the hybrid kit. Choose from:  
*port2Kit* naCalKit\_85032F\_N50  
 naCalKit\_85033E\_3\_5  
 naCalKit\_85032B\_N50  
 naCalKit\_85033D\_3\_5  
 naCalKit\_85038A\_7\_16  
 naCalKit\_85052C\_3\_5\_TRL  
 naCalKit\_User7  
 naCalKit\_User8  
 naCalKit\_User9  
 naCalKit\_User10

*p1sex* **(enum NAPortSex)** - Specifies the sex of the connector at that port. Choose from:  
*p2sex* naMale  
 naFemale  
 naDon'tCare

*adapter* **(enum NAAadapter)** -Choose from:  
 naUserkit - the electrical length of the adapter in the userKit specifications  
 naZeroLength - no adapter

*userKit* **(enum NACalKit)** - The Hybrid kit - Choose from the previous list of kits

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
app.BuildHybridKit
naCalKit_85033E_3_5,naMale,naCalKit_85038A_7_16
,naFemale,naUserkit,naCalKit_User8
```

**C++ Syntax** HRESULT BuildHybridKit/(tagNACalKit port1Kit, tagNAPortSex port1Sex, tagNACalKit port2Kit, tagNAPortSex port2Sex, tagNAAadapter adapter, tagNACalKit userKit)

**Interface** IApplication

## CalculateErrorCoefficients Method

---

**Description** This method is the final call in a calibration process. It calculates error-correction terms, turns error-correction ON and saves the error-correction terms to the channel's Cal Register or a User Cal Set.

Do NOT use this command during an ECAL.

**Note:** The destination (Cal Register or User Cal Set) is determined by the setting of the [RemoteCalStoragePreference](#) property.

**VB Syntax** *cal*.CalculateErrorCoefficients

**Variable** [\(Type\)](#) - Description

*cal* [Calibrator](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** Cal.CalculateErrorCoefficients

**C++ Syntax** HRESULT CalculateErrorCoefficients()

**Interface** ICalibrator

---

Last Modified:

16-Apr-2007    MX Added link to Remote...

## Calculate Method

---

**Description** Calculates the Input or Output frequencies of the mixer setup, applies the mixer setup to the mixer object, and turns the channel ON.

**VB Syntax** *obj.Calculate (port)*

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*port* (enum as MixerCalculation) Port of the mixer for which to calculate start and stop frequencies. Choose from:

enum		1st or only stage requires:	In addition, 2nd stage requires:
0	mixCalculateINPUT	<ul style="list-style-type: none"> <li>Output Start and Stop frequencies</li> <li>LO frequency</li> <li>Output sideband (High or Low)</li> </ul>	<ul style="list-style-type: none"> <li>IF Start and Stop frequencies</li> <li>2nd LO frequency</li> <li>IF sideband (High or Low)</li> </ul>
1	mixCalculateINPUT AndOUTPUT (2 stage mixers ONLY)	NA	<ul style="list-style-type: none"> <li>IF Start and Stop frequencies</li> <li>Both LO frequencies</li> </ul>
2	mixCalculateOUTPUT	<ul style="list-style-type: none"> <li>Input Start and Stop frequencies</li> <li>LO frequency</li> <li>Output sideband (High or Low)</li> </ul>	<ul style="list-style-type: none"> <li>IF Start and Stop frequencies</li> <li>2nd LO frequency</li> <li>IF sideband (High or Low)</li> </ul>
3	mixCalculateLO1	<ul style="list-style-type: none"> <li>Input Start and Stop frequencies</li> <li>Output frequency</li> <li>Output sideband (High or Low)</li> </ul>	<ul style="list-style-type: none"> <li>IF Start and Stop frequencies</li> <li>2nd LO frequency</li> <li>IF sideband (High or Low)</li> </ul>
4	mixCalculateLO2	NA	<ul style="list-style-type: none"> <li>Input Start and stop</li> </ul>

frequencies

- 1st LO start and stop frequencies
- Output frequency
- IF sideband(High or Low)
- Output sideband(High or Low)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `obj.Calculate (mixCalculateOUTPUT)`

**C++ Syntax** HRESULT Calculate()

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009    Added converter

## ChangeParameter Method

---

**Description** Changes the parameter of the measurement.

**VB Syntax** `meas.ChangeParameter(param,src)`

**Variable** [\(Type\)](#) - Description

*meas* A [Measurement](#) (object)

*param* (**string**) - New parameter. Case insensitive.

### For S-parameters:

[Any S-parameter that can be measured by your PNA.](#)

Single-digit port numbers can be separated by "\_" (underscore). For example: "S21" or "S2\_1"

Double-digit port numbers MUST be separated by underscore. For example: "S10\_1"

### For Ratioed measurements:

Any two receivers in your PNA separated by "/". For example: "A/R1"

See the [block diagram](#) showing the receivers in YOUR PNA.

### For Unratioed (absolute power) measurements:

Any receiver in the PNA. For example: "A"

See the [block diagram](#) showing the receivers in YOUR PNA

With PNA Rev 6.2, **Ratioed** and **Unratioed** measurements can also use **logical receiver notation** to refer to receivers. This notation makes it easy to refer to receivers with an [external test set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port. [Learn more.](#)

### For ADC measurements

Any ADC receiver in the PNA.

For example: "AI1" indicates the Analog Input1.

[Learn more about ADC receiver measurements.](#)

### For Balanced S-parameter measurements:

"*topology:Sabxy*"

*topology* - Choose from:

- **sbal** - single-ended to balanced
- **ssb** - single-ended / single-ended to balanced



- **bbal** - balanced to balanced

**Sabxy** -

**Where**

**a** - device output (receive) mode

**b** - device input (source) mode

(choose from the following for both a and b:)

- **d** - differential
- **c** - common
- **s** - single ended

**x** - device output (receive) logical port number

**y** - device input (source) logical port number

For example: "**sbal:sdd42**"

[See an example program](#)

For **Imbalance** and **Common Mode Rejection** measurements:

"*topology:parameter*" Choose from:

Choose this:	To get this:	
	Topology	Parameter
" <b>SBAL:IMBSB</b> "	single-ended to balanced	imbalance
" <b>SBAL:CMRRSB1</b> "	single-ended to balanced	common mode rejection (Sds21/Scs21)
" <b>SBAL:CMRRSB2</b> "	single-ended to balanced	common mode rejection (Ssd12/Scs12)
" <b>SSB:IMB1SSB</b> "	single-ended / single-ended to balanced	imbalance 1
" <b>SSB:IMB2SSB</b> "	single-ended / single-ended to balanced	imbalance 2
" <b>SSB:CMRRSSB1</b> "	single-ended / single-ended to balanced	common mode rejection (Sds31/Scs31)
" <b>SSB:CMRRSSB2</b> "	single-ended / single-ended to balanced	common mode rejection (Sds32/Scs32)

"BBAL:IMB1BB"	balanced to balanced	imbalance 1
"BBAL:IMB2BB"	balanced to balanced	imbalance 2
"BBAL:CMRRBB"	balanced to balanced	common mode rejection (Sdd21/Sc21)

*src* (long integer)

- Ignored if *param* is an S-Parameter
- Source port if *param* is a ratioed or unratioed measurement (including ADC measurements).

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
meas.ChangeParameter "S11",2 '2 is the source port
meas.ChangeParameter "A/R1",2 '2 is the source port
meas.ChangeParameter "a1/b1",1 '1 is the source port
meas.ChangeParameter "R1",2 '2 is the source port

'to change to a parameter with a string name
dim app
set app = CreateObject("Agilentpna835x.application")
dim capabilities
set capabilities = app.Capabilities
dim portnum
portnum = Capabilities.GetPortNumber("Src2 Out1")
app.activemeasurement.ChangeParameter "A",portnum
```

**C++ Syntax** HRESULT ChangeParameter(BSTR parameter, long lPort)

**Interface** IMeasurement

Last Modified:

24-Apr-2008 Added example for string names

30-Apr-2007 Modified for ADC and src strings



## CheckPower Method

---

**Description** Measures power at a specified frequency. Use this method to test power level before and/or after applying a source power calibration.

**VB Syntax** *pow* = *pwrCal*.**CheckPower** (*device*, *freq* [, *unit*])

**Variable**    [\(Type\)](#) - **Description**

*pow*    **(double)** Variable to store power value returned by this method.

*pwrCal*    A [SourcePowerCalibrator](#) **(object)**

*device*    **(enum NAPowerAcquisitionDevice)** The specific sensor on the power meter to be used for the acquisition. Choose from:

**0** – naPowerSensor\_A

**1** – naPowerSensor\_B

To use the sensor that currently corresponds to the frequency of interest, use the value from the [PowerAcquisitionDevice](#) property.

*freq*    **(double)** Frequency (Hz) at which the sensor is to read the power.

*unit*    **(enum NAPowerUnit)**

Optional argument. Choose from:

**naDBM** – Returns the power in dBm.(default)

**naWATT** – Returns the power in Watts.

**Return Type** Double

**Default** Not Applicable

**Examples**    `watt = powerCalibrator.CheckPower(naPowerSensor_A, 1E9, naWATT)`

**C++ Syntax** HRESULT put\_CheckPower(tagNAPowerAcquisitionDevice enumAcqDevice, double dFreq, tagNAPowerUnit enumPowerUnit, double \*pdPower);

**Interface** ISourcePowerCalibrator2

## Clear Method

---

**Description** Clears the current diagnostic information.

**VB Syntax** *embedLODiag*.Clear

**Variable** [\(Type\)](#) - Description

*embedLODiag* An [EmbeddedLODiagnostic](#) (object)

**Default** Not Applicable

**Examples** embedLO.Clear 'write

**C++ Syntax** HRESULT Clear();

**Interface** IEmbeddedLODiagnostic

---

Last Modified:

12-Apr-2007    MX New topic

## Clear Method

---

**Description** Clears the FIFO data buffer.

**VB Syntax** *fifo*.Clear

**Variable** [\(Type\)](#) - Description

*fifo* A [FIFO](#) (object)

**Default** Not Applicable

**Examples** `fifo.Clear 'write`

**[C++ Syntax](#)** HRESULT Clear();

**Interface** IFIFO

---

Last Modified:

3-Nov-2008    MX New topic

**CloseCalSet Method** **Superseded**

---

<b>Description</b>	<p>This command is no longer necessary. The CalSet.get... and put... commands that required this command have been replaced,</p> <p>Closes read/write access to the Cal Set.</p> <p>See <a href="#">OpenCalSet</a> for an explanation of gaining access to the Cal Set.</p> <p>When you are finished reading and writing data from or to the Cal Set, close the Cal Set. Subsequent read/writes will require a new OpenCal Set call.</p> <p>Reading and writing Cal Set data is performed with the PutStandard, GetStandard, PutErrorTerm, GetErrorTerm method calls. These methods are provided by the <a href="#">ICal Set</a> and ICalData2 interfaces.</p>
<b>VB Syntax</b>	<i>CalSet</i> . <b>CloseCalSet</b>
<b>Variable</b>	<b>(Type) - Description</b>
<i>CalSet</i>	<b>(object)</b> - A <a href="#">Cal Set</a> object
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<b>CSet.CloseCalSet</b>
<b>C++ Syntax</b>	HRESULT CloseCalSet
<b>Interface</b>	ICalSet

## ComputeErrorTerms Method

---

<b>Description</b>	<p>Computes error terms for the caltype specified by a preceding OpenCal Set call.</p> <p>The Cal Set must first be opened using <a href="#">OpenCalSet</a>. If this call has not been made, the following error is issued:</p> <p>E_NA_Cal Set_ACCESS_DENIED</p> <p>The standards data required for the CalType must be available in the Cal Set or this error will be returned: E_NA_STANDARD_NOT_FOUND.</p> <p><b>Note:</b> Error term computation requires data for the actual calibration kit standards from the current kit definition. ComputeErrorTerms assumes that the standards were acquired using only one standard per class.</p>
<b>VB Syntax</b>	<i>CalSet</i> . <b>ComputeErrorTerms</b>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
	<i>CalSet</i> <b>(object)</b> - A <a href="#">Cal Set</a> object
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<b>CalSet . ComputeErrorTerms</b>
<b>C++ Syntax</b>	HRESULT ComputeErrorTerms( )
<b>Interface</b>	ICalSet



## ConfigEnhancedNB2 Method

**Description** **Note:** This command replaces [ConfigEnhancedNB Method](#).

This subroutine determines, then returns, the proper configuration for pulsed measurements on the **PNA-X ONLY** using the spectral nulling technique. The configuration returned needs to be sent to the PNA and any other related external equipment.

The routine will take a desired Pulse Repetition Frequency (PRF) and measurement IFBW and return a possibly modified PRF and IFBW for proper pulsed operation on the PNA.

**VB Syntax** *Pulsed.ConfigEnhancedNB2 (PRF, BW, PhysicalIF, NCO, ClockFreq, Stage1TapArray, Stage2TapArray, Stage3TapArray, FixedPRF, GateDelay, GateWidth, SWGateDelay, SWGateWidth, SWGateRamp)*

**Variable** [\(Type\) - Description](#)

*Pulsed* **(interface)** An interface to the agilentpnapulsed.dll application interface.

*PRF* **(Double)** The Pulse Repetition Frequency.

**[out]** The pulse repetition frequency that has been optimized for use with the PNA.  
NOTE: This value may be different from the value requested.

**[in]** The desired pulse repetition frequency.

*BW* **(Long)** The PNA IF Bandwidth.

**[out]** The PNA IF bandwidth that has been optimized for use with the PNA. NOTE: This value may be different from the value requested. Zero (0) is returned if no solution is found for the specified *PRF* and *BW*.

**[in]** The desired PNA IF bandwidth.

*PhysicalIF* **(Double)**

**[out]** Returns physical intermediate frequency.

*NCO* **(Double)**

**[out]** Returns numeric controlled oscillator frequency.

*ClockFreq* **(Double)**

**[out]** Returns the clock frequency (in Hz) of the PNA-X.

*Stage1TapArray* **(Long array)**

**[out]** Returns the stage 1 filter coefficients

*Stage2TapArray* **(Long array)**

**[out]** Returns the stage 2 filter coefficients

*Stage3TapArray* **(Long array)**

**[out]** Returns the stage 3 filter coefficients

*FixedPRF* **(Boolean)**

**[in]**

- **1 (True)** Signals the .DLL routine to NOT adjust the PRF value; rather adjust ONLY the IF Bandwidth. This is the default setting.
- **0 (False)** Adjust both the PRF and IF Bandwidth values as necessary.

*GateDelay* **(Double)**

**[in]** Highest delay value in seconds used in any of the receiver gates.

*GateWidth* **(Double)**

**[in]** Widest pulse width value in seconds used in any of the receiver gates.

*SWGateDelay* **(Double)**

**[out]** Returns the SW gate delay in seconds.

*SWGateWidth* **(Double)**

**[out]** Returns the SW Gate width in seconds.

*SWGateRamp* **(Long)**

**[out]** Returns the SW Gate ramp

**Return Type** Not Applicable

**Default** Not Applicable

**Example** [See an example using this command.](#)

**C++ Syntax** HRESULT **ConfigEnhancedNB2**(double \*pPRF, long \*pBW, double \*pIF, double \*pNCO, double \*clock, double \*pStg1, double \*pStg2, double \*pStg3, VARIANT\_BOOL fixPRF, double gateDelay, double gateWidth, double \*SWgateDelay, double \*SWgateWidth, long \*SWgateRamp)

**Interface** AgilentPNAPulsed.Application

---

Last Modified:

28-Mar-2007    MX New topic

## ConfigEnhancedNBIFAtten Method

---

**Description** Sets PNA-X receivers to auto gain setting.

**VB Syntax** *Pulsed*.**ConfigEnhancedNBIFAtten** (*PRF*, *RxWidth*, *IFAtten*)

**Variable** [\(Type\)](#) - **Description**

*Pulsed* **(interface)** An interface to the agilentpnapulsed.dll application interface.

*PRF* **(Double)**

**[in]** The Pulse Repetition Frequency.

*RxWidth* **(Double)**

**[in]** Receiver gate width.

*IFAtten* **(Long Integer)**

**[out]** IF attenuation value.

**Return Type** Not Applicable

**Default** Not Applicable

**Example** [See an example using this command.](#)

**C++ Syntax** HRESULT **ConfigEnhancedNBIFAtten**(double \*pPRF, double \*pWidth, long \*pIF)

**Interface** AgilentPNAPulsed.Application

---

Last Modified:

18-Jun-2007    MX New topic

## ConfigNarrowBand3 Method

**Description** **Note:** This method replaces [ConfigNarrowBand2 Method](#). The BW argument now returns 0 if no solution is found for the specified PRF and BW. In addition, adjustments were made to the filter finder algorithm

This subroutine determines, then returns, the proper configuration for pulsed measurements on the PNA using the spectral nulling technique. The configuration returned needs to be sent to the PNA and any other related external equipment such as pulse generators. The routine will take a desired Pulse Repetition Frequency (PRF) and measurement IFBW and return a possibly modified PRF and IFBW for proper pulsed operation on the PNA. The routine will also return the Sample Rate, Number of Taps, and Offset that must be sent to the PNA to configure it in pulsed mode using the spectral nulling technique.

Although the example below uses COM programming to communicate with the PNA, these commands can be replaced with [SCPI equivalents](#).

**Note:** The pulsed application may set the offset frequency (option 080) of the PNA to some value other than zero (the default value). If the stop frequency is set to the maximum of the PNA model, then an error message may appear on the PNA stating that the response frequency has exceeded the maximum allowed frequency. To fix this, set the stop frequency to a value that is at least 2 KHz less than the maximum allowed. For example, if you have a 20 GHz PNA, and the stop frequency is set to 20 GHz, and the error message appears, then set the stop frequency to 19.999998 GHz

**VB Syntax** *Pulsed*.**ConfigNarrowBand** (*PRF, NumTaps, BW, OffSet, SampleRate, Precision, FixedPRF, PG81110*)

**Variable** [\(Type\)](#) - Description

*Pulsed* **(interface)** An interface to the agilentpnapulsed.dll application interface.

*PRF* **(Double)** The Pulse Repetition Frequency.

**[out]** The pulse repetition frequency that has been optimized for use with the PNA.  
NOTE: This value may be different from the value requested.

**[in]** The desired pulse repetition frequency.

*NumTaps* **(Long)** The number of taps to send to the PNA for pulsed operation.

*BW* **(Long)** The PNA IF Bandwidth.

**[out]** The PNA IF bandwidth that has been optimized for use with the PNA. NOTE: This value may be different from the value requested. Zero (0) is returned if no solution is found for the specified *PRF* and *BW*.

**[in]** The desired PNA IF bandwidth.

*Offset* **(Double)** The offset value to send to the PNA for pulsed operation. The offset value is used to adjust the PNA for the two different possible sample rates that may be returned.

*SampleRate* **(Double)**

**[out]** The sample rate to send to the PNA for pulsed operation.

**[in]** Passing a value of 6.2 us will make sure that the offset frequency is not shifted and therefore could be used with converter measurements. Otherwise enter 0.

*Precision* **(Double)** The precision variables sets the precision that will be used to decrement the PRF when running the configuration routines. This variable can be set to the precision required by the external pulse generators so that the configuration routine will not return a PRF that is not within the precision limits of the pulse generators.

*FixedPRF* **(Boolean)**

**1 (True)** Signals the .DLL routine to NOT adjust the PRF value; rather adjust ONLY the IF Bandwidth. This is the default setting.

**0 (False)** Adjust both the PRF and IF Bandwidth values as necessary.

*PG81110* **(Boolean)**

**1 (True)** You are using an Agilent 81110 as the pulse generator. This allows increased accuracy in adjustments for offset and PRF.

**0 (False)** Not using an Agilent 81110.

**Return Type** Not Applicable

**Default** Not Applicable

**Example** [See an example using this command.](#)

**C++ Syntax** HRESULT ConfigNarrowBand(double \*pPRF, long \*pNumTaps, long \*pBW, double \*pOffset, double \*pSampleRate, int Precision)

**Interface** AgilentPNAPulsed.Application

---

Last Modified:

21-Sep-2007    Modified sample rate and

## ConfigurationFile Method

---

**Description** Recalls an Interface Control file from the hard drive into the analyzer.

**VB Syntax** *IntControl*.**ConfigurationFile** (*filename*)

**Variable** [\(Type\)](#) - **Description**

*IntControl* An [InterfaceControl](#) (**object**)

*filename* (**string**) - Full path, file name, and extension (.xml) of the file to recall.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents"

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `IntControl.ConfigurationFile ("C:/Program Files/Agilent/Network Analyzer/Documents/MySettings.xml")`

**C++ Syntax** HRESULT ConfigurationFile(BSTR bstrFile)

**Interface** IInterfaceControl

## Configurations Property

---

**Description** Returns an array of stored configuration names that can be used with [DeleteConfiguration Method](#) and [LoadConfiguration Method](#)

**VB Syntax** *names* = *pathMgr*.**Configurations**

**Variable** [\(Type\)](#) - **Description**

*names* **(Variant array)** Variable to store the returned configuration names.

*pathMgr* [PathConfigurationManager](#) **(object)**

**Return Type** Variant array

**Default** Not Applicable

**Examples** `names = path.Configurations`

**C++ Syntax** HRESULT get\_Configurations (VARIANT\* configurations );

**Interface** IPathConfigurationManager

---

Last Modified:

14-Dec-2006    MX New topic

## Configure Method

---

**Description** Restarts as an "N-port" PNA using the specified multiport test set.

[See other commands to configure multiport test sets.](#)

**VB Syntax** *app.Configure (model, address)*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (**object**)

*model* **String** - Model of the test set with which to restart.

Use **"Native"** to restart without a test set.

To see a list of supported test sets, use

*address* **Integer** - GPIB Address of the test set. Use 0 for native restart.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See an example using this command.](#)

```
app.Configure ("N44xx",18)
```

**C++ Syntax** HRESULT Configure(BSTR model, long address);

**Interface** IApplication9



## Continuous Method

---

**Description** The channel continuously responds to trigger signals.

**Note:** This command does **NOT** change [TriggerSignal](#) to Continuous.

**VB Syntax** *chan*.Continuous

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `chan.Continuous`

**C++ Syntax** HRESULT Continuous()

**Interface** IChannel

## Copy Method

---

**Description** Creates a new Cal Set and copies the current Cal Set data into it. Therefore, you now have a clone Cal Set with a different ID. Use this command to manipulate data on a Cal Set without corrupting the original cal data.

**VB Syntax** *CalSet.Copy*

**Variable** [\(Type\)](#) - **Description**

*CalSet* **(object)** - A [Cal Set](#) object

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim mgr As CalManager
Dim ocalset As CalSet
Dim newcalset As CalSet
Set mgr = pna.GetCalManager
'Create a new (empty) Cal Set.
Set ocalset = mgr.CreateCalSet(1)
ocalset.Description = "original calset"
pna.Channel(1).SelectCalSet ocalset.GetGUID, True

'Launch the cal wizard and allow the user to perform the
calibration.
If pna.LaunchCalWizard(False) Then
'If the Launch returns true then the calibration finished.
ocalset.Save

'Copy the Cal Set to the new one.
Set newcalset = ocalset.Copy
newcalset.Description = "copy of original calset"

Else
'If the cal doesn't finish, delete the old Cal Set
'so it isn't taking up unnecessary memory.
mgr.DeleteCalSet ocalset.GetGUID
End If
```

As a result, the programmer can manipulate the data in the new Cal Set and always revert back to the old Cal Set as needed.

**C++ Syntax** HRESULT Copy( ICalSet\*\* pCalSet);

**Interface** ICalSet



## CopyToChannel Method

---

**Description** Sets up another channel as a copy of this object's channel.

**VB Syntax** `chan.CopyToChannel(IChanNum)`

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*IChanNum* **(long integer)** – Number of the channel to become a copy of this channel.

**Return Type** None

**Default** Not Applicable

**Examples**

```
Dim chan As Channel
Set chan = PNAapp.ActiveChannel
Const INEW_CHAN_NUM As Long = 2
chan.CopyToChannel(INEW_CHAN_NUM)
```

**C++ Syntax** `HRESULT CopyToChannel(long IChanNum);`

**Interface** IChannel2

## CreateSParameterEx Method

---

**Description** Creates a new S-Parameter measurement in an existing or new window and specifies the load port for 3-port devices.

**VB Syntax** `app.CreateSParameterEx chan,recvr,source[,loadPort][,window]`

**Variable** [\(Type\)](#) - Description

*app* [Application](#) (object)

*chan* (long integer) - Channel number of the new measurement.

*recvr* (long integer) - Port number of the test port receiver.

*source* (long integer) - Port number of the source.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*loadPort* (long integer) - Port number of the load. Required for reflection measurements of 3-port devices on multiport PNA models.

*window* (long integer) - Optional argument. Choose between 1 and the [maximum number of windows allowed on the PNA..](#) See also [Traces, Channels, and Windows on the PNA](#).  
If unspecified, the S-Parameter will be created in the Active Window.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
app.CreateSParameterEx 1,2,1,1 'Creates a new S21 measurement in
channel 1 and New window(1)
app.CreateSParameterEx 2,1,1,3,1 'Creates a new S11 measurement
on channel 2 with port 3 as the load. Create in the active
window
```

**C++ Syntax** HRESULT CreateSParameterEx(long ChannelNum, long RcvPort, long SrcPort, long LoadPort, long windowNumber)

**Interface** IApplication

Last modified:

24-Apr-2008 Modified string names note

9/12/06 Modified for number of windows



## CreateCalSet Method

---

**Description** Creates a new Cal Set.

The new cal set is initialized with the stimulus settings from the channel whose number is passed as the argument to this method. Stimulus settings include frequency, bandwidth, number of points, and so forth.

Use this method when you want to manually upload data to the Cal Set using the returned ICal Set interface handle..

The channel number does not restrict the usage of this Cal Set on any other channel. It simply provides a link to the originating channel so that the stimulus values can be stored in the Cal Set.

**Note:** Be sure to SAVE the CalSet you are creating. Use [ICalSet::Save](#).

**VB Syntax** *calMgr*.**CreateCalSet** (*chan*)

**Variable** [\(Type\)](#) - **Description**

*calMgr* **(object)** - A [CalManager](#) object

*chan* **(long)** - channel number of the new Cal Set.

**Return Type** ICal Set Interface

**Default** Not Applicable

**Example** `calMgr.CreateCalSet 1`

**C++ Syntax** HRESULT CreateCalSet( long ChannelNumber, ICal Set\*\* pCal Set);

**Interface** ICalManager

## CreateCustomCal Method

---

**Description** Creates a custom cal object.

**VB Syntax** *calmgr*.**CreateCustomCal**(*CalType*)

**Variable** [\(Type\)](#) - **Description**

*calMgr* [Cal Manager](#) (**Object**)

*CalType* (**String**) Name of the calibration. Choose from:

"VMC" or "VectorMixerCal.VMCType"

"SMC" or "ScalarMixerCal.SMCType"

See Also

[SMCType](#) Object

[VMCType](#) Object

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim CalMgr As ICalManager2
Dim SMC As ISMCType
Set SMC = CreateCustomCal("SMC")
```

See [SMC](#) and [VMC](#) examples using this command.

**C++ Syntax** HRESULT CreateCustomCal( BSTR CustomCal)

**Interface** ICalManager2



## CreateCustomCalEx Method

---

**Description** Returns [IGuidedCalibration](#) for the specified channel.

With a handle to the IGuidedCalibration interface, you can query it for the following objects for properties appropriate to the calibration setup for the particular meastype (channel).

IGuidedCalibration interface is used to configure a calibration (specify connectors, cal kits, and so forth). It is also used to access any custom calibration properties required for unique application channels like Noise Figure or Gain Compression. To access these special properties, make this call on the IGuidedCalibration interface:

**CustomInterface = IGuidedCalibration.[CustomCalConfiguration](#)();**

The interface returned by this call can be used to set and get the custom properties on the following application cal objects:

- [NoiseCal Object](#)
- [GainCompressionCal Object](#)
- [SweptIMDCal Object](#)

**Note:** Use [CreateCustomCal Method](#) to create FCA calibration objects.

**VB Syntax** *calmgr*.**CreateCustomCalEx** (*chan*)

**Variable** [\(Type\)](#) - **Description**

*calMgr* [Cal Manager](#) (**Object**)

*chan* (**long integer**) Channel number in which to create the Cal object.

**Return Type** [IGuidedCalibration](#)

**Default** Not Applicable

**Examples**

```
Dim guidedcal
Set guidedcal = CalManager.CreateCustomCalEx(1)
```

**See Also**

- [Noise Figure example](#)
- [Gain Compression example](#)

**C++ Syntax** HRESULT CreateCustomCalEx(long channel, IDispatch\*\* ppObject);

**Interface** ICalManager5

---

Last Modified:

22-Jul-2009 Added description text per SW

29-May-2007 MN New topic

## CreateCustomMeasurementEx Method

---

**Description** Creates a new custom measurement.

**VB Syntax** `app.CreateCustomMeasurementEx chanNum,MeasClass,MeasName [,window]`

**Variable** [\(Type\)](#) - **Description**

*app* **(object)** - An [Application](#) object

*chanNum* **(long)** -Channel number used by the new measurement; can exist or be a new channel.

*MeasClass* **(string)** - Measurement class of the new custom measurement object. The new custom measurement must be installed and registered on the PNA.

Choose from the following:

- "Vector Mixer/Converter"
- "Scalar Mixer/Converter"
- "Gain Compression"
- "Noise Figure Cold Source"
- "Noise Figure Converters"
- "Swept IMD"
- "IM Spectrum"
- "Swept IMD Converters"
- "IM Spectrum Converters"

*MeasName* **(variant)** Measurement names to create:

Meas Class	Measurement Name	Description
Vector Mixer/Converter	"S11"	<a href="#">Learn about VMC parameters</a>
	"VC21"	
	"S22"	
Scalar Mixer/Converter	"S11"	<a href="#">Learn about SMC parameters</a>
	"SC21"	
	"SC12"	
	"S22"	
	"lpwr"	

	<b>"RevIPwr"</b> <b>"Opwr"</b> <b>"RevOPwr"</b>
Gain Compression <a href="#">Learn more</a>	<b>"CompIn21"</b> Input power at the compression point. <b>"CompOut21"</b> Output power at the compression point. <b>"CompGain21"</b> Gain at the compression point. <b>"CompS11"</b> Input Match at the compression point <b>"RefS21"</b> Linear Gain <b>"DeltaGain21"</b> CompGain21 -Linear Gain <b>"S11", "S21", "S12", "S22"</b> Standard S-parameters; measured at port 1 and port 2
	<b>Noise Figure AND NFX:</b>
	<b>"NF"</b> Noise figure
	<b>"ENR"</b> Validate noise source measurements.
	<b>"T-Eff"</b> Effective noise temperature.
	<b>"DUTRNP"</b> DUT noise power ratio. <b>"DUTRNPI"</b> (Noise power expressed in Kelvin divided by 290).
	<b>"SYSRNP"</b> System noise power ratio <b>"SYSRNPI"</b>
	<b>"DUTNPD"</b> DUT noise power density. <b>"DUTNPDI"</b> (Noise power expressed in dBm/Hz).
	<b>"SYSNPD"</b> System noise power density. <b>"SYSNPDI"</b>

Noise Figure Cold Source <a href="#">Learn more</a>  Noise Figure Converters <a href="#">Learn more</a>	"OvrRng"	Indication that the noise receiver is being over powered.
	"T-Rcvr"	Temperature reading (in Kelvin) of the noise receiver board.
	<b>Noise Figure ONLY - NOT NFX:</b>	
	"S11", "S21", "S12", "S22"	Standard S-parameters; measured with the port1 and port2 noise switches set for noise mode.
	"A,1", "A,2" ...and so forth.	Unratioed parameters; with notation: "receiver, source port"
	<b>NFX ONLY:</b>	
	"S11" "SC21" "SC12" "S22" "lpwr" "RevIPwr" "Opwr" "RevOPwr"	Mixer parameters
	"ALO1", "BLO1" ...and so forth.	Test port receiver at LO1 frequency
	"R1_1", "B_2" ..and so forth.	Unratioed parameters with notation: "receiver_source port"
	<p>There are over 150 possible Swept IMD parameters, too many to list here.</p> <p>Build the parameters with the <a href="#">Swept IMD Parameter</a> dialog, then copy the parameter name to the remote command.</p> <p>The following are a few example parameters:</p>	

Swept IMD Swept IMD Converters <a href="#">Learn more</a>	"PwrMainLo"	Absolute power of the Low tone at the DUT output.
	"IM3"	Power of the third product relative to the average power of the f1 and f2 tones measured at the DUT output.
	"OIP3"	Theoretical power level at which the third product will be the same power level as the average of the main tones at the output of the DUT.
IM Spectrum IM Spectrum Converters <a href="#">Learn more</a>	"Output"	View signals OUT of the DUT and into PNA port 2 (B receiver).
	"Input"	View signals IN to the DUT (R1 receiver).
	"Reflection"	View signals reflected off the DUT input and back into PNA port 1 (A receiver)

*window* **(long)** Optional argument. Number of the window the new custom measurement will be placed in. Choose between 1 and the [maximum number of windows allowed on the PNA](#). If unspecified, the measurement is placed in the active window.

**Return Type** IMeasurement

**Default** Not Applicable

**Examples**

```
'To create a scalar mixer measurement in channel 2:
Dim MyMeas as Agilent835x.Measurement
Set MyMeas = app.CreateCustomMeasurementEx (2, "Scalar
Mixer/Converter", "SC21")

'To create a vector mixer measurement in channel 2:
Dim MyMeas as Agilent835x.Measurement
Set MyMeas = app.CreateCustomMeasurementEx (2, "Vector
Mixer/Converter", "VC21")
```

**C++ Syntax** HRESULT put\_CreateCustomMeasurementEx (long ChannelNum, BSTR guid, VARIANT initData, long windowNumber, IMeasurement\*\* ppMeasurement );

**Interface** IApplication3

---

Last Modified:

2-Mar-2010	Fixed IM spectrum converters
29-Oct-2009	Added NFX
23-Feb-2009	Added IMD Converters and Noise Figure I
18-Sep-2008	Added IMD and IM Spectrum
8-Nov-2007	Updated for NF and GCA

## CustomCalConfiguration Property

---

**Description** Calibration for the following PNA Applications is performed using the [GuidedCalibration Object](#). This command provides access to additional Properties and Methods which extends the GuidedCal Object.

Meas Type	Custom Cal Interface
Gain Compression	<a href="#">IGainCompressionCal Object</a>
Noise Figure	<a href="#">INoiseCal Object</a>
Swept IMD	<a href="#">ISweptIMD Cal Object</a>

**VB Syntax** `set custCal = cal.CustomCalConfiguration()`

**Variable** [\(Type\)](#) - **Description**

*custCal* **(object)** The handle to an interface that provides application-specific calibration properties..

*cal* [GuidedCalibration Object](#).

**Return Type** Depends on the MeasType. See above table.

**Default** None

**Examples** See examples: [NoiseFigure](#) [GainCompression](#) [SweptIMD](#)

**C++ Syntax** HRESULT CustomCalConfiguration(IDispatch\*\* value);

**Interface** IGuidedCalibration4

---

Last Modified:

22-Jul-2009 Updated description

8-Sep-2008 MX New topic



## CreateMeasurement Method

---

**Description** Creates a new measurement.

**VB Syntax** `app.CreateMeasurement chanNum,param,lPort[,window]`

**Variable** [\(Type\) - Description](#)

*app* Application (**object**)

*chanNum* (**long**) - Channel number of the new measurement; can exist or be a new channel

*param* (**string**) - New parameter. Case insensitive.

### For S-parameters:

[Any S-parameter that can be measured by your PNA.](#)

Single-digit port numbers can be separated by "\_" (underscore). For example: "S21" or "S2\_1"

Double-digit port numbers MUST be separated by underscore. For example: "S10\_1"

### For Ratioed measurements:

Any two receivers in your PNA separated by "/". For example: "A/R1"

See the [block diagram](#) showing the receivers in YOUR PNA.

### For Unratioed (absolute power) measurements:

Any receiver in the PNA. For example: "A"

See the [block diagram](#) showing the receivers in YOUR PNA

With PNA Rev 6.2, **Ratioed** and **Unratioed** measurements can also use **logical receiver notation** to refer to receivers. This notation makes it easy to refer to receivers with an [external test set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port. [Learn more.](#)

### For ADC measurements

Any ADC receiver in the PNA.

For example: "AI1" indicates the Analog Input1.

[Learn more about ADC receiver measurements.](#)

### For Balanced S-parameter measurements:

"topology:Sabxy"

*topology* - Choose from:

- **sbal** - single-ended to balanced

- **ssb** - single-ended / single-ended to balanced
- **bbal** - balanced to balanced

**Sabxy** -

**Where**

**a** - device output (receive) mode

**b** - device input (source) mode

(choose from the following for both a and b:)

- **d** - differential
- **c** - common
- **s** - single ended

**x** - device output (receive) logical port number

**y** - device input (source) logical port number

For example: "**sbal:sdd42**"

[See an example program](#)

**For Imbalance and Common Mode Rejection measurements:**

"*topology:parameter*" Choose from:

Choose this:	To get this:	
	Topology	Parameter
"SBAL:IMBSB"	single-ended to balanced	imbalance
"SBAL:CMRRSB1"	single-ended to balanced	common mode rejection (Sds21/Scs21)
"SBAL:CMRRSB2"	single-ended to balanced	common mode rejection (Ssd12/Scs12)
"SSB:IMB1SSB"	single-ended / single-ended to balanced	imbalance 1
"SSB:IMB2SSB"	single-ended / single-ended to balanced	imbalance 2
"SSB:CMRRSSB1"	single-ended / single-ended to balanced	common mode rejection (Sds31/Scs31)

"SSB:CMRRSSB2"	single-ended / single-ended to balanced	common mode rejection (Sds32/Scs32)
"BBAL:IMB1BB"	balanced to balanced	imbalance 1
"BBAL:IMB2BB"	balanced to balanced	imbalance 2
"BBAL:CMRRBB"	balanced to balanced	common mode rejection (Sdd21/Scs21)

*lPort* (long)

- **Ignored** if *param* is an S-Parameter, balanced, imbalance, or CMRR parameter.
- **Source port** if *param* is ratioed or unratioed (including [ADC](#)) measurements.
- Use **0** for [N5264A](#).

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*window* (long) Optional argument. Window number of the new measurement. Choose between 1 and the [maximum number of windows allowed on the PNA](#).. If unspecified, the measurement will be created in the Active Window.

See also [Traces, Channels, and Windows on the PNA](#)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
app.CreateMeasurement(1, "A/R1", 1, 0)
app.CreateMeasurement(1, "a1/b1", 1, 0)
app.CreateMeasurement(1, "bba1:Sdd21", 1)
app.CreateMeasurement(1, "AI2", 2)
app.CreateMeasurement(1, "R1", 0) ' for N5264A
```

**C++ Syntax** HRESULT CreateMeasurement(long ChannelNum, BSTR strParameter, long lPort, long windowNumber)

**Interface** IApplication

Last modified:

23-Oct-2008	Added support for N5264A
24-Apr-2008	Clarify ports with string names
23-Jul-2007	Added source port link
25-Apr-2007	Updated for ADC measurements.
12-Sept-2006	MQ Updated for logical receiver notation.

## DataToMemory Method

---

<b>Description</b>	Stores the active measurement data into memory creating a memory trace. The memory can then be displayed or used in calculations with the measurement data.
<b>VB Syntax</b>	<i>meas</i> . <b>DataToMemory</b>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>meas</i>	A Measurement ( <b>object</b> )
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>meas.DataToMemory</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT DataToMemory()
<b>Interface</b>	IMeasurement

## Delete Method

---

**Description** Deletes the measurement.

**VB Syntax** *meas.Delete*

**Variable** [\(Type\)](#) - Description

*meas* The Measurement object to delete (**object**)

**Return  
Type** Not Applicable

**Default** Not Applicable

**Examples** `meas.Delete`

**[C++ Syntax](#)** HRESULT Delete()

**Interface** IMeasurement

## DeleteMarker Method

---

**Description** Deletes a marker from the measurement.

**VB Syntax** *meas.DeleteMarker(Mnum)*

**Variable** [\(Type\)](#) - Description

*meas* A Measurement (**object**)

*Mnum* (**long**) - Any existing marker number in the measurement

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `meas.DeleteMarker(1)`

**C++ Syntax** HRESULT DeleteMarker(long IMarkerNumber)

**Interface** IMeasurement

## DeleteAllMarkers Method

---

**Description** Deletes all of the markers from the measurement.

**VB Syntax** *meas*.DeleteAllMarkers

**Variable** [\(Type\)](#) - Description

*meas* **(object)** - The Measurement object from which markers will be deleted.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `meas.DeleteAllMarkers`

**C++ Syntax** HRESULT DeleteAllMarkers()

**Interface** IMeasurement



## DeleteCalSet Method

---

**Description** Deletes a Cal Set from the set of available Cal Sets. This method immediately updates the Cal Set file on the hard drive. If the Cal Set is currently being used by a channel or does not exist, this request will be denied and an error is returned.

Using the [Cal Sets collection](#) is a convenient way to manage Cal Sets.

**VB Syntax** *calMgr.DeleteCalSet (calset)*

**Variable** [\(Type\)](#) - Description

*calMgr* **(object)** - A [CalManager](#) object

*calset* **(string)** - Cal Set to be deleted. Specify the Cal Set by **GUID** or **Name**. Use [EnumerateCalSets](#) to list the available Cal Sets by name.

**Return Type** Not Applicable

**Default** Not Applicable

**Example**

```
Set pna=CreateObject("AgilentPNA835x.Application")
Set cmgr = pna.GetCalManager
cmgr.DeleteCalSet ("MyCalSet")
```

**C++ Syntax** HRESULT DeleteCalSet( BSTR strCalset);

**Interface** ICalManager

---

Last Modified:

6-Mar-2008    Added Name argument

## DeleteConfiguration Method

---

**Description** Deletes the specified configuration name from the PNA. The factory configurations cannot be deleted. This is the only method of programmatically distinguishing a factory configuration from a user-named configuration.

**VB Syntax** *pathMgr.DeleteConfiguration name*

**Variable** [\(Type\)](#) - Description

*pathMgr* [PathConfigurationManager](#) (object)

*name* (String) Configuration name to be deleted.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `path.DeleteConfiguration "myMixer"`

**C++ Syntax** HRESULT StoreConfiguration (long channelNum, BSTR configName );

**Interface** IPathConfigurationManager

---

Last Modified:

14-Dec-2006    MX New topic

## DeleteShortCut Method

---

**Description** Removes a macro from the list of macros in the analyzer. Does not remove the file.

**Note:** There are always 12 macro positions. They do not have to be sequential. For example, you can have number 7 but no numbers 1 to 6.

**VB Syntax** `app.DeleteShortCut item`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*item* **(long integer)** number of the macro to be deleted.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.DeleteShortCut 2`

**C++ Syntax** HRESULT DeleteShortcut(long Number )

**Interface** IApplication

## DisallowAllEvents Method

---

**Description** Sets event filtering to monitor NO eventst.

**VB Syntax** *app*.DisallowAllEvents

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return  
Type** Not Applicable

**Default** Not Applicable

**Examples** `app.DisallowAllEvents`

**[C++ Syntax](#)** HRESULT DisallowAllEvents()

**Interface** IApplication

Write only

## DiscardChanges Method

---

**Description** Cancels changes that have been made to the Converter setup and reverts to the previously-saved setup. Same as the **Cancel** button on the [mixer setup dialog box](#).

**VB Syntax** *conv*.**DiscardChanges**

**Variable** [\(Type\)](#) - Description

*conv* A [Converter Object](#)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `conv.DiscardChanges`

**C++ Syntax** HRESULT DiscardChanges();

**Interface** IConverter

---

Last Modified:

2-Feb-2009    New topic

## DisplayNAWindowDuringCalAcquisition Method

---

<b>Description</b>	<p>Set the 'show' state of the window to be displayed during a calibration.</p> <p>When this command is sent, the specified window is 'flagged' to be shown during calibration. The flag is cleared when the window is closed. A Preset or Instrument State Recall also closes the window. If the same window number is reopened, this command must be sent again to show the window during a calibration. The flag is NOT saved with an instrument state.</p> <p>Send this command for each additional window to show during a calibration.</p>
<b>VB Syntax</b>	<code>calMgr.DisplayNAWindowDuringCalAcquisition (winNum, State)</code>
<b>Variable</b>	<b>(Type) - Description</b>
<i>calMgr</i>	<b>(object)</b> - A <a href="#">CalManager</a> object
<i>winNum</i>	<b>(long)</b> - Window number to show during a calibration. The calibration window will also be shown with this window.
	The window must already be created.
	Use <a href="#">NaWindows.count</a> or <a href="#">app.WindowNumber</a> to read existing window numbers.
<i>state</i>	<b>(Boolean)</b> Window state. Choose from:
	<b>True</b> - Show the specified window during calibration.
	<b>False</b> - Do NOT show the specified window during calibration.
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Example</b>	<pre>calMgr.DisplayNAWindowDuringCalAcquisition 2,True</pre> <p><a href="#">See example using this command</a></p>
<b>C++ Syntax</b>	HRESULT DisplayNAWindowDuringCalAcquisition( long WinNum, VARIANT_BOOL bVal);
<b>Interface</b>	ICalManager5

---

Last Modified:

28-Jan-2009    Removed 'Read'

8-Nov-2007    MX New topic

## DisplayOnlyCalWindowDuringCalAcquisition Method

---

<b>Description</b>	Clears the flags for windows to be shown during calibrations other than the Cal Window. To flag a window to be shown see <a href="#">DisplayNAWindowDuringCalAcquisition</a>
<b>VB Syntax</b>	<i>calMgr</i> . <b>DisplayOnlyCalWindowDuringCalAcquisition</b>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
	<i>calMgr</i> <b>(object)</b> - A <a href="#">CalManager</a> object
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Example</b>	<pre>calMgr.DisplayOnlyCalWindowDuringCalAcquisition</pre> <a href="#">See example using this command</a>
<b>C++ Syntax</b>	HRESULT DisplayOnlyCalWindowDuringCalAcquisition()
<b>Interface</b>	ICalManager5

---

Last Modified:

8-Nov-2007    MX New topic

## DoPrint Method

---

**Description** Prints the screen to the default Printer.

**VB Syntax** *app*.DoPrint

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (**object**)

**Return  
Type** Not Applicable

**Default** Not Applicable

**Examples** `app.DoPrint`

**[C++ Syntax](#)** HRESULT DoPrint()

**Interface** IApplication



## DoECAL1PortEx Method

---

**Description** This method replaces [DoECAL1Port Method](#).  
 Does a 1-Port calibration using an ECAL module. You must first have a 1-port measurement active to perform the calibration.  
 The characterization within the ECal module that will be used for the calibration is specified by [ECALCharacterizationEx](#). The default value is 0.

**VB Syntax** *cal.DoECAL1PortEx [port][,module]*

---

### Variable [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

*port* (**long integer**) Optional argument - Port number to calibrate. Choose from:  
 1 - Calibrate port 1 (default if unspecified)  
 2 - Calibrate port 2

*module* (**long integer**) Optional argument. ECal module.  
 Choose from modules **1** through **8**  
 Use [IsECALModuleFoundEx](#) to determine the number of modules connected to the PNA  
 Use [GetECALModuleInfoEx](#) to returns the model and serial number of each module.

**Return Type** None

**Default** Not Applicable

---

**Examples** `cal.DoECAL1PortEx, 2, 2`

---

**C++ Syntax** HRESULT DoECAL1PortEx(long port, long moduleNumber = 1);

**Interface** ICalibrator4

## DoECAL2PortEx Method

---

<b>Description</b>	<p>This method replaces <a href="#">DoECAL2Port Method</a>.</p> <p>Does a 2-port calibration using an ECal module.</p> <p>2-port refers to the number of ports to calibrate; NOT to the number of ECal module ports.</p> <p>You must first have a measurement active to perform the calibration.</p> <p>The characterization within the ECal module that will be used for the calibration is specified by <a href="#">ECalCharacterizationEx</a>. The default value is 0.</p>
<b>VB Syntax</b>	<i>cal</i> .DoECAL2PortEx [ <i>portA</i> ][ <i>,portB</i> ][ <i>,module</i> ]
<b>Variable</b>	<b>(Type) - Description</b>
<i>cal</i>	A Calibrator ( <b>object</b> )
<i>portA</i>	<p><b>(long integer)</b> Optional argument - Number of the receive port to calibrate. Choose from:</p> <ul style="list-style-type: none"> <li>1 - Calibrate port 1 (default, if unspecified)</li> <li>2 - Calibrate port 2</li> <li>3 - Calibrate port 3</li> </ul> <p>And so forth for all available PNA / test set ports.</p>
<i>portB</i>	<p><b>(long integer)</b> Optional argument - Number of the source port to calibrate. Choose from:</p> <ul style="list-style-type: none"> <li>1 - Calibrate port 1</li> <li>2 - Calibrate port 2 (default, if unspecified)</li> <li>3 - Calibrate port 3</li> </ul> <p>And so forth for all available PNA / test set ports.</p>
<i>module</i>	<p><b>(long integer)</b> Optional argument. ECal module.</p> <p>Choose from modules <b>1</b> through <b>8</b></p> <p>Use <a href="#">IsECALModuleFoundEx</a> to determine the number of modules connected to the PNA</p> <p>Use <a href="#">GetECALModuleInfoEx</a> to returns the model and serial number of each module.</p>
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<i>cal</i> .DoECAL2PortEx, 1, 2, 3
<b>C++ Syntax</b>	HRESULT DoECAL2PortEx( long portA = 1, long portB =2, long moduleNumber = 1);
<b>Interface</b>	ICalibrator4

---

Last Modified:

1-Jan-2007    Corrected Port B default

## DoneCalConfidenceCheckECAL Method

---

**Description** Concludes the Confidence Check and sets the ECal module back into the idle state.

**VB Syntax** *cal*.DoneCalConfidenceCheckECAL

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator (**object**)

**Return Type** None

**Default** None

**Examples** *cal*.DoneCalConfidenceCheckECAL

**[C++ Syntax](#)** HRESULT DoneCalConfidenceCheckECAL( );

**Interface** ICalibrator

## DoReceiverPowerCal Method

---

<b>Description</b>	<p><b>Note:</b> This command replaces <a href="#">DataToDivisor</a>, <a href="#">LogMagnitudeOffset</a>, <a href="#">Normalization</a>, <a href="#">InterpolateNormalization</a>.</p> <p>Immediately performs a receiver power calibration. The connection to the receiver must be in place when this command is sent.</p> <p>A Receiver Power Cal requires that the active measurement be an <a href="#">Unratioed</a> power measurement.</p>
<b>VB Syntax</b>	<code>cal.DoReceiverPowerCal(param, srcPort [,pwrOffset])</code>
<b>Variable</b>	<b>(Type) - Description</b>
<i>cal</i>	A <a href="#">Calibrator</a> (object)
<i>param</i>	<p><b>(string)</b> – Receiver to be calibrated. Choose any receiver in your PNA. <a href="#">See a block diagram of your PNA.</a></p> <p>With PNA Rev 6.2, receivers can also be referred to using <b>logical receiver notation</b>. This notation makes it easy to refer to receivers with an <a href="#">external test set</a> connected to the PNA. You do not need to know which physical receiver is used for each test port. <a href="#">Learn more.</a></p>
<i>srcPort</i>	<p><b>(long integer)</b> – Number of the port which will supply source power to the receiver during this cal.</p> <p><b>Note:</b> If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use <a href="#">chan.getPortNumber</a> to translate the string into a port number. To learn more see <a href="#">Remotely Specifying a Source Port</a>.</p>
<i>pwrOffset</i>	<p><b>(double)</b> – Optional argument. Offset value in dB. Adjusts a receiver power cal to account for components or adapters that are added between the source port and receiver while performing this cal. Specify loss as a negative number; and gain as a positive number.</p>
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>cal.DoReceiverPowerCal "B", 1, -10</code>
<b>C++ Syntax</b>	<code>HRESULT DoReceiverPowerCal(BSTR parameter, long lSrcPort, double dPowerOffset);</code>
<b>Interface</b>	ICalibrator5

---

Last Modified:

24-Apr-2008 Added note for string names

30-Apr-2007 Edited for src strings

## EnumerateCalSets Method

---

**Description** Returns an array of Cal Set names being stored on the PNA.

**VB Syntax** *value* = *calMgr*.EnumerateCalSets

**Variable** [\(Type\)](#) - Description

*value* **(variant)** - Variable to store the returned Cal Set names

*calMgr* **(object)** - A [CalManager](#) object

**Return Type** VARIANT array

**Default** Not Applicable

**Example**

```
Dim pna
set
pna=CreateObject ("AgilentPNA835x.Application")

Dim catalog
catalog=pna.getcalmanager.EnumerateCalSets
for i=lbound(catalog) to Ubound(catalog)
wscript.echo catalog(i)
next
```

**[C++ Syntax](#)** HRESULT EnumerateCalSets(VARIANT\* names);

**Interface** ICalManager4

## Execute Method

---

**Description** Allows the use of COM to send a SCPI command.

This method can be used with :SYST:ERR? to convert scpi errors into text.

[See an example](#) of how to return error information when using the [Parse method](#).

**Note:** The SCPIStringParser Methods can NOT be used with [SCPI Status Reporting](#). However, the \*OPC? will work.

**VB Syntax** Scpi.**Execute** (*SCPI\_Command*)

**Variable** [\(Type\)](#) - **Description**

*scpi* A [ScpiStringParser](#) (**Object**)

*SCPI\_Command* (**String**) - Any valid SCPI command

**Return Type** String

**Default** Not Applicable

**Examples**

```
Dim scpi As ScpiStringParser
Set scpi = app.ScpiStringParser
scpi.Execute("SYST:PRES");
ErrorString = scpi.Execute("SYST:ERRor?");
```

**C++ Syntax** Execute(BSTR SCPI\_Command, BSTR \* pQueryResponse);

**Interface** IScpStringParser2

---

Last Modified:

27-Apr-2009    Added note



## ExecuteShortcut Method

---

**Description** Executes a Macro (shortcut) stored in the analyzer. Use [app.getShortcut](#) to list existing macros. Use app.[putShortcut](#) to associate the macro number with the file.

**VB Syntax** *app.ExecuteShortcut index*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*index* (**long integer**) - Number of the macro stored in the analyzer.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.ExecuteShortcut 1`

**C++ Syntax** HRESULT ExecuteShortcut(long index)

**Interface** IApplication

## GenerateGlobalDeltaMatchSequence Method

---

**Description** Initiates a global delta match calibration.

[Learn more about Delta match calibration.](#)

See example of a complete Delta Match calibration.

**VB Syntax** *numSteps* = *guided*.**GenerateGlobalDeltaMatchSequence** *conn*,*cKit*

**Variable** [\(Type\)](#) - **Description**

*numSteps* **Long Integer** - Variable to store the returned number of connection steps required by the Global Delta Match Cal.

*guided* [GuidedCalibration](#) (object)

*conn* **String** Connector Type for port 1.

*cKit* **String** Cal Kit for all ports.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `guided.GenerateGlobalDeltaMatchSequence "APC 3.5 female", "85052B"`

**C++ Syntax** HRESULT GenerateGlobalDeltaMatchSequence(BSTR port\_1\_conn, BSTR cal\_kit, long \*num\_steps);

**Interface** IGuidedCalibration2

Write-only

## GenerateErrorTerms Method

---

**Description** Generates the error terms for the specified calibration type, stores the error terms in a Cal Set, saves the Cal Set, and returns the Cal Set GUID.  
If ALL the data for the cal type has NOT been acquired an error message is returned.  
**Note:** The manner in which the calibration is assigned to a Cal Set (Cal Register or User Cal Set) is determined by the setting of [RemoteCalStoragePreference](#).

**VB Syntax** *value* = *obj*.GenerateErrorTerms

**Variable** [\(Type\)](#) - Description

*value* (String) - Variable to store the returned GUID or error message.

*obj* Any of the following:  
[GuidedCalibration](#) (object)  
[SMCType](#) (object)  
[VMCType](#) (object)

**Return Type** String

**Default** Not Applicable

**Examples** `string = SMC.GenerateErrorTerms`

**C++ Syntax** HRESULT GenerateErrorTerms(BSTR\* calsetGUID);

**Interface** IGuidedCalibration  
SMCType  
VMCType

---

Last Modified:

28-Jan-2009 Fixed C++ Syntax

## GenerateSteps Method

---

<b>Description</b>	<p>Returns the number of steps required to complete the calibration.</p> <p>For an ECal User Characterization this generate steps for the ECal User Characterization process. The channel must already be calibrated using the same, or greater number of PNA ports as the ECal module. Also, the PNA ports must begin with Port 1 and use sequential port numbers.</p> <p>After this command is executed, subsequent commands can be used to query the number of measurement steps, issue the acquisition commands, query the connection description strings, and subsequently complete a User Characterization calibration.</p>
<b>VB Syntax</b>	<i>value</i> = <i>obj</i> . <b>GenerateSteps</b>
<b>Variable</b>	<b>(Type) - Description</b>
<i>value</i>	(long) - Variable to store the returned number of steps
<i>obj</i>	Any of the following: <a href="#">GuidedCalibration</a> (object) <a href="#">SMCType</a> (object) <a href="#">VMCType</a> (object) <a href="#">ECalUserCharacterizer</a> (object)
<b>Return Type</b>	Long
<b>Default</b>	Not Applicable
<b>Examples</b>	<b><code>value = SMC.GenerateSteps</code></b>
<b>C++ Syntax</b>	HRESULT put_GenerateSteps(long* steps);
<b>Interface</b>	IGuidedCalibration SMCType VMCType IECalUserCharacterizer

---

Last Modified:

4-Nov-2008    Added ECal object

## GetAllSegments Method

---

**Description** Downloads a segment table from the PNA.

**VB Syntax** *segdata* = Segs.**GetAllSegments**

**Variable** [\(Type\)](#) - **Description**

*segs* A [Segments](#) (**Collection**)

*segdata* (Variant) A 2-dimensional array of Segment data:

- Dimension 0 is the number of elements in each segment.
- Dimension 1 is the number of segments that will be used.

All elements in the returned array are Variant. The type inside each Variant will be as is listed below.

The returned array will contain values for all elements regardless of the settings of [IFBandwidthOption](#), [SweepTimeOption](#), [SourcePowerOption](#) and [CouplePorts](#) properties. Ignore the values for the properties that are set to false.

The following is a list of dimension 0 elements for each segment:

- 0 = Segment state (Boolean True or False)
- 1 = Number of Points in this segment (Integer)
- 2 = Start Freq (Double)
- 3 = Stop Freq (Double)
- 4 = IFBW (Double)
- 5 = Dwell Time (Double)
- 6 + N = Power (Double) where N is the number of source ports of the PNA. For example, with a 4-port, 1-source PNA, indices 6 through 9 correspond to the per-segment power levels for Ports 1 to 4. Use [SourcePortCount Property](#) and [SourcePortNames Property](#) to see the available source ports for the PNA.

**Return Type** Variant, containing an array.

**Default** Not Applicable

**Examples** [See a VB example using this command](#)

[C++ Syntax](#) HRESULT GetAllSegments (VARIANT \*pSegments );

[Interface](#) ISegments5

---

Last Modified:

28-Apr-2009    MX New topic

## GetAuxIO Method

---

**Description** This method returns the [IAuxIO](#) interface.

**VB Syntax** *app*.GetAuxIO

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return  
Type** IHWAuxIO

**Default** Not Applicable

**Example**

```
Dim app As AgilentPNA835x.Application
Dim aux As HWAuxIO
Set aux = app.GetAuxIO
```

**C++ Syntax** HRESULT GetAuxIO (IHWAuxIO \*\*pAux);

**Interface** IApplication

## GetCalStandard Method

---

<b>Description</b>	Returns a handle to a calibration standard for modifying its definitions. To select a standard for performing a calibration (use Calibrator. <a href="#">AquireCalStandard</a> ).
<b>VB Syntax</b>	<i>calKit</i> . <b>GetCalStandard</b> ( <i>index</i> )
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>calKit</i>	A calKit ( <b>object</b> )
<i>index</i>	<b>(long)</b> - Number of calibration standard. Choose <b>1</b> to <b>30</b> ; (there are 30 cal standards in every kit).
<b>Return Type</b>	calStandard
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim short As CalStandard Set short = calKit.getCalStandard(1) short.label = "myShort"</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT GetCalStandard(long standardNumber, ICalStandard **pCalStd)
<b>Interface</b>	ICalKit



## GetCalManager Method

---

**Description** This method returns the [ICalManager](#) interface.

**VB Syntax** *app*.GetCalManager()

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** ICalManager\*

**Default** Not Applicable

**Example**

```
dim app as AgilentPNA835x.Application
dim mgr as CalManager
set mgr = app.GetCalManager()
```

**C++ Syntax** HRESULT GetCalManager( ICalManager \*\*mgr);

**Interface** IApplication

---

Last Modified:

30-Jan-2008    Added parenthesis

## Get CalSetByGUID Method

---

**Description** Requests a Cal Set by GUID. Returns an ICal Set interface.

**VB Syntax** *calMgr*.GetCalSetByGUID (*GUID*)

**Variable** [\(Type\)](#) - Description

*calMgr* **(object)** - A CalManager object

*GUID* **(string)** - GUID of the Cal Set being requested.

**Return Type** Interface object

**Default** Not Applicable

**Example** `calMgr.GetCalSetByGUID (2B893E7A-971A-11d5-8D6C-00108334AE96)`

**C++ Syntax** HRESULT GetCalSetByGUID( BSTR\* strGUID, ICal Set\* pCalSet);

**Interface** ICalManager

## GetCalSetCatalog Method - Superseded

---

<b>Description</b>	This method is replaced with <a href="#">EnumerateCalSets</a> Returns a string containing a list of comma-separated GUIDs in the following format: {FD6F863E-9719-11d5-8D6C-00108334AE96}, {1B03B2CE-971A-11d5-8D6C-00108334AE96}, {2B893E7A-971A-11d5-8D6C-00108334AE96}
<b>VB Syntax</b>	<i>value</i> = <i>calMgr</i> . <b>GetCalSetCatalog</b>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>value</i>	<b>(string)</b> - Variable to store the returned GUID list
<i>calMgr</i>	<b>(object)</b> - A CalManager object
<b>Return Type</b>	String
<b>Default</b>	Not Applicable
<b>Example</b>	<code>value = calMgr.GetCalSetCatalog</code>
<b>C++ Syntax</b>	HRESULT GetCalSetCatalog( BSTR);
<b>Interface</b>	ICalManager

---

Last Modified:

6-Mar-2008    Superseded

## GetCalSetUsageInfo Method

---

<b>Description</b>	<p>Returns a string identifying the Cal Set currently in use by the specified channel.</p> <p>This method identifies the Cal Set being used by returning its GUID.</p> <p>This method also identifies the "Error Term set" within the Cal Set.</p> <p>Error term sets are identified by integers, with set 0 belonging to the original (non-interpolated) terms. As stimulus values for a channel are changed causing interpolation to be required, a new Error Term set is constructed within the Cal Set to hold the interpolated Error Terms. The sets are sequentially numbered 1, 2, 3, and so forth. These Error Term sets are destroyed when they are no longer being used.</p> <p>If there is no Cal Set in use for the given channel, the &lt;GUID&gt; argument is set to the empty string.</p>
<b>VB Syntax</b>	<i>calMgr</i> . <b>GetCalSetUsageInfo</b> ( <i>chan</i> , <i>GUID</i> , <i>setNumber</i> )
<b>Variable</b>	<a href="#">(Type) - Description</a>
<i>calMgr</i>	<b>(object)</b> - A CalManager object
<i>chan</i>	<b>(long [in])</b> - channel of the Cal Set being requested
<i>GUID</i>	<b>(string [out])</b> - variable to store the GUID of the Cal Set being requested. If there is no Cal Set in use for the given channel, the <GUID> argument is set to the empty string.
<i>setNumber</i>	<b>(long [out])</b> - variable to store the error term ID being requested. If the returned argument is greater than 0, the set is being interpolated.
<b>Return Type</b>	String , Long Integer
<b>Default</b>	Not Applicable
<b>Example</b>	<code>calMgr.GetCalSetUsageInfo (1, GUID, EtermID)</code>
<b>C++ Syntax</b>	HRESULT GetCalSetUsageInfo (long IChannel, BSTR* CalSetGUID, long* etermSetID);
<b>Interface</b>	ICalManager

## GetCalTypes Method

---

**Description** Returns a list of available calibration types known to the PNA. The Standard CalTypes are the same on all PNA's, but the Custom CalTypes are not necessarily the same. They are dependent on the custom measurement in the PNA. [Learn more about applying Cal Types.](#)

See also [CalibrationTypeID](#) to apply a Cal Type containing in a Cal Set.

**VB Syntax** `v = mgr.GetCalTypes`

**Variable** [\(Type\)](#) - Description

*mgr* A [CalManager](#) (Object)

*v* Name/GuidPair that contains the calibration type name and associated GUID for each cal type known to the PNA.

**Return Type** (variant) Two dimensional array.

**Default** Not Applicable

**Examples** `v =CalManager.GetCalTypes`

**C++ Syntax** HRESULT GetCalTypes( VARIANT \* NameGuidPair )

**Interface** ICalManager2

## GetComplex Method

---

<b>Description</b>	Retrieves complex data from the specified location. See also <a href="#">getNAComplex</a> , <a href="#">getData</a> , and <a href="#">getPairedData</a> Methods
<b>VB Syntax</b>	<i>measData</i> . <b>getComplex</b> <i>location</i> , <i>numPts</i> , <i>real()</i> , <i>imag()</i>
<b>Variable</b>	<a href="#">(Type) - Description</a>
<i>measData</i>	An IArrayTransfer interface which supports the Measurement object
<i>location</i>	<b>(enum NADataStore - IArrayTransfer)</b> - Where the data you want is residing. Choose from: 0 - naRawData 1 - naCorrectedData 2 - naMeasResult 3 - naRawMemory 4 - naMemoryResult 5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using <a href="#">DataToDivisor Method</a> .
<i>numPts</i>	<b>(long integer)</b> - Number of data points requested [out] - specifies number of data elements returned [in] - specifies the data being requested or the capacity of the arrays
<i>real</i>	<b>(single)</b> - Array to store the real values
<i>imag</i>	<b>(single)</b> - Array to store the imaginary values
<b>Return Type</b>	Single
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim real(201) AS Single Dim imag(201) AS Single Dim pts as Integer Dim measData As IArrayTransfer Set measData = app.ActiveMeasurement measData.getComplex naCorrectedData, pts, real(0), imag(0)</pre>
<b>C++ Syntax</b>	IArrayTransfer - HRESULT getComplex(tagNADataStore DataStore, long* pNumValues, float* pReal, float* plmag)
<b>Interface</b>	IArrayTransfer



## GetConverter Method

---

**Description** This method returns a handle to a [Converter](#) object.

**VB Syntax** *chan*.GetConverter( )

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

**Return Type** IConverter

**Default** Not Applicable

**Example**

```
Dim app as AgilentPNA835x.Application
Set app =
CreateObject("AgilentPNA835x.Application")
Dim chan As IChannel
Set chan = app.ActiveChannel
Dim convert
Set convert = chan.GetConverter
```

**C++ Syntax** HRESULT GetConverter( IConverter \*\*obj);

**Interface** IChannel

---

Last Modified:

2-Feb-2009    New topic



## getDataByString Method

---

**Description** Retrieves variant data from the specified location in your choice of formats.

The PNA returns complex trace data which is ratioed if required by the measurement parameter, such as S11 or A/B. Otherwise it is raw receiver data, such as A or B.

Equation Editor Notes:

- When equation editor is active on a trace in a standard S-parameter channel, GetData returns the data from the parameter on the trace that was measured last. For example, for the equation "S22 + S33 + S11", then S33 is the last measured parameter because it uses source port 3.
- In [applications](#), if equation editor is active and the original parameter for the trace is not requested anywhere in the channel, then zeros are returned. If the original parameter is being measured within the channel, then data for the original parameter is returned.
- In general, if an equation contains no measurement parameters, then data for the original parameter is returned.

**VB Syntax** `data = meas.getDataByString location, format`

**Variable** [\(Type\)](#) - **Description**

*data* **(variant)** - Array to store the data.

*meas* **(object)** - A Measurement object

*location* **(string)** – Name of the buffer to be read. Choose from:

"naRawData"

"naCorrectedData"

"naMeasResult"

"naRawMemory"

"naMemoryResult"

"naDivisor" - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

See [Data Access Map](#)

*format* **(enum NADDataFormat)** - Format in which you would like the data. It does not have to be the displayed format. Choose from:

0 - naDataFormat\_LinMag

1 - naDataFormat\_LogMag

2 - naDataFormat\_Phase

- 3 - naDataFormat\_Polar
- 4 - naDataFormat\_Smith
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

[Learn more about Data Format.](#)

\* Specify **Smith** or **Polar** formats to obtain complex data pairs, which require a two-dimensional array **varData (numpts, 2)** to accommodate both real and imaginary data.

All scalar formats return a single dimension **varData(numpts)**.

**Return Type** Variant array

**Default** Not Applicable

**Examples** `meas.getDataByString "naMeasResult", naDataFormat_Phase`

**C++ Syntax** HRESULT getDataByString( BSTR location, tagDataFormat dataFormat, VARIANT \* pData );

**Interface** IMeasurement

---

Last Modified:

- 11-Jun-2009 Added EE notes
- 1-Oct-2007 Added temperature formats
- 19-Jul-2007 Corrected example

## GetData Method

---

**Description** Retrieves variant data from the specified location in your choice of formats. To get smoothed data from any of the specified locations, the format must be the same as the displayed format.

The PNA returns complex trace data which is ratioed if required by the measurement parameter, such as S11 or A/B. Otherwise it is raw receiver data, such as A or B.

This method returns a variant which is less efficient than methods available on the [IArrayTransfer interface](#).

If you plan to **Put** this data back into analyzer, [putDataComplex](#) (variant data) method requires complex, two-dimensional data. Therefore, request the data in **Polar** format.

Equation Editor Notes:

- When equation editor is active on a trace in a standard S-parameter channel, GetData returns the data from the parameter on the trace that was measured last. For example, for the equation "S22 + S33 + S11", then S33 is the last measured parameter because it uses source port 3.
- In [applications](#), if equation editor is active and the original parameter for the trace is not requested anywhere in the channel, then zeros are returned. If the original parameter is being measured within the channel, then data for the original parameter is returned.
- In general, if an equation contains no measurement parameters, then data for the original parameter is returned.

**VB Syntax** *data = meas.GetData location, format*

**Variable** [\(Type\)](#) - **Description**

*data* Variant array to store the data.

*meas* A Measurement (**object**)

*location* (**enum NADataStore**) - Where the data you want is residing. See [Data Access Map](#). Choose from:

0 - naRawData

1 - naCorrectedData

2 - naMeasResult

3 - naRawMemory

4 - naMemoryResult

5 - naDivisor When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

*format* (**enum NADataFormat**) - Format in which you would like the data. It does not have to be the displayed format. Choose from:

- 0 - naDataFormat\_LinMag
- 1 - naDataFormat\_LogMag
- 2 - naDataFormat\_Phase
- 3 - naDataFormat\_Polar\***
- 4 - naDataFormat\_Smith\***
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

[Learn more about Data Format.](#)

\* **Specify Smith or Polar** formats to obtain complex data pairs, which require a two-dimensional array **varData (numpts, 2)** to accommodate both real and imaginary data.

All scalar formats return a single dimension **varData(numpts)**.

**naDataFormat\_Phase** and **naDataFormat\_PhaseUnwrapped** returns degrees.

However, [putDataScalar](#) method accepts data in radians (not degrees) and displays in degrees.

**Return Type** Variant array - automatically dimensioned to the size of the data

**Default** Not Applicable

**Examples**

```
Dim varData As Variant
varData = meas.GetData(naMeasResult, naDataFormat_Phase)
Print Data
For i = 0 to chan.NumberOfPoints-1
    Print varData(i)
Next i
```

[See a C# example.](#)

**C++ Syntax** HRESULT getData(tagNADataStore DataStore, tagDataFormat DataFormat, VARIANT \*pData)

**Interface** IMeasurement

---

Last Modified:

11-Jun-2009    Added EE notes

14-Apr-2009    Added link to C#

1-Oct-2007    Added temperature formats

**GetECALModuleInfoEx Method**

---

**Description** This property replaces [Get ECALModuleInfo Method](#).

Returns the following information about the connected ECAL module: model number, serial number, connector type, calibration date, min and max frequency.

The characterization within the ECal module that this information will be read from is specified by [ECALCharacterizationEx](#). The default value is 0.

**VB Syntax** *moduleInfo* = *cal*.**GetECALModuleInfoEx** (*module*)

**Variable** [\(Type\)](#) - Description

*moduleInfo* **(string)** - variable to store the module information

*cal* A Calibrator **(object)**

*module* **(long integer)** ECal module.

Choose from modules **1** through **8**

Use [IsECALModuleFoundEx](#) to determine the number of modules connected to the PNA

**Return Type** String

**Default** Not Applicable

**Examples** `info = cal.GetECALModuleInfoEx(2)`

**Example return string:**

```
ModelNumber: 85092-60007, SerialNumber: 01386, ConnectorType:
N5FN5F, PortAConnector: Type N (50) female, PortBConnector: Type
N (50) female, MinFreq: 30000, MaxFreq: 91000000000,
NumberOfPoints: 250, Calibrated: July 4 2002
```

**C++ Syntax** HRESULT GetECALModuleInfoEx(long moduleNumber, BSTR\* info);

**Interface** ICalibrator4

## GetEcalUserCharacterizer Method

---

**Description** This method returns a handle to an [ECalUserCharacterizer](#) object.

**VB Syntax** *calMgr*.GetEcalUserCharacterizer( )

**Variable** [\(Type\)](#) - Description

*calMgr* A [CalManager Object](#) (object)

**Return Type** IEcalUserCharacterizer

**Default** Not Applicable

**Example**

```
Dim mgr as ICalManager
Set mgr = app.GetCalManager
Dim ecalCharacterizer
Set ecalCharacterizer =
mgr.GetEcalUserCharacterizer()
```

**C++ Syntax** HRESULT GetEcalUserCharacterizer( IEcalUserCharacterizer  
\*\*obj);

**Interface** ICalManager6

---

Last Modified:

6-Mar-2009 MX New topic

## GetENRData Method

---

**Description** Read the ENR calibration data from PNA memory.

**VB Syntax** *vData* = *enr*.GetENRData( )

**Variable** [\(Type\)](#) - Description

*vData* Variable to store the returned ENR data. Frequency value in Hz, followed by corresponding ENR value in dB.

*enr* An [ENRFile](#) (object)

**Return Type** Variant Array

**Default** Not Applicable

**Examples** [See example program](#)

**C++ Syntax** HRESULT GetENRData (VARIANT vdata);

**Interface** IENRFile

---

Last Modified:

2-Aug-2007    MX New topic



## GetErrorCorrection Method

---

<b>Description</b>	<p>Reads the error correction state for the channel.</p> <p>Use <a href="#">ErrorCorrection Property</a> to set this value.</p> <p>When this command returns true, some measurements on the channel MAY not have error correction ON. This is because the Cal Set currently in place may not contain the appropriate calibration data. To read the error correction state for a measurement, use <a href="#">Error Correction Property</a>.</p>
<b>VB Syntax</b>	<i>chan</i> . <b>GetErrorCorrection</b> ( <i>boolean</i> )
<b>Variable</b>	<b>(Type) - Description</b>
<i>chan</i>	A <a href="#">Channel</a> ( <b>object</b> )
<i>boolean</i>	<b>(boolean)</b> Variable to store the returned value. <b>False</b> - Error correction has been set OFF <b>True</b> - Error correction has been set ON
<b>Return Type</b>	Boolean
<b>Default</b>	<a href="#">About Error Correction</a>
<b>Examples</b>	<code>chan.GetErrorCorrection(value)</code>
<b>C++ Syntax</b>	HRESULT GetErrorCorrection (VARIANT_BOOL *bState)
<b>Interface</b>	IChannel8

**GetErrorTerm Method - Superseded**

**Description** **Note:** This command is replaced by [Get ErrorTermByString\\_Method](#)

Retrieves error term data that is used for error correction. The data is complex pairs. Memory for the returned Variant is allocated by the server. The server returns a variant containing a two-dimensional safe Array.

This method returns a variant which is less efficient than [getErrorTermComplex](#) on the ICalData interface.

[Learn about reading and writing Calibration data.](#)

**VB Syntax** `data = cal.getErrorTerm term, rcv. src`

**Variable** [\(Type\)](#) - Description

*data* Variant array to store the data.

*cal* A Calibrator (**object**)

*term* (**enum As NaErrorTerm**). Choose from:

naErrorTerm\_Directivity\_Isolation

naErrorTerm\_Match

naErrorTerm\_Tracking

*rcv* (**long integer**) - Receiver Port

*src* (**long integer**) - Source Port

To get this	Specify these parameters:		
Error Term	<i>term</i>	<i>rcv</i>	<i>src</i>
Fwd Directivity	naET_Directivity Isolation	1	1
Rev Directivity	naET_Directivity Isolation	2	2
Fwd Isolation	naET_Directivity Isolation	2	1
Rev Isolation	naET_Directivity Isolation	1	2
Fwd Source Match	naErrorTerm_Match	1	1
Rev Source Match	naErrorTerm_Match	2	2
Fwd Load Match	naErrorTerm_Match	2	1
Rev Load Match	naErrorTerm_Match	1	2
Fwd Reflection Tracking	naErrorTerm_Tracking	1	1
Rev Reflection Tracking	naErrorTerm_Tracking	2	2

Fwd Trans Tracking	naErrorTerm_Tracking	2	1
Rev Trans Tracking	naErrorTerm_Tracking	1	2

---

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
Dim varError As Variant
varError = cal.getErrorTerm(naErrorTerm_Tracking, 2, 1)
```

**C++ Syntax** HRESULT getErrorTerm(tagNAErrorTerm ETerm, long ReceivePort, long SourcePort, VARIANT\* pData)

**Interface** ICalibrator

**GetErrorTerm Method** **Superseded**

<b>Description</b>	<p>This command has been replaced with <a href="#">Get ErrorTermByString</a></p> <p>Returns error term data from the Cal Set. The returned data is complex pairs.</p> <p>Learn more about <a href="#">Reading and Writing Cal Data</a></p> <p>See examples of <a href="#">Reading</a> and <a href="#">Writing</a> Cal Set Data</p>
<b>VB Syntax</b>	<i>data</i> = <i>calSet</i> . <b>getErrorTerm</b> ( <i>setNumber</i> , <i>term</i> , <i>rcv</i> , <i>src</i> )
<b>Variable</b>	<b>(Type) - Description</b>
<i>data</i>	<p><b>(Variant)</b> Two-dimensional safe array to store the returned data. Memory for the returned Variant is allocated by the PNA and must be released by client.</p> <p><b>Note:</b> See also <a href="#">getErrorTermComplex</a> on the ICalData2 interface to avoid using the variant data type.</p>
<i>calSet</i>	A <a href="#">Cal Set</a> ( <b>object</b> )
<i>setNumber</i>	<p><b>(Long)</b> There can be more than one set of error terms in a Cal Set.</p> <ul style="list-style-type: none"> <li>SetNumber <b>0</b> contains the original set of error terms for a Cal Set.</li> <li>SetNumbers &gt; <b>0</b> contain Interpolated error terms. Interpolated error terms are generated when interpolation is required and destroyed when no longer used. <a href="#">Learn about Interpolation</a>.</li> <li>To determine the SetNumber in use by a channel, see <a href="#">GetCalSetUsageInfo</a></li> </ul>
<i>term</i>	<p><b>(enum As NaErrorTerm2).</b> Choose from:</p> <p>0 - naET_Directivity (rcv = src)</p> <p>1 - naET_SourceMatch (rcv = src)</p> <p>2 - naET_ReflectionTracking (rcv = src)</p> <p>3 - naET_TransmissionTracking (rcv src)</p> <p>4 - naET_LoadMatch (rcv src)</p> <p>5 - naET_Isolation (rcv src)</p>
<i>rcv</i>	<b>(Long)</b> - Receiver Port
<i>src</i>	<b>(Long)</b> - Source Port
<b>Return Type</b>	Variant
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim varError As Variant varError = CalSet.getErrorTerm(0, naET_TransmissionTracking, 2, 1)</pre>

**C++ Syntax** HRESULT getErrorTerm(long setID, tagNAErrorTerm2 ETerm, long ReceivePort, long SourcePort, VARIANT\* pData)

**Interface** ICalSet

## GetErrorTermByString Method

---

**Description** Returns error term data from the Cal Set by specifying the string name of the error term.

- Learn more about [Reading and Writing Cal Data](#)
- See examples of [Reading](#) and [Writing](#) Cal Set Data
- See [GetCalSetUsageInfo](#) to determine the setNumber.

**VB Syntax** `pdata = calset.GetErrorTermByString(setNumber, errorTerm)`

**Variable** [\(Type\)](#) - **Description**

*pdata* **(Variant)** Two-dimensional safe array to store the returned data. Memory for the returned Variant is allocated by the PNA and must be released by client.

**Note:** See also [getErrorTermComplexByString](#) on the ICalData3 interface to avoid using the variant data type.

*calset* A [Cal Set](#) (**object**)

*setNumber* **(Long)** Set number of the required Cal Set data.

See [GetCalSetUsageInfo](#) to determine the setNumber.

- SetNumber **0** contains the original "master" set of error terms for a Cal Set.
- SetNumbers **> 0** refers to the PNA channel number that contains the error terms. When retrieving channel error terms, Correction must be ON.

New beginning PNA Rev 7.2: The channel error term data contains [interpolation](#), [fixturing](#), and [port extension](#) data if each is ON.

- For Balanced Measurements, interpolation, fixturing, and port extensions can be ON independently.
- For Standard S-parameters, to get port extension data, both fixturing and port extensions must be ON.

*errorTerm* **(String)** The string name used to identify a particular error term in the Cal Set. An example string for port 3 directivity in a full 2 port cal might be "Directivity(3,3)". To determine the string names of error terms, see [GetErrorTermList2](#).

**Return Type** Variant

**Default** Not Applicable

**Examples** [See an Example](#)

**C++ Syntax** HRESULT GetErrorTermByString (long SetNumber, BSTR bufferName, VARIANT\* pdata);

**Interface** ICalSet2

**GetErrorTermComplex Method** **Superseded**

<b>Description</b>	<p>This command has been replaced by <a href="#">Get ErrorTermComplexByString</a></p> <p>Retrieves error term data from the error correction buffer. The data is in complex pairs. Learn more about <a href="#">reading and writing Cal Data using COM</a>.</p> <p>This method exists on a non-default interface. If you cannot access this method, use the <a href="#">GetErrorTerm</a> Method on ICalibrator.</p>
<b>VB Syntax</b>	<i>eData</i> . <b>GetErrorTermComplex</b> <i>term</i> , <i>rcv</i> , <i>src</i> , <i>numPts</i> , <i>real()</i> , <i>imag()</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>eData</i>	An ICalData pointer to the Calibrator object
<i>term</i>	<b>(enum NAErrorTerm)</b> - The error term to be retrieved. Choose from: <ul style="list-style-type: none"> <li>• <b>naErrorTerm_Directivity_Isolation</b></li> <li>• <b>naErrorTerm_Match</b></li> <li>• <b>naErrorTerm_Tracking</b></li> </ul>
<i>rcv</i>	<b>(long integer)</b> - Receiver Port
<i>src</i>	<b>(long integer)</b> - Source Port
<i>numPts</i>	<b>(long integer)</b> - on input, max number of data points to return; on output: indicates the actual number of data points returned.
<i>real()</i>	<b>(single)</b> - array to accept the <b>real</b> part of the error-term. One-dimensional for the number of data points.
<i>imag()</i>	<b>(single)</b> - array to accept the <b>imaginary</b> part of the error-term. One-dimensional for the number of data points.



	To get this	Specify these parameters:		
	Error Term	<i>term</i>	<i>rcv</i>	<i>src</i>
	Fwd Directivity	naET_Directivity Isolation	1	1
	Rev Directivity	naET_Directivity Isolation	2	2
	Fwd Isolation	naET_Directivity Isolation	2	1
	Rev Isolation	naET_Directivity Isolation	1	2
	Fwd Source Match	naErrorTerm_Match	1	1
	Rev Source Match	naErrorTerm_Match	2	2
	Fwd Load Match	naErrorTerm_Match	2	1
	Rev Load Match	naErrorTerm_Match	1	2
	Fwd Reflection Tracking	naErrorTerm_Tracking	1	1
	Rev Reflection Tracking	naErrorTerm_Tracking	2	2
	Fwd Trans Tracking	naErrorTerm_Tracking	2	1
	Rev Trans Tracking	naErrorTerm_Tracking	1	2
<b>Return Type</b>	Single			
<b>Default</b>	Not Applicable			
<b>Examples</b>	<pre>ReDim rel(numpts) ReDim img(numpts) Dim eData As ICalData Set eData = chan.Calibrator eData.getErrorTermComplex naErrorTerm_Directivity_Isolation, 1, 1, 201, rel(0), img(0)</pre>			
<b>C++ Syntax</b>	HRESULT raw_getErrorTermComplex(tagNAErrorTerm ETerm, long ReceivePort, long SourcePort, long* pNumValues, float* pReal, float* plmag)			
<b>Interface</b>	ICalData			

**GetErrorTermComplex Method** **Superseded**

**Description** This command is [replaced](#) with [Get ErrorTermComplexByString](#)

Returns error term data from the Cal Set. The data is in complex pairs.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**Note:** This method exists on a non-default interface. If you cannot access this method, use the [GetErrorTerm](#) Method on ICal Set.

**VB Syntax** *iCalData2*.**GetErrorTermComplex** *setNumber, term, rcv, src, numPts, real(), imag()*

**Variable** [\(Type\)](#) - **Description**

*iCalData2* An [ICalData2](#) pointer to the Cal Set object

*setNumber* **(Long)** There can be more than one set of error terms in a Cal Set.

- setNumber **0** contains the original set of error terms for a Cal Set.
- setNumbers > **0** contain Interpolated error terms. Interpolated error terms are generated when interpolation is required and destroyed when no longer used. [Learn about Interpolation](#).
- To determine the setNumber in use by a channel, see [GetCalSetUsageInfo](#)

*term* **(enum NAEErrorTerm2)** - The error term to be retrieved. Choose from:

- 0 - naET\_Directivity
- 1 - naET\_SourceMatch
- 2 - naET\_ReflectionTracking
- 3 - naET\_TransmissionTracking
- 4 - naET\_LoadMatch
- 5 - naET\_Isolation

*rcv* **(Long)** - Receiver Port

*src* **(Long)** - Source Port

*numPts* **(Long)** An In/Out parameter.

On the way **in**, you specify the **max** number of values being requested.

On the way **out**, the PNA returns number of values actually returned.

*real()* **(single)** - array to accept the **real** part of the error-term. One-dimensional for the number of data points.

*imag()* **(single)** - array to accept the **imaginary** part of the error-term. One-dimensional for the number of data points.

**Return Type** Single

**Default** Not Applicable

**Examples**

```
dim numpts as long
numpts = ActiveChannel.NumberOfPoints
ReDim r(numpts) ' real part
ReDim i(numpts) ' imaginary part
Dim CalSet as CalSet
set CalSet = pna.GetCalManager.GetCal SetByGUID( txtGUID )
Dim eData As ICalData2
Set eData = CalSet
eData.getErrorTermComplex 0, naET_LoadMatch, 1, 2, numpts,
r(0),i (0)
```

**C++ Syntax** HRESULT getErrorTermComplex(long setID, tagNAErrorTerm2 ETerm, long ReceivePort, long SourcePort, long\* pNumValues, float\* pReal, float\* plmag)

**Interface** ICalData2

## GetErrorTermComplexByString Method

---

**Description** Returns error term data from the Cal Set by specifying the string name.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**Note:** This method exists on a non-default interface. If you cannot access this method, use [GetErrorTermByString](#)

**VB Syntax** *ICalData3*.**GetErrorTermComplexByString** *setNumber, errorTerm, numPoints, real(0), imag(0)*

**Variable** [\(Type\)](#) - **Description**

*ICalData3* An ICalData3 pointer to a [CalSet](#) (Object)

*setNumber* **(Long)** There can be more than one set of error terms in a Cal Set.

- *setNumber* **0** contains the original set of error terms for a Cal Set.
- *setNumbers* **> 0** contain Interpolated error terms. Interpolated error terms are generated when interpolation is required and destroyed when no longer used. [Learn about Interpolation](#).
- To determine the *setNumber* in use by a channel, see [GetCalSetUsageInfo](#)

*errorTerm* **(String)** The string name of error term in the Cal Set. An example string for port 3 directivity in a full 2 port cal might be "Directivity(3,3)".

For a list error term string names, use [Get ErrorTermList2](#)

*numPoints* **(Long)** An In/Out parameter.

On the way **in**, you specify the **max** number of values being requested.

On the way **out**, the PNA returns number of values actually returned.

*real* **(Single)** The real component of the complex data.

*imag* **(Single)** The imaginary component of the complex data.

**Return Type** Single

**Default** Not Applicable

**Examples** [See example](#)

**C++ Syntax** HRESULT GetErrorTermComplexByString(long etermSetID, BSTR bufferName, long\* lnumPoints, single\* real, single\* imag);

**Interface** ICalData3

**GetErrorTermList Method** **Superseded**

**Description** **Note:** This command is replaced by [CalSet.getErrorTermList2](#)

Returns the list of Error Terms contained in this Cal Set for the CalType specified in the [OpenCal Set](#) method. Learn more about [reading and writing Cal Data using COM](#).

The list is a comma separated, textual representation of the error terms with the term name followed by the port path in parentheses:

Term (n, n),  
Term (m,n)

Before calling this method you must open the Cal Set with [OpenCal Set](#). If the Cal set is not open, this method returns E\_NA\_Cal Set\_ACCESS\_DENIED.

Use [StringToNAErrorTerm2](#) to convert the list entrees to values that can be used with [GetErrorTerm](#) and [PutErrorTerm](#).

**Note:** The port path designation (m n) indicates the ports that contribute to the error being compensated. Directivity, source match and reflection tracking are single port characteristics, designated in this list by (n n) where n equals the port being characterized.

Other terms characterize the interaction between ports. For example, the load match term is describing the match at port (m) while looking into port (n). Thus the notation (m n) indicates the two ports that contribute to the loadmatch error.

**VB Syntax** *CalSet*.**GetErrorTermList** (SetID, count, strList)

**Variable** [\(Type\)](#) - **Description**

*CalSet* **(object)** - A Cal Set object

*SetID* (long) - specifies the error term set to query. Use 0 for the master set.

*count* (long) - the number of error terms in the returned list

*strList* **(string)** - comma separated list of error terms found in Cal Set

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
dim count as Integer
dim list as string
OpenCalSet (naCalType_TwoPortSOLT 1, 2)
GetErrorTermList( 0, count, list)
CloseCalSet( )
```

Assuming the cal set contained the full set of error terms for this two-port Cal, the returned list would be:

```
"Directivity(1 1),SourceMatch(1 1),ReflectionTracking(1
1),TransmissionTracking(2 1),LoadMatch(2 1),Isolation(2
```

```
1), Directivity(2 2), SourceMatch(2 2), ReflectionTracking(2  
2), TransmissionTracking(1 2), LoadMatch(1 2), Isolation(1 2) "
```

**C++ Syntax** HRESULT GetErrorTermList (long etermSetID, long\* count, BSTR\* strList);

**Interface** ICalSet

## GetErrorTermList2 Method

---

**Description** Returns a list of error terms names found in the Cal Set containing the specified prefix.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** *list = CalSet.GetErrorTermList2(setNumber, calTypePrefix)*

**Variable** [\(Type\)](#) - **Description**

*list* **(Variant)** Variant containing a string array of error term names.

*CalSet* **(object)** - A [CalSet](#) object

*setNumber* **(Long)** There can be more than one set of error terms in a Cal Set.

- setNumber **0** contains the original set of error terms for a Cal Set.
- setNumbers > **0** contain Interpolated error terms. Interpolated error terms are generated when interpolation is required and destroyed when no longer used. [Learn about Interpolation](#).
- To determine the setNumber in use by a channel, see [GetCalSetUsageInfo](#)

*caltypePrefix* **(String)** The string used to identify Cal Set data as belonging to a specific Cal Type. This string is used as a filter so that only the error term names of interest are returned. If the prefix is empty, all terms are returned.

An example prefix for a two port cal on ports 2 and 3 might be: "Full 2 Port Cal (2,3)".

**Return Type** Variant

**Default** Not Applicable

**Examples** [See an Example](#)

**C++ Syntax** HRESULT GetErrorTermList2 (long SetNumber, BSTR caltypePrefix, VARIANT\* list)

**Interface** ICalSet2



## GetExtendedCalInterface Method

---

**Description** Returns an interface that exposes the properties of Noise Calibration.

**VB Syntax** *Cal2*.**GetExtendedCalInterface** (*interface*)

**Variable** [\(Type\)](#) - **Description**

*Cal2* An ICalibrate2 (**object**)

*interface* (object) Returns a handle to the specified interface. Choose from '  
"NoiseCal"

### Return Type

**Default**

**Example**

```
dim noiseCal
dim noiseCalExtensions
set noiseCal= Get Calmanager?.CreateCustomCalEx("NoiseCal")
set noiseCalExtensions =
noiseCal.GetExtendedCalInterface("INoiseCal")
```

**C++ Syntax** HRESULT GetExtendedCalInterface();

**Interface** ICalibrate2

---

Last Modified:

29-May-2007    MN New topic

## Get ExternalTestSetIO Method

---

**Description** This method returns the [IExternalTestSetIO](#) interface.

**VB Syntax** `app.GetExternalTestSetIO`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** IHWExternalTestSetIO

**Default** Not Applicable

**Example**

```
Dim app As AgilentPNA835x.Application
Dim ets As IHWExternalTestSetIO
Set ets = app.GetExternalTestSetIO
```

**C++ Syntax** HRESULT GetExternalTestSetIO (IHWExternalTestSetIO  
\*\*ptestset);

**Interface** IApplication

## GetFilterStatistics Method

---

**Description** Returns all four Filter Statistics resulting from a [SearchFilterBandwidth](#). To retrieve individual filter statistics, use [meas.FilterCF](#), [meas.FilterBW](#), [meas.FilterLoss](#), [meas.FilterQ](#) properties.

**VB Syntax** *meas*.**GetFilterStatistics** *cf,bw,loss,q*

**Variable** [\(Type\)](#) - **Description**

*meas* A Measurement (**object**)

*cf,bw,loss,q* Dimensioned variables to store the returned values

**Return Type** **(double)** *cf*  
**(single)** *bw,loss,q*

**Default** Not Applicable

**Examples** **'Dimension variables**

```
Dim cf as Double  
Dim bw as Single  
Dim loss as Single  
Dim q as Single
```

```
meas.GetFilterStatistics cf,bw,loss,q
```

**C++ Syntax** HRESULT GetFilterStatistics(double\* centerFreq, float\* bw, float\* loss, float\* quality)

**Interface** IMeasurement

## GetGuid Method

---

**Description** Returns a string containing the GUID identifying this Cal Set. Each Cal Set is assigned a GUID (global unique ID). GUIDs are used to retrieve and select Cal Sets on the PNA. Learn more about [reading and writing Cal Data using COM](#).

**VB Syntax** *value* = *CalSet*.**GetGuid**

**Variable** [\(Type\)](#) - **Description**

*value* **(string)** - Variable to store the returned GUID

*CalSet* **(object)** - A Cal Set object

**Return Type** String

**Default** Not Applicable

**Examples** `guid = CalSet.GetGuid 'Read`

[C++ Syntax](#) HRESULT GetGUID( BSTR\* pGUIDString);

**Interface** ICalSet

## InputVoltageEX Property

---

<b>Description</b>	This command replaces <a href="#">get InputVoltage Method</a> Reads the ADC voltage from the specified location.
<b>VB Syntax</b>	<i>volts</i> = <i>AuxIO</i> .InputVoltageEX <i>loc</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>volts</i>	<b>(double)</b> - variable to store the return value
<i>AuxIO</i>	<b>(object)</b> - A <a href="#">Hardware Auxiliary Input / Output</a> object
<i>loc</i>	<b>(Long)</b> Location from which to read data. <b>For PNA-X models:</b> , reads ADC voltages from the <a href="#">Power I/O connector</a> . Choose from: <ol style="list-style-type: none"> <li>1 Reads voltage on Analog In 1 port (pin 7).</li> <li>2 Reads voltage on Analog In 2 port (pin 8).</li> <li>3 Reads voltage on GndSens (pin 6).</li> <li>4 Reads voltage on Analog Out 1 port (pin 3).</li> <li>5 Reads voltage on Analog Out 2 port (pin 4).</li> </ol> <b>For all other PNA models:</b> <ol style="list-style-type: none"> <li>1 Reads voltage on the Analog IN (pin 14) of the AUX IO connector.</li> <li>4 Reads voltage on Analog Out 1 port (pin 3).</li> <li>5 Reads voltage on Analog Out 2 port (pin 2).</li> </ol>
<b>Return Type</b>	Double
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim aux as HWAuxIO Set aux = PNA.getAuxIO volts = aux.InputVoltageEX 1 'for PNA-X, read voltage on PowerI/O pin 7 'for all other models, reads voltage on Aux I/O Analog In (pin 14)</pre>
<b>C++ Syntax</b>	HRESULT get_InputVoltageEX (long muxLoc, double* vtVoltage );
<b>Interface</b>	HWAuxIO2

---

Last Modified:

5-Aug-2008    Removed 'get'

10-Jul-2007    MX New topic

## get\_Input1 Method

---

<b>Description</b>	<p>Reads a hardware latch that captures high to low transitions on Input1 of the Material Handler IO. Reading the latch causes it to reset and is ready for the next transition. The hardware latch is only capable of capturing one transition per query. Additional transitions are ignored until after the next query.</p> <p>Momentarily driving Input1 high, then low, causes a transition to be detected and latched.</p>
<b>VB Syntax</b>	<i>inp1</i> = <i>handlerIo</i> .get_Input1
<b>Variable</b>	<b>(Type) - Description</b>
<i>inp1</i>	<b>(variant)</b> - A variable to store the return value
<i>handlerIo</i>	<b>(object)</b> - A HandlerIO object
<b>Return Type</b>	Variant - <b>0</b> - a high to low transition occurred at Input1 since the last time it was queried. <b>1</b> - no high to low transition occurred.
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>input1 = handlerIo.get_Input1 'Read</pre>
<b>C++ Syntax</b>	HRESULT get_Input1 (VARIANT* Data );
<b>Interface</b>	IHWMaterialHandlerIO

## GetIPConfigurationStruct Method

---

**Description** Returns an NA\_IPConfiguration data structure which contains information about the current status of the PNA's computer networking configuration. This is the same set of information that is returned in a single string by the [LANConfiguration](#) property.

**VB Syntax** *value* = *app*.GetIPConfigurationStruct

**Variable** [\(Type\)](#) - [Description](#)

*value* **(NA\_IPConfiguration)** Variable to receive the PNA IP (LAN) configuration information.

*app* An [Application](#) **(object)**

**Return Type** NA\_IPConfiguration

**Default** Not Applicable

**Examples**

```
Dim networkConfigInfo As NA_IPConfiguration
networkConfigInfo = app.GetIPConfigurationStruct()

MsgBox "Host name = " & networkConfigInfo.HostName

MsgBox "Domain name = " & networkConfigInfo.DomainName

MsgBox "IP address = " & networkConfigInfo.IPAddress

If Not networkConfigInfo.DHCPEnabled Then
    MsgBox "IP address is static"
Else
    MsgBox "IP address is dynamic"
End If

MsgBox "Subnet mask = " & networkConfigInfo.SubNet

MsgBox "Gateway = " & networkConfigInfo.DefaultGateway

MsgBox "Primary DNS server = " & networkConfigInfo.DNSServer1

MsgBox "Secondary DNS server = " & networkConfigInfo.DNSServer2

MsgBox "First suffix in DNS suffix search order = " &
networkConfigInfo.DNSSuffix1

MsgBox "Second suffix in DNS suffix search order = " &
networkConfigInfo.DNSSuffix2

MsgBox "Primary WINS server = " &
networkConfigInfo.PrimaryWINSServer
```



```
MsgBox "Secondary WINS server = " &
networkConfigInfo.SecondaryWINSServer

MsgBox "Network adapter device ID = " &
networkConfigInfo.DeviceID

MsgBox "Description of network adapter = " &
networkConfigInfo.Description

MsgBox "MAC address = " & networkConfigInfo.MacAddress
```

**C++ Syntax** HRESULT GetIPConfigurationStruct (tagNA\_IPConfiguration \* pIPConfig);

**Interface** IApplication14

---

Last Modified:

2-Jun-2008    MX New topic

**GetIsolationPaths Method**

---

**Description** Gets the list of paths (port pairings) for which isolation standards will be measured during calibration.

**VB Syntax** *value* = *obj*.**GetIsolationPaths**

**Variable** [\(Type\)](#) - **Description**

*value* (Variant) - Variable to store the returned port paths in pairs. One-dimensional array of Long Integers.

*obj* Any of the following:  
[GuidedCalibration](#) (object)

**Return Type** Variant – Containing one-dimensional array of Long Integers.

**Default** No port pairs (empty Variant variable)

**Examples**

```
pathList = guidedCal.GetIsolationPaths
'displaying the paths separated by commas,
'with a dash (-) between the pair of port numbers comprising
each path

For i = LBound(portList) To UBound(portList) Step 2
    msg = msg + CStr(portList(i)) + "-" + CStr(portList(i+1))
    If i+1 < UBound(portList) Then msg = msg + ", "
Next
MsgBox msg, 0, "List of isolation paths"
```

**C++ Syntax** HRESULT GetIsolationPaths(VARIANT\* pathList);

**Interface** IGuidedCalibration

---

Last Modified:

16-Apr-2007    MX New topic

## Get MaterialHandlerIO Method

---

**Description** This method returns the [MaterialHandlerIO](#) interface.

**VB Syntax** *app*.GetMaterialHandlerIO

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** IHWMaterialHandlerIO

**Default** Not Applicable

**Example**

```
Dim app As AgilentPNA835x.Application
Dim hand As IHWMaterialHandlerIO
Set hand = app.GetMaterialHandlerIO
```

**C++ Syntax** HRESULT GetMaterialHandlerIO (IHWMaterialHandlerIO  
\*\*phand);

**Interface** IApplication

## GetNAComplex Method

---

**Description** Retrieves complex data from the specified location.  
See also [getComplex](#) and [getData](#) Method.

**VB Syntax** *measData.getNAComplex location, numPts, data*

**Variable** [\(Type\) - Description](#)

*measData* An IArrayTransfer interface which supports the Measurement object

*location* **(enum NADataStore)** - Where the data you want is residing. Choose from:  
 0 - naRawData  
 1 - naCorrectedData  
 2 - naMeasResult  
 3 - naRawMemory  
 4 - naMemoryResult  
 5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

*numPts* **(long integer)** - Number of data points requested  
 [out] - specifies number of data elements returned  
 [in] - specifies the data being requested or the capacity of the *dComplex* array

*data* **(NAComplex)** - A one-dimensional array of NaComplex to store the data.

**Return Type** NAComplex

**Default** Not Applicable

**Examples**

```
Dim dComplex(201) AS NaComplex
Dim measData As IArrayTransfer
Dim pts as Long
Set measData = app.ActiveMeasurement
measData.getNAComplex naCorrectedData, pts, dComplex(0)
```

**Notes** The data is stored as Real and Imaginary (**Re** and **Im**) members of the NaComplex user defined type. You can access each number individually by iterating through the array.

```
For i = 0 to NumPts-1
    dReal (i) = dcomplex (i).Re
    dImag (i) = dcomplex (i).Im
Next i
```

**C++ Syntax** HRESULT getNAComplex(tagNADataStore DataStore, long\* pNumValues, TsComplex\* pComplex)

**Interface** IArrayTransfer

## GetNumberOfGroups Method

---

**Description** Returns the number of groups a channel has yet to acquire. To set the number of groups for a channel, use [Number Of Groups Method](#)

**VB Syntax** *value* = *chan*.**GetNumberOfGroups**

**Variable** [\(Type\)](#) - Description

*value* **(Long Integer)** - Number of groups

*chan* Channel **(object)**

**Return Type** **(Long Integer)**

**Default** Not Applicable

**Examples** `groups = chan.GetNumberOfGroups 'Read`

**C++ Syntax** HRESULT GetNumberOfGroups(long\* numberOfGroups);

**Interface** IChannel3

## get\_Output Method

---

**Description** [Type 1 and Type2 configurations](#): Returns the last value written to the selected output pin.

[Type3 configuration](#): Returns the current state of the selected output pin. If an Input1 trigger occurs, the state may not be the same value as was written.

All configurations: Data is written using [put\\_Output](#) Method.

**VB Syntax** `data = handlerIo.get_Output (pin)`

**Variable** [\(Type\) - Description](#)

*data* **(variant)** - A variable to store the return value. The returned value will be one of the following:

**0** - TTL Low

**1** - TTL High

*handlerIo* **(object)** - A HandlerIO object

*pin* **(enum as NAMatHandlerOutput)** - output to read. Choose from:

**naOutput1 (0)**

**naOutput1User (1)**

**naOutput2 (2)**

**naOutput2User (3)**

[Learn about User Output](#)

**Return Type** Variant

**Default** Not Applicable

**Examples** `data = handlerIo.get_Output (naOutput1)`

**C++ Syntax** `HRESULT get_Output ( tagNAMatHandlerOutput Output, VARIANT* Data );`

**Interface** IHWMaterialHandlerIO

## get\_OutputVoltage Method

---

**Description** **E836x and PNA-L:** Reads voltages on the DAC/Analog Output 1|2 of the Auxiliary IO connector.

**PNA-X:** Reads voltage on the [Power I/O connector](#) AnalogOut1|2.

**VB Syntax** *volts = AuxIO.get\_OutputVoltage (output)*

**Variable** [\(Type\)](#) - **Description**

*volts* **(double)** - variable to store the return value

*AuxIO* **(object)** - A Hardware Auxiliary Input / Output object

*output* **(variant)** Number of the output DAC from which to read voltage. Choose from:

**1** Output 1 (Aux I/O pin 3) and (Power I/O pin 3)

**2** Output 2 (Aux I/O pin 2) and (Power I/O pin 4)

**Return Type** Double

**Default** Not Applicable

**Examples**

```
Dim aux as HWAuxIO
Set aux = PNA.getAuxIO
volts = aux.get_OutputVoltage(1)
'read voltage from Analog Out 1 (Aux I/O pin3) or (Power I/O pin 3)
```

**C++ Syntax** HRESULT get\_OutputVoltage( VARIANT Output, double\* Voltage );

**Interface** IHWAuxIO

---

Last Modified:

10-Jul-2007    Added PNA-X capability



**get OutputVoltageMode Method**

---

**Description** This command returns the mode of the selected "Analog Out" line on the [Auxiliary IO connector](#) and [Power I/O connector](#). The modes give the user the option to have the requested voltage applied immediately or not until the sweep is done. To set the mode, use [put OutputVoltageMode Method](#).

**VB Syntax** *mode* = *auxIo*.get\_OutputVoltageMode (*output*)

**Variable** [\(Type\)](#) - Description

*mode* **(enum NAOOutputVoltageMode )** - variable to store the returned mode.

**naWaitEOS** - While in this mode any voltage changes sent to the selected analog out will only get applied to the output between sweeps.

**naNoWait** - While in this mode any voltage changes sent to the selected analog out will occur right away without waiting until the end of a sweep, the voltage gets applied immediately.

*auxIo* **(object)** - A Hardware Auxiliary Input / Output object

*output* (double) Analog Output. Choose from **1** or **2**

**Return Type** enum as NAOOutputVoltageMode

**Default** naWaitEOS

**Examples** `vOutMode = auxIo.get_OutputVoltageMode (1)`

**C++ Syntax** HRESULT get\_OutputVoltageMode(VARIANT Output, tagNAOutputVoltageMode\* pMode);

**Interface** IHWAuxIO

---

Last Modified:

10-Jul-2007    Added PNA-X capability

## GetPairedData Method

---

**Description** Retrieves pairs of data from the specified location.

**Note:** This method exists on a non-default interface. If you cannot access this method, use the [Get Data](#) Method on IMeasurement.

**VB Syntax** *measData.getPairedData location, format, numPts, d1, d2*

**Variable** [\(Type\)](#) - Description

*measData* An IArrayTransfer interface which supports the Measurement object

*location* **(enum NADataStore)** - Where the data you want is residing. Choose from:  
0 - naRawData  
1 - naCorrectedData  
2 - naMeasResult  
3 - naRawMemory  
4 - naMemoryResult  
5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).  
See [Data Access Map](#)

*format* **(enum NAPairedDataFormat)** - Format in which you would like the Paired data. Choose from:

**0 -naLogMagPhase** - Log magnitude and phase

**1 -naLinMagPhase** - Linear magnitude and phase

**2 -naReallImaginary** - Real and Imaginary

**Note:** Selecting **naReallImaginary** format is the same as using the [getComplex](#) method

*numPts* **(long integer)** - Number of data points requested  
[out] - specifies number of data elements returned  
[in] - specifies the data being requested or the capacity of the *dPaired* array

*d1* **(single)** - Array to store the magnitude / real values

*d2* **(single)** - Array to store the phase / imaginary values

**Return Type** Two Single arrays

**Default** Not Applicable

**Examples**

```
Dim logm() As Single
Dim phase() As Single
Public measData As IArrayTransfer
```

```
Set measData = app.ActiveMeasurement
Dim numpts As Long
numPoints = app.ActiveChannel.NumberOfPoints
ReDim logm(numPoints)
ReDim phase(numPoints)

measData.getPairedData naCorrectedData, naLogMagPhase,
numPoints, logm(0), phase(0)

Print values(0), values(1)
```

**C++ Syntax** HRESULT getPairedData(tagNADataStore DataStore, tagNAPairedDataFormat  
PairFormat, long\* pNumValues, float\* pReal, float\* plmag)

**Interface** IArrayTransfer

## get\_Port Method

**Description** Returns the value from the specified "readable" port.

**VB Syntax** `data = handlerIo.get_Port (port)`

**Variable** [\(Type\) - Description](#)

*data* **(variant)** - A variable to store the return value. The following table shows what the returned data represents:

Port	MSB.....LSB 8.....0
C	C3...C0
D	D3...D0
E	D3...D0 + C3...C0

*handlerIo* **(object)** - A HandlerIO object

*port* **(enum as NAMatHandlerPort)** - port to get data from. Choose from:

**naPortC - (2)**

**naPortD - (3)**

**naPortE - (4)**

**Note:** Reading data from the Write-only ports (A,B,F,G,H) will return an error. Ports C and D must be put in Read mode before reading from C, D, or E using [PortMode Property](#).

**Return Type** Variant

**Default** 0

**Examples** `data = handlerIo.get_Port (naPortC)`

**C++ Syntax** `HRESULT get_Port ( tagNAMatHandlerPort Port, VARIANT* Data );`

**Interface** IHWMaterialHandlerIO

## get\_PortCData Method

---

**Description** Reads a 4-bit value from Port C of the Aux I/O connector (pins 22-25) and the Material Handler IO (pins 21-24 Anritsu) - (pins 22-25 Avantest).

**Note:** These lines are connected to both the Handler IO and Aux IO in the PNA.

**VB Syntax** `value = AuxIO.get_PortCData`

**Variable** [\(Type\)](#) - Description

`value` **(variant)** - Variable to store the returned data

`AuxIO` **(object)** - A Hardware Auxiliary Input / Output object

**Return Type** Integer

**Default** None

**Examples** `value = auxIo.get_PortCData 'Reading a value of 15 when in Positive Logic indicates Port C lines C0, C1, C2, C3 are High. If in Negative Logic they are Low.'`

**C++ Syntax** `HRESULT get_PortCData( VARIANT* Data );`

**Interface** IHWAuxIO

## GetPortNumber Method

---

**Description** Returns the port number that is associated with the specified port name. These numbers are used with several commands to specify a PNA port.

To learn more, see [Remotely Specifying a Source Port](#).

**VB Syntax** *value* = *object*.**GetPortNumber** (*portName*)

**Variable** [\(Type\)](#) - **Description**

*value* **(Long)** - Variable to store the returned Port Number integer value.

*object* A [Channel](#) **(object)** - always more complete than capabilities object.

A [Capabilities](#) **(object)** - use when a channel is not available

*portName* **(String)** Name of the PNA port.

- Use [SourcePortNames Property](#) to return a list of PNA port (string) names.
- If an external source is selected, specify the external source name that is used in the [Select an External Source dialog](#).

**Return Type** Long Integer

**Default** Not Applicable

**Examples** `value = chan.GetPortNumber ("Src2 Out1") 'Read`

**C++ Syntax** HRESULT GetPortNumber(BSTR name, long \*number);

**Interface** IChannel13  
ICapabilities4

---

### Last Modified:

12-Oct-2009	Added parens
23-May-2008	Added channel object
24-Apr-2008	Fixed example
30-Apr-2007	Edited for Ext Source Control

## GetRaw2DData Method

---

**Description** Returns raw data at all frequency and power data points for **any** GCA sweep. Previously 2D sweep only.

If correction is on, corrected data are returned. Otherwise, raw data are returned.

**VB Syntax** `data = gca.GetRaw2DData location, format, param`

**Variable** [\(Type\)](#) - **Description**

*data* Variant array in which to store returned measurement data.

*gca* A [GainCompression](#) (**object**)

*location* (enum NADataStore) - Where the data you want is residing. Choose from:  
 0 - naRawData  
 1 - naCorrectedData

*format* (enum NADataFormat) - Format in which you would like the data. It does not have to be the displayed format. Choose from:  
 0 - naDataFormat\_LinMag  
 1 - naDataFormat\_LogMag  
 2 - naDataFormat\_Phase  
 3 - naDataFormat\_Polar\*  
 4 - naDataFormat\_Smith\*  
 5 - naDataFormat\_Delay -- **Not valid for this command.**  
 6 - naDataFormat\_Real  
 7 - naDataFormat\_Imaginary  
 8 - naDataFormat\_SWR  
 9 - naDataFormat\_PhaseUnwrapped  
 10 - naDataFormat\_InverseSmith

[Learn more about Data Format.](#)

\* Specify Smith or Polar formats to obtain complex data pairs, which require a two-dimensional array varData (numpts, 2) to accommodate both real and imaginary data.

All scalar formats return a single dimension varData(numpts).

naDataFormat\_Phase and naDataFormat\_PhaseUnwrapped returns degrees.

*param* (**String**) Parameter of data to return. Not case-sensitive. The specified parameter need NOT be displayed.

Choose from:

- **"pin"** - input power at each data point.
- **"pout"** - output power at each data point.
- **"gain"** - device gain (S21) at each data point.
- **"inputmatch"** - input match (S11) at each data point.
- **"DeltaGain"** - Measured Gain (watts) / Ref Gain (watts). [Learn more.](#)
- **"AI1"** and **"AI2"** - ADC measurements at the specified compression level. [Learn more.](#)

**Return Type** Variant Array

**Default** Not Applicable

**Examples** `data = gca.GetRaw2DDData naRawData, naDataFormat_Real, "pin"`

**[C++ Syntax](#)** HRESULT GetRaw2DDData(tagNADataStore location, tagNADataFormat format, BSTR data\_name, VARIANT\* pData);

**Interface** IGainCompression

---

Last Modified:

31-Mar-2009	Added new params
12-May-2008	Edited for any GCA sweep
22-Oct-2007	MX New topic



## GetDataIm Method

---

**Description** Reads the Imaginary part of the data acquired from **any** GCA sweep. Previously 2D sweep only.

**VB Syntax** `data = gca.GetDataIm stim, dPoint, param`

**Variable** [\(Type\)](#) - Description

*data* Variant array in which to store returned measurement data.

*gca* A [GainCompression](#) (object)

*stim* (**NAGCAIndexSelect**)

- **naFrequencySelect** - for the specified frequency data point, returns all of the measured data for each power stimulus.
- **naPowerSelect** - for the specified power data point, returns all of the measured data for each frequency stimulus.

*dPoint* Data point (Frequency or Power) for which data is returned.

*param* Parameter of data to return. Not case-sensitive. Choose from:

- **"pin"** - input power at each data point.
- **"pout"** - output power at each data point.
- **"gain"** - device gain (S21) at each data point.
- **"inputmatch"** - input match (S11) at each data point.
- **"DeltaGain"** - Measured Gain (watts) / Ref Gain (watts). [Learn more.](#)
- **"AI1"** and **"AI2"** - ADC measurements at the specified compression level. [Learn more.](#)

**Return Type** Variant Array

**Default** Not Applicable

**Examples** For the fifth frequency data point, returns 'Power Output' imaginary (phase) data from all power stimulus values. If there are 30 power sweep points, 30 values are returned.

```
data = gca.GetDataIm naFrequencySelect, 5, "pout"
```

For the 30th stimulus power data point, returns 'Power Output' imaginary (phase) data from all frequency stimulus values. If there are 201 frequency sweep points, 201 values

are returned.

```
data = gca.GetDataIm naPowerSelect, 30, "pout"
```

**C++ Syntax** HRESULT GetDataIm(tagNAGCAIndexSelect index\_select, int index, BSTR data\_name, VARIANT\* pData);

**Interface** IGainCompression

---

Last Modified:

31-Mar-2009	Added new params
12-May-2008	Edited for any GCA sweep
22-Oct-2007	MX New topic

## GetDataRe Method

---

**Description** Reads the REAL part of the data acquired from **any** GCA sweep. Previously 2D sweep only.

**VB Syntax** `data = gca.GetDataRe stim, dPoint, param`

**Variable** [\(Type\)](#) - Description

*data* Variant array in which to store returned measurement data.

*gca* A [GainCompression](#) (object)

*stim* (**NAGCAIndexSelect**)

- **naFrequencySelect** - for the specified frequency data point, returns all of the measured data for each power stimulus.
- **naPowerSelect** - for the specified power data point, returns all of the measured data for each frequency stimulus.

*dPoint* Data point (Frequency or Power) for which data is returned.

*param* Parameter of data to return. Not case-sensitive. Choose from:

- **"pin"** - input power at each data point.
- **"pout"** - output power at each data point.
- **"gain"** - device gain (S21) at each data point.
- **"inputmatch"** - input match (S11) at each data point.
- **"DeltaGain"** - Measured Gain (watts) / Ref Gain (watts). [Learn more.](#)
- **"AI1"** and **"AI2"** - ADC measurements at the specified compression level. [Learn more.](#)

**Return Type** Variant Array

**Default** Not Applicable

**Examples** For the fifth frequency data point, returns 'Power Output' REAL data from all power stimulus values. If there are 30 power sweep points, 30 values are returned.

```
data = gca.GetDataRe naFrequencySelect, 5, "pout"
```

For the 30th stimulus power data point, returns 'Power Output' REAL data from all frequency stimulus values. If there are 201 frequency sweep points, 201 values are

returned.

```
data = gca.GetDataRe naPowerSelect, 30, "pout"
```

**C++ Syntax** HRESULT GetDataRe(tagNAGCAIndexSelect index\_select, int index,BSTR data\_name, VARIANT\* pData);

**Interface** IGainCompression

---

Last Modified:

31-Mar-2009	Added new params
12-May-2008	Edited for any GCA sweep
22-Oct-2007	MX New topic

## GetRxLevelingConfiguration Method

---

**Description** This method returns a handle to a [RxLevelingConfiguration](#) object.

**VB Syntax** *chan*.GetRxLevelingConfiguration()

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

**Return Type** IRxLevelingConfiguration

**Default** Not Applicable

**Example**

```
dim app as AgilentPNA835x.Application
dim mgr as RxLevelingConfiguration
set mgr = app.GetRxLevelingConfiguration()
```

**C++ Syntax** HRESULT GetRxLevelingConfiguration( IRxLevelingConfiguration  
\*\*mgr);

**Interface** IChannel17

---

Last Modified:

3-Mar-2009    MX New topic

## GetSourceByRole Method

---

**Description** Returns the name of a source that is assigned to the specified role.

**VB Syntax** *source* = *conv*.**GetSourceByRole** (*role*)

**Variable** [\(Type\)](#) - **Description**

*source* (String) Source name, from [Source Configuration dialog](#), that is assigned to the specified role.

*conv* A [Converter Object](#)

*role* (String) Role for which the source name will be returned. Use [GetSourceRoles](#) for a list of valid roles.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `"LO" = conv.GetSourceByRole ("RF2")`

**C++ Syntax** HRESULT GetSourceByRole(BSTR roleID, BSTR deviceName);

**Interface** IConverter

---

Last Modified:

2-Feb-2009    New topic

[Read only](#)

## GetSourceRoles Method

---

**Description** Returns the defined role names ("RF2", "LO1").

**VB Syntax** *conv*.**GetSourceRoles** (*roles*)

**Variable** [\(Type\)](#) - Description

*conv* A [Converter Object](#)

*roles* (Variant array) Variable to store returned list of valid roles.

**Return Type** Variant

**Default** Not Applicable

**Examples** `conv.GetSourceRoles (roles)`

**C++ Syntax** HRESULT GetSourceRoles(VARIANT\* roles);

**Interface** IConverter

---

Last Modified:

3-Feb-2009    New topic

## GetReferenceMarker Method

---

**Description** Returns a handle to the reference marker.

**VB Syntax** *meas*.**GetReferenceMarker**

**Variable** [\(Type\)](#) - **Description**

*meas* A Measurement (**object**)

**Return Type** Object

**Default** Not Applicable

**Examples** `meas.GetReferenceMarker`

**[C++ Syntax](#)** HRESULT GetReferenceMarker(IMarker\*\*  
refMarker)

**Interface** IMeasurement



**GetRequiredEtermNames**

---

**Description** Returns an array of strings specifying the error terms required by the caltype's correction algorithm in order to correct the specified parameter.

This function interrogates a specific caltype ( caltypeGUID) for the list of error terms it would need in order to correct the specified parameter. All the standard S Parameter calibration types embed port specifiers in the error term name. The specific port information is gleaned from the passed parameter. For example, to query the error term requirements specific to a two port cal on ports 1 and 3, issue this with a parameter of S13 or S31. The buffer names returned will be formatted in this way:

Full 1 Port SOLT(1,3):TransmissionTracking(3,1)

**VB Syntax** **EtermNames = GetRequiredEtermNames**(CalTypeGUID As String, Parameter As String)

**Variable** [\(Type\)](#) - **Description**

*caltypeGUID*: [in] the GUID of the desired calibration type

*parameter* [in] string specifying the parameter to be corrected

EtermNames [out] array of strings containing the error term names.

**Note:** In C++ Allocated by server. Must be freed by caller using SysFreeString.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** enames = GetRequiredEtermNames(ctGUID, Parm)

**C++ Syntax** HRESULT GetRequiredEtermNames( BSTR caltypeGUID, BSTR parameter, VARIANT\* EtermNames )

**Interface** ICalManager2

## GetScalar Method

---

**Description** Retrieves scalar data (ONE number per data point) from the specified location.

**Note:** This method exists on a non-default interface. If you cannot access this method, use the [Get Data](#) Method on IMeasurement.

**VB Syntax** *measData.getScalar location, format, numPts, data*

**Variable** [\(Type\)](#) - **Description**

*measData* An IArrayTransfer interface which supports the Measurement object

*location* **(enum NADataStore)** - Where the data you want is residing. Choose from:

- 0 - naRawData
- 1 - naCorrectedData
- 2 - naMeasResult
- 3 - naRawMemory
- 4 - naMemoryResult
- 5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

*format* **(enum NADataFormat)** - Format in which you would like the data. Choose from:

- 0 - naDataFormat\_LinMag
- 1 - naDataFormat\_LogMag
- 2 - naDataFormat\_Phase
- 3 - naDataFormat\_Polar
- 4 - naDataFormat\_Smith
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

**Note:** Polar, Smith, and Inverse Smith are invalid formats for this command. See [Get Complex Method](#).

[Learn more about Data Format.](#)

*numPts* **(long integer)** - Number of data points requested  
[out] - specifies number of data elements returned  
[in] - specifies the data being requested or the capacity of the *dScalar* array

*data* **(single)** - Array to store the scalar data.

**Return Type** Single

**Default** Not Applicable

**Examples**

```
Dim dScalar() As Single
Dim measData As IArrayTransfer
Set measData = app.ActiveMeasurement
Dim numpts as Long
numpts = app.ActiveChannel.NumberOfPoints
ReDim dScalar(numPoints)

measData.getScalar naCorrectedData, naDataFormat_LogMag, numpts,
dScalar(0)
Print dScalar(0), dScalar(1)
```

**C++ Syntax** HRESULT getScalar(tagNADataStore DataStore, tagNADataFormat DataFormat, long\* pNumValues, float\* pVals)

**Interface** IArrayTransfer

---

Last Modified:

1-Oct-2007    Added temperature formats

## GetShortcut Method

---

**Description** Returns the Title, Path, and optional argument strings, of the specified Macro (shortcut). Use this method to list the titles and paths of macros in the analyzer.

**VB Syntax** *app.GetShortcut index, title, path, arguments*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

*index* **(long)** - Number of the macro. Use a number between **1** and **12**.

*title* **(string)** - **Title** of the specified macro. (Appears in the softkey label)

*path* **(string)** - **Pathname** of the specified macro.

*arguments* **(string)** - Arguments for the specified macro

**Return Type** String

**Default** Not Applicable

**Example**

```
Dim t As String
Dim p As String
Dim arg As String
Dim i As Integer
For i = 1 to 12
    app.GetShortcut i,t,p,arg
    Print t,p
Next
```

**C++ Syntax** HRESULT GetShortcut(long Number, BSTR\* title, BSTR\* pathname, BSTR\* arguments)

**Interface** IApplication

**Remarks** Shortcuts can also be defined and accessed using the macro key on the front panel. However, the benefit of this feature is primarily for the interactive user

**GetSnPData Method** **Superseded**

<b>Description</b>	<p><b>Note:</b> this command is replaced by <a href="#">Get SnpDataWithSpecifiedPorts Method</a>.</p> <p>Reads SnP data from the selected measurement. <a href="#">Learn more about SnP that is returned from the PNA.</a></p>
<b>VB Syntax</b>	<i>data</i> = <i>meas</i> . <b>GetSnPData</b> <i>type</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>data</i>	Variant array to store the data.
<i>meas</i>	A Measurement ( <b>object</b> )
<i>type</i>	<p><b>(string)</b> - Type of SnP data to return. If unspecified, &lt;n&gt; is set to 2. Choose from:</p> <p><b>"S1P"</b> returns data for the active measurement.</p> <p><b>"S2P"</b> returns data for the current 2-port measurement (4 S-parameters).</p> <p><b>"S3P"</b> returns data for the current 3 port measurement (9 S-parameters). Valid only on instruments with 3 ports or more.</p> <p><b>"S4P"</b> returns data for the current 4 port measurement (16 S-parameters). Valid only on instruments with 4 ports or more.</p> <p>SnP data can be output using several data formatting options. See <a href="#">SnPFormat Property</a></p>
<b>Return Type</b>	<p>Variant - 3 dimensional array.</p> <ul style="list-style-type: none"> <li>• First dimension size is number of parameters returned.</li> <li>• Second dimension size is number of points in the channel</li> <li>• Third dimension size is 2 (real,imaginary)</li> </ul> <p>For example:</p> <p>Data(0,5,1) returns the imaginary value of the fifth data point of S11 (if the s2p request includes port #1)</p>
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>snp = meas.GetSnPData("s1p")</pre>
<b>C++ Syntax</b>	HRESULT GetSnPData( BSTR snptype, VARIANT * response)
<b>Interface</b>	IMeasurement3

Last Modified:

14-Nov-2008 Added returned data detail

## GetSnPDataWithSpecifiedPorts Method

---

**Description** **Note:** This command replaces [Get SnPData](#). This command is more explicit regarding the data to be returned, and works for PNAs with multiport test sets.

Reads SnP data for the measurement by specifying the PNA port numbers. [Learn more about SnP that is returned from the PNA.](#)

**VB Syntax** *data = meas.GetSnPDataWithSpecifiedPorts ports*

**Variable** [\(Type\)](#) - Description

*data* **(Variant)** array to store the data.

*meas* A [Measurement](#) **(object)**

*ports* **(Variant Array)** One-dimensional array containing the list of port numbers for which data is required.

**Return Type** Variant - 3 dimensional array.

- First dimension size is number of parameters returned.
- Second dimension size is number of points in the channel
- Third dimension size is 2 (real,imaginary)

For example:

Data(0,5,1) returns the imaginary value of the fifth data point of S11 (if the s2p request includes port #1)

**Default** Not Applicable

**Example** 'This VBScript example can be pasted into a notepad file and run on the PNA as a macro. [Learn how.](#)

```
Dim pna
Dim meas
Dim param
Dim point
Dim snp
Dim ports

'List the port numbers for required data
ports = Array(3,4)
Set pna = CreateObject("AgilentPnA835x.application")
Set meas = pna.ActiveMeasurement
```

```

'limit amount of data to display
set chan=pna.ActiveChannel
chan.NumberOfPoints=2
snp = meas.GetSnPDataWithSpecifiedPorts (ports)
' returns a 3 dimensional array
' snp(param,point,data pair)
'-----
' show me the data
For param = LBound(snp, 1) To UBound(snp, 1)
    MsgBox ("Parameter: " & (param + 1))
    For point = LBound(snp, 2) To UBound(snp, 2)
        MsgBox "Point:" & (point + 1) & " " & snp(param, point, 0) &
        "," & snp(param, point, 1)
    Next
Next

```

**C++ Syntax** HRESULT GetSnPDataWithSpecifiedPorts(VARIANT portsToMeasure,VARIANT\* response);

**Interface** IMeasurement7

Last modified:

13-Nov-2008 Added detail to return type

9/18/06 MQ Added for multiport



## getSourcePowerCalDataEx Method

**Description** **Note:** This method replaces [getSourcePowerCalData Method](#)

Retrieves (as variant data type) source power calibration data, if it exists, from the channel.

If the channel is sweeping the source backwards, then the first data point is the highest frequency value; the last data point is the lowest. Use the [Get X-Axis Values](#) command to return the X-axis values in the displayed order.

**Note:** This method returns a variant which is less efficient than methods available on the ISourcePowerCalData interface

**VB Syntax** `data = chan.getSourcePowerCalDataEx (buffer, srcPort)`

**Variable** [\(Type\)](#) - Description

*data* **(variant)** – Array to store the data.

*chan* **(object)** – A [Channel](#) object

*buffer* **(enum NASourcePowerCalBuffer)** - The requested source power cal data buffer.

**0 - naCorrectionValues** Last iteration of Cal data

**1 - naPriorIterationCorrectionValues** Prior iteration of Cal data. This argument can be used to determine the final power reading at each point of the power cal, for a cal that did not pass tolerance limits.

The following formula can be used to determine the power reading (in dB):

Power reading = Target power at the source port + specified power cal offset value + 'prior' iteration corr value – actual power corr value.

The "actual" value in this equation is returned with **naCorrectionValues**.

*srcPort* **(long integer)** – The source port for which calibration data is being requested.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

**Return Type** Variant array – automatically dimensioned to the size of the data.

**Default** Not Applicable

**Examples**

```
Dim varData As Variant
Const port1 As Long = 1
varData = chan.getSourcePowerCalDataEx (naCorrectionValues,
port1)
'Print the data
```

```
For i = 0 to chan.NumberOfPoints - 1
Print varData(i)
Next i
```

**C++ Syntax** HRESULT getSourcePowerCalDataEx(tagNASourcePowerCalBuffer bufSelect, long sourcePort, VARIANT \*pData);

**Interface** IChannel4

---

Last Modified:

March 17, 2010     Added Prior argument

24-Apr-2008     Added note for string names

27-Jun-2007     Updated for PNA-X source port names

**getSourcePowerCalDataScalarEx Method**

**Description** **Note:** This method replaces [getSourcePowerCalDataScalar Method](#)

Retrieves (as scalar values) source power calibration data, if it exists, from this channel.

If the channel is sweeping the source backwards, then the first data point is the highest frequency value; the last data point is the lowest. Use the [Get X-Axis Values2](#) command to return the X-axis values in the displayed order.

**Note:** This method exists on a non-default interface. If you cannot access this method, use the [getSourcePowerCalDataEx](#) Method on IChannel4.

**VB Syntax** *chanData.getSourcePowerCalDataScalarEx buffer, srcPort, numValues, data*

**Variable** [\(Type\)](#) - Description

*chanData* **(interface)** – An [ISourcePowerCalData2](#) interface on the Channel object.

*buffer* **(enum NASourcePowerCalBuffer)** - The requested source power cal data buffer.

**0 - naCorrectionValues** Last iteration of Cal data.

**1 - naPriorIterationCorrectionValues** Prior iteration of Cal data. This argument can be used to determine the final power reading at each point of the power cal, for a cal that did not pass tolerance limits.

The following formula can be used to determine the power reading (in dB):

Power reading = Target power at the source port + specified power cal offset value + 'prior' iteration corr value – actual power corr value.

The "actual" value in this equation is returned with **naCorrectionValues**.

*srcPort* **(long integer)** – The source port for which calibration data is being requested.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

*numValues* **(long integer)** – Number of data values.

[out] – specifies number of data values returned.

[in] – specifies number of values being requested (this must not be larger than the capacity of the data array).

*data* **(single)** – Array to store the data.

**Return Type** Single

**Default** Not Applicable

**Examples**

```
Dim numValues As Long
Dim scalarCalValues() As Single
Dim chanData As ISourcePowerCalData2
Const port1 As Long = 1
numValues = app.ActiveChannel.NumberOfPoints
ReDim scalarCalValues(numValues)
Set chanData = app.ActiveChannel

chanData.getSourcePowerCalDataScalarEx naCorrectionValues,
port1, numValues, scalarCalValues(0)

'Print the data
For i = 0 to numValues - 1
Print scalarCalValues(i)
Next i
```

**C++ Syntax** HRESULT getSourcePowerCalDataScalarEx(tagNASourcePowerCalBuffer bufSelect, long sourcePort, long \*pNumValues, float \*pData);

**Interface** ISourcePowerCalData2

---

Last Modified:

17-Mar-2010	Added Prior argument
24-Apr-2008	Added note for string names
27-Jun-2007	Updated for PNA-X source port names

**GetStandard Method** **Superseded**

**Description** This command has been replaced with [Get StandardByString](#)  
 Returns standard acquisition data from the Cal Set. The returned data is complex pairs.  
 Learn more about [Reading and Writing Cal Data](#)  
 See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** `data = CalSet.getStandard (standard, rcv, src)`

**Variable** [\(Type\)](#) - Description

*data* **(Variant)** Two-dimensional safe array to store the returned data. Memory for the returned Variant is allocated by the PNA and must be released by client.

**Note:** See also [getStandardComplex](#) on the ICalData2 interface to avoid using the variant data type.

*CalSet* A **Cal Set (object)**

*standard* **(enum NACalClass)** Standard data to be read. Choose from:

- 1 - naClassA
- 2 - naClassB
- 3 - naClassC
- 4 - naClassD
- 5 - naClassE
- 6 - naReferenceRatioLine
- 7 - naReferenceRatioThru

**SOLT Standards**

- 1 - naSOLT\_Open
- 2 - naSOLT\_Short
- 3 - naSOLT\_Load
- 4 - naSOLT\_Thru
- 5 - naSOLT\_Isolation

**TRL Standards**

- 1 - naTRL\_Reflection
- 2 - naTRL\_Line\_Reflection

3 - naTRL\_Line\_Tracking

4 - naTRL\_Thru

5 - naTRL\_Isolation

*rcv* **(long)** - Receiver Port

*src* **(long)** - Source Port

**Return Type** **(variant)**

**Default** Not Applicable

**Examples**

```
Dim varStd As Variant
Dim varStd2 As Variant

Cal Set.OpenCalSet( naCalType_TwoPortSOLT, 1, 2)
varStd = CalSet.getStandard(naSOLT_Thru,2,1)
varStd2 = Cal Set.getStandard(naSOLT_Thru,1,2)
Cal Set.CloseCalSet( )
```

**C++ Syntax** HRESULT getStandard(tagNACalClass stdclass, long ReceivePort, long SourcePort, VARIANT\* pData)

**Interface** ICalSet

## GetStandardByString Method

---

**Description** Returns standard acquisition data from the Cal Set. The returned data is complex pairs.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** `data = calSet.GetStandardByString(stdName)`

**Variable** [\(Type\)](#) - Description

*data* **(Variant)** Two-dimensional safe array to store the returned data. Memory for the returned Variant is allocated by the PNA and must be released by client.

**Note:** See also [Get StandardComplexByString](#) on the ICalData2 interface to avoid using the variant data type.

*calSet* A [CalSet](#) (**Object**)

*stdName* **(String)** The string used to identify a particular standard in the Cal Set. An example string requesting the data for the Load standard in a full 2 port cal might be "S11C(3,3)".

**Return Type** Variant

**Default** Not Applicable

**Examples** See an [example](#)

**C++ Syntax** HRESULT GetStandardByString( BSTR bufferName, VARIANT\* pdata)

**Interface** ICalSet2

**GetStandardComplex Method** **Superseded**

**Description** This command is [replaced](#) with [GetStandardComplexByString](#)

Returns standard acquisition data from the Cal Set. The returned data is complex pairs.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**Note:** This method exists on a non-default interface. If you cannot access this method, use the [GetStandard](#) Method on ICal Set

**VB Syntax** *ICalData2.getStandardComplex class, rcv, src, numPts, real(), imag()*

**Variable (Type) - Description**

*ICalData2* An [ICalData2](#) pointer to the Cal Set object

*class* **(enum NACalClass)** Standard data to be read. Choose from:

- 1 - naClassA
- 2 - naClassB
- 3 - naClassC
- 4 - naClassD
- 5 - naClassE
- 6 - naReferenceRatioLine
- 7 - naReferenceRatioThru

**SOLT Standards**

- 1 - naSOLT\_Open
- 2 - naSOLT\_Short
- 3 - naSOLT\_Load
- 4 - naSOLT\_Thru
- 5 - naSOLT\_Isolation

**TRL Standards**

- 1 - naTRL\_Reflection
- 2 - naTRL\_Line\_Reflection
- 3 - naTRL\_Line\_Tracking
- 4 - naTRL\_Thru



## 5 - naTRL\_Isolation

*rcv* **(long)** - Receiver Port

*src* **(long)** - Source Port

*numPts* **(Long)** An In/Out parameter.

On the way **in**, you specify the **max** number of values being requested.

On the way **out**, the PNA returns number of values actually returned.

*real()* **(single)** - array to accept the real part of the calibration data. One-dimensional for the number of data points.

*imag()* **(single)** - array to accept the imaginary part of the calibration data. One-dimensional for the number of data points.

**Return Type** **(single)**

**Default** Not Applicable

**Examples**

```
Dim numpts as long
numpts = ActiveChannel.NumberOfPoints
ReDim r(numpts) ' real part
ReDim i(numpts) ' imaginary part
Dim Cal Set as Cal Set
set Cal Set = pna.GetCalManager.GetCal SetByGUID( txtGUID )
Dim sData As ICalData2
Set sData = Cal Set
sdata.getStandardComplex naSOLT_Open, 1, 1, numpts, r(0), i(0)
```

**C++ Syntax** HRESULT getStandardComplex(tagNACalClass stdclass, long ReceivePort, long SourcePort, long\* pNumValues, float\* pReal, float\* plmag)

**Interface** ICalData2

## GetStandardComplexByString Method

---

**Description** Returns standard acquisition data from the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** *ICalData3.GetStandardComplexByString stdName, InumPoints, real(0), imag(0)*

**Variable** [\(Type\)](#) - **Description**

*ICalData3* An [ICalData3](#) pointer to a [CalSet](#) (**Object**)

*stdName* (**String**) The string used to identify a particular standard in the Cal Set. An example string requesting the data for the Load standard in a full 2 port cal might be "S11C(3,3)".

*InumPoints* (**Long**) An In/Out parameter.  
On the way **in**, you specify the **max** number of values being requested.  
On the way **out**, the PNA returns number of values actually returned.

*real* (**Single**) The real component of the complex data.

*imag* (**Single**) The imaginary component of the complex data.

**Return Value** Single

**Default** Not Applicable

**Examples** [See example](#)

**C++ Syntax** HRESULT GetStandardComplexByString( BSTR bufferName, long\* InumPoints, float\* real, float\* imag);

**Interface** ICalData3

**GetStandardsList Method** **Superseded**

**Description** **Note:** This command is replaced by [CalSet.getStandardList2](#).

Returns the list of Standards contained in this Cal Set for the CalType specified in the [OpenCal Set](#) method. Learn more about [reading and writing Cal Data using COM](#).

The list is a comma separated, textual representation of the error terms with the term name followed by the port path in parentheses.

Standard (n, n),  
Standard (m, n)

Before calling this method you must open the Cal Set with [OpenCal Set](#). If the Cal Set is not open, this method returns E\_NA\_Cal Set\_ACCESS\_DENIED.

Use [StringToNACalClass](#) to convert the list entrees to values that can be used with [GetStandard](#) and [PutStandard](#).

**Note:** The port path designation (m n) indicates the receive and source ports for the measurement. Shorts, opens and loads are single port devices, designated in this list by (n n) where n equals the port to which the device is connected. These devices are all characterized by reflection measurements.

The dual port thru device is characterized by both transmission and reflection measurements in order to compensate for load match and tracking terms.

The notation (n n) indicates the reflection measurement for this device.

The notation (m n) indicates the transmission measurement, where the source and receive ports are different.

**VB Syntax** *CalSet*.**GetStandardsList** (*count*, *list*)

**Variable** [\(Type\)](#) - **Description**

*CalSet* **(object)** - A Cal Set object

*count* **(long [out])** - indicates the number of items returned in the list

*list* **(string)** - Variable to store the returned Comma separated list of items.

**Return Type** String

**Default** Not Applicable

**Examples**

```
dim count as Integer
dim list as string
OpenCalSet (naCalType_TwoPortSOLT, 1, 2)
GetStandardsList( count, list)
CloseCalSet( )
```

Assuming the Cal Set contained the full set of standards for this two port cal, the returned list would be:

```
"Open(1 1),
Short(1 1),
```

```
Load(1 1),  
Thru(1 1),  
Isolation(2 1),  
Open(2 2),  
Short(2 2),  
Load(2 2),  
Thru(2 2),  
Isolation(1 2)  
Thru(2 1),  
Thru(1 2) "
```

**C++ Syntax** HRESULT GetStandardsList( long\* count, BSTR\* list);

**Interface** ICalSet

## GetStandardList2 Method

---

**Description** Returns a list of standards contained by this Cal Set for the specified Cal Type.

**VB Syntax** *list* = *calset*.**GetStandardList2**(*calType*)

**Variable** [\(Type\)](#) - **Description**

*calset* **(object)** - A [CalSet](#) object

*list* **(Variant)** Variant containing a string array of standards for the specified calType.

*calType* **(String)** The string used to identify Cal Set data as belonging to a specific Cal Type. This string is used as a filter so that only the standard names of interest are returned. If the prefix is empty, all names are returned.

An example prefix for a two port cal on ports 2 and 3 might be: "Full 2 Port Cal (2,3)".

**Return Type** Variant

**Default** Not Applicable

**Examples** [See an example](#)

**C++ Syntax** HRESULT GetStandardList2 (BSTR caltype, VARIANT\* list)

**Interface** ICalSet2

## GetStandardsForClass Method

---

**Description** Get the calibration standard numbers for a specified calibration class. To set the calibration number use [SetStandardsForClass Method](#)

**VB Syntax** *calkit*.**GetStandardsForClass** (*calclassorder*, *std1*, *std2*, *std3*, *std4*, *std5*, *std6*, *std7*)

**Variable** [\(Type\)](#) - **Description**

*calKit* A CalKit (**object**)

*calclassorder* (**enum NACalClassOrder**) Choose from:

- 0 - naRefl\_1\_S11
- 1 - naRefl\_2\_S11
- 2 - naRefl\_3\_S11
- 3 - naTran\_1\_S21
- 4 - naRefl\_1\_S22
- 5 - naRefl\_2\_S22
- 6 - naRefl\_3\_S22
- 7 - naTran\_1\_S12
- 8 - naRefl\_1\_S33
- 9 - naRefl\_2\_S33
- 10 - naRefl\_3\_S33
- 11 - naTran\_1\_S32
- 12 - naTran\_1\_S23
- 13 - naTran\_1\_S31
- 14 - naTran\_1\_S13
- 15 - naTRL\_T
- 16 - naTRL\_R
- 17 - naTRL\_L

*std1...std7* (**long**) Calibration Standard Number. Nominal values from **1** through **30**. **0** indicates that a standard number has not been selected.

**Return Type** Not applicable

**Default** Not applicable

**Examples** `calkit.GetStandardsForClass naRef1_3_S11, std1, std2, std3, std4, std5, std6, std7`

**C++ Syntax** HRESULT GetStandardsForClass(NACalClassOrder calclassorder, long std1, long std2, long std3, long std4, long std5, long std6, long std7)

**Interface** ICalkKit

## GetStepDescription Method

---

<b>Description</b>	Returns the description of the specified step in the calibration process. For an ECal User Characterization this returns the description of the specified step in the ECal User Characterization process.
<b>VB Syntax</b>	<i>value</i> = <i>obj</i> . <b>GetStepDescription</b> ( <i>n</i> )
<b>Variable</b>	<b>(Type) - Description</b>
<i>value</i>	(string) - Variable to store the returned number of steps.
<i>obj</i>	Any of the following: <a href="#">GuidedCalibration</a> (object) <a href="#">SMCType</a> (object) <a href="#">VMCType</a> (object) <a href="#">ECalUserCharacterizer</a> (object)
<i>n</i>	(Long) Step in the process. Use <a href="#">GenerateSteps</a> to determine the total number of steps.
<b>Return Type</b>	String
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>value = SMC.GetStepDescription(5)</code>
<b>C++ Syntax</b>	HRESULT get_GetStepDescription(long step, BSTR* str);
<b>Interface</b>	IGuidedCalibration SMCType VMCType IECalUserCharacterizer

---

Last Modified:

4-Nov-2008    Added ECal object



## GetSupportedALCModes Method

---

**Description** Returns the valid ALC Modes for the PNA.  
See [ALCLevelingMode](#) for a list of supported ALC Modes.

**VB Syntax** *value* = *chan*.**GetSupportedALCModes** (*sourcePort*)

**Variable** [\(Type\)](#) - **Description**

*value* Variant Array variable to store the returned valid ALC Modes.

*chan* **(object)** - A [Channel](#) object

*sourcePort* **(long integer)** - Source port.

**Note:** If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use [chan.getPortNumber](#) to translate the string into a port number. To learn more see [Remotely Specifying a Source Port](#).

**Return Type** Variant array

**Default** Not Applicable

**Examples** `modes = chan.GetSupportedALCModes (4) 'Read`

**C++ Syntax** HRESULT GetSupportedALCModes(long port, VARIANT \* ALCModes );

**Interface** IChannel9

---

### Last Modified:

22-Apr-2010 Fixed C++syntax and interface (9)  
24-Apr-2008 Added note for string names  
30-Apr-2007 Edited for src strings  
16-Jan-2007 MX New topic

## GetTestResult Method

---

**Description** Returns the result of limit line testing. There are three ways to use this command:

- If neither optional parameter is specified, limit results for ALL data is returned.
- If one parameter is specified (*start*), the limit result for that data point is returned.
- If both parameters are specified, limit results are returned beginning with *start*, and ending with (*start+size*)-1

**VB Syntax** `testRes = limts.GetTestResult [start,size]`

**Variable** [\(Type\)](#) - **Description**

*testRes* **(enum NALimitTestResult)** - A dimensioned variable to store test results. If a limit line is not tested, a PASS is returned.

0 - naLimitTestResult\_None

1 - naLimitTestResult\_Fail

2 - naLimitTestResult\_Pass

*limts* A LimitTest **(object)**

*start* **(long)** - Optional argument. A start data point number to return limit test results.

*size* **(long)** - Optional argument. Number of data points from *start* to return limit test results.

**Return Type** Long Integer

**Default** Not Applicable

**Examples**

```
Dim testRes As NALimitTestResult
testRes = limts.GetTestResult
Select Case testRes

Case 1
Print "Fails"

Case 2
Print "Pass"

End Select
```

**C++ Syntax** HRESULT GetTestResult(long lStart, long lSize, tagNALimitTestResult \*pVal)

**Interface** ILimitTest

## GetTraceStatistics Method

---

**Description** Returns all four Trace Statistics. To retrieve individual Trace statistics, use [Mean](#), [PeakToPeak](#), [StandardDeviation](#) properties. Use [ShowStatistics](#) to display the statistics of the screen.

**VB Syntax** *meas*.**GetTraceStatistics** *pp,mean,stdev*

**Variable** [\(Type\)](#) - **Description**

*meas* A Measurement (**object**)

*pp,mean,stdev* (**double**) - Dimensioned variables to store the returned values

**Return Type** Double

**Default** Not Applicable

**Examples**

```
'Dimension variables
Dim pp As Double
Dim mean As Double
Dim stdv As Double
meas.GetTraceStatistics pp, mean, stdv
```

**C++ Syntax** HRESULT GetTraceStatistics(double\* pp, double\* mean, double\* stdDeviation)

**Interface** IMeasurement

## GetXAxisValues2 Method

---

**Description** Returns the channel's X-axis values into a dimensioned Typed array. GetXAxisValues2 is a convenient method for determining the frequency of each point when the points are not linearly spaced - as in segment sweep.

**Note:** This method will fail if called using a scripting client such as VBScript or Agilent Vee, [\(see remarks\)](#)

**Note:** In Segment Sweep, chan.[NumberOfPoints](#) will return the total number of data points for the combined segments.

**VB Syntax** *chan*.GetXAxisValues2 *numPts*,*data*

**Variable** [\(Type\)](#) - Description

*chan* **(object)** - A Channel object

*numPts* **(long integer)** - Number of data points in the channel

*data* **(double)** Single dimensioned array of data matching the number of points in the channel.

**Return Type** double

**Default** Not applicable

**Examples**

```
Dim App As Application
Set App = New Application
Dim numPoints As Long
Dim values() As Double
numPoints = App.ActiveChannel.NumberOfPoints
ReDim values(numPoints)
App.ActiveChannel.GetXAxisValues2 numPoints, values(0)
Print values(0), values(1)
```

**C++ Syntax** HRESULT GetXAxisValues2(long\* pNumValues, double\* stimulus)

**Interface** IChannel

### Remarks:

This method will fail if called using a scripting client such as VBScript or Agilent Vee. Use the [GetXAxisValues](#) method as a replacement for these COM environments.

This method also cannot be called using late-bound typing in Visual Basic. For instance, if, in the example above, the first line were replaced with "Dim App as Object", then this method would fail.

## GetXAxisValues Method

---

**Description** Returns the stimulus values for the measurement. To understand how this property is useful, see [IMeasurement2 Interface](#).

**VB Syntax** *data* = *meas*.GetXAxisValues

**Variable** [\(Type\)](#) - [Description](#)

*data* **(Variant)** Array to store the data.

*meas* A Measurement **(object)**

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
Dim varData As Variant
Dim i As Integer
varData = meas.GetXAxisValues
'Print Data
For i = 0 To meas.NumberOfPoints - 1
Print varData(i)
Next i
```

[See C++ example](#)

**C++ Syntax** HRESULT GetXAxisValues(VARIANT\* xData);

**Interface** IMeasurement2

---

Last Modified:

15-Oct-2007    Added link to C++ example

## GetXAxisValues Method

---

**Description** Returns the channel's X-axis values. GetXAxisValues is a convenient method for determining the frequency of each point when the points are not linearly spaced - as in segment sweep.

See the [Measurement2 Interface](#) to learn how this method differs from [meas.GetXAxisValues](#).

**Note:** This method returns a variant which is less efficient than [GetXAxisValues2](#).

**Note:** In Segment Sweep, chan.NumberOfPoints will return the total number of data points for the combined segments.

**VB Syntax** `data = chan.GetXAxisValues`

**Variable** [\(Type\)](#) - Description

`data` Variant array to store the data.

`chan` A Channel (**object**)

**Return Type** Variant

**Default** Not Applicable

**Examples**

```
Dim varData As Variant
Dim i As Integer
varData = chan.GetXAxisValues
'Print Data
For i = 0 To chan.NumberOfPoints - 1
    Print varData(i)
Next i
```

**C++ Syntax** HRESULT GetXAxisValues (VARIANT\* xData)

**Interface** IChannel

## HasCalType Method

---

**Description** Verifies that the Cal Set object contains the error terms required to perform the specified correction (CalType) to an appropriate measurement.

The argument list includes specifiers for up to 3 ports. The number of arguments required depends on the CalType specified. The value for each port is set to 0 if not specified.

**VB Syntax** *check* = *CalSet*.**HasCalType** (*calType*, *p1*, *p2*, *p3*)

**Variable** [\(Type\) - Description](#)

*check* **(boolean)** - variable to store the returned value

**TRUE (1)** - Cal Set has all of the error terms necessary to apply the specified correction CalType.

**FALSE(0)** - Cal Set DOES NOT have all of the error terms necessary to apply the specified CalType.

*CalSet* **(object)** - A Cal Set object

*calType* (enum as naCalType) - type of correction to be applied. Choose from:

Caltype	p arguments required
0 - naCalType_Response_Open	p1
1 - naCalType_Response_Short	p1
2 - *naCalType_Response_Thru	p1 (rcv), p2 (src)
3 - *naCalType_Response_Thru_And_Isol	p1 (rcv), p2 (src)
4 - naCalType_OnePort	p1
5 - naCalType_TwoPort_SOLT	p1, p2
6 - naCalType_TwoPort_TRL	p1, p2
7 - naCalType_None	N/A
8 - naCalType_ThreePort_SOLT	p1, p2, p3
9 - Custom	N/A
10 - naCalType_FourPort_SOLT	p1, p2, p3 (port 4 is assumed)

\* order of port arguments is significant for these CalTypes

*p1* **(long)** - required. This argument must be specified.

This specifies either:

- the one significant port for an open/short response cal or a 1 port cal.
- or one of the ports involved in a 2, 3, or 4 port cal
- or the **receive** port for a thru response / thru-isolation cal.

*p2* **(long)** - required for any CalType involving more than one port

This specifies either:

- one of the ports involved in a 2, 3, or 4 port cal (order independent)
- or the **source** port for a thru response / thru-isolation cal

*p3* **(long)** - required for 3 and 4-port cal

This specifies one of the ports involved in a 3 or 4 port cal (order independent)

**Return Type** **VARIANT\_BOOL**

**Default** Not Applicable

**Examples** `value = CalSet.HasCalType(naCalType_TwoPort_TRL, 1, 2)`

**C++ Syntax** `HRESULT HasCalType( tagNACalType, long port1, long port2, long port3, BOOL *pVal);`

**Interface** ICalSet



## Hold Method

---

**Description** Puts the channel in Hold - not sweeping.  
See [chans.Hold](#) to put ALL channels in hold.

**VB Syntax** *chan.Hold [sync]*

**Variable** [\(Type\)](#) - Description

*chan* A [Channel](#) (object)

*[sync]* The [sync] argument is ignored.  
Program control ALWAYS waits until the channel is in the Hold state.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `chan.Hold`

**C++ Syntax** HRESULT Hold

**Interface** IChannel

---

Last Modified:

20-Apr-2010 Updated sync

## Hold (channels) Method

---

<b>Description</b>	Places ALL channels in hold mode.  To resume all channels sweeping, use <a href="#">chans.Resume</a> . (Must be the same instance of chans).  To place a single channel in hold mode, use <a href="#">channel.Hold</a> Method.
<b>VB Syntax</b>	<i>chans.Hold</i>
<b>Variable</b>	<b><a href="#">(Type)</a> - Description</b>
<i>chans</i>	A <a href="#">Channel collection</a> ( <b>object</b> )
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>chans.Hold</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT Hold();
<b>Interface</b>	Channels2

## Initialize Method

---

**Description** Begins a calibration.

**Note:** *chan* must be the active channel.

For ECal User Characterization, use [Initialize \(ECal\)](#).

**VB Syntax** *obj.Initialize* (*chan*, *useCalStorPref*)

**Variable** [\(Type\)](#) - **Description**

*obj* Any of the following:  
[GuidedCalibration](#) (object)  
[SMCType](#) (object)  
[VMCType](#) (object)

*chan* (Long) Channel number to calibrate.

*useCalStorPref* (boolean)

**True** or **1** - Assignment of Cal Set will be based on the setting of the [RemoteCalStoragePreference](#) COM property.

**False** or **0** – If the channel currently has a selected Cal Set, the calibration will be stored to that Cal Set. Otherwise, the assignment of Cal Set is based upon the setting of the [RemoteCalStoragePreference](#) COM property.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `smc.Initialize(2, True)`

**C++ Syntax** HRESULT put\_Initialize(long channelNumber, VARIANT\_BOOL bCalPref);

**Interface** IGuidedCalibration  
 SMCType  
 VMCType

---

Last Modified:

4-Nov-2008    Added Note about ECal User Char

**Initialize Method Superseded**

---

<b>Description</b>	<b>Note:</b> This command is replaced with <a href="#">InitializeEx Method</a> Initiates a User Characterization of an ECal module. The specified channel number must be an S-parameter measurement channel. The User characterization number must already be set before issuing this command using <a href="#">CharacterizationNumber Property</a> .
<b>VB Syntax</b>	<i>userChar</i> . <b>Initialize</b> ( <i>chan</i> )
<b>Variable</b>	<b>(Type) - Description</b>
<i>userChar</i>	<a href="#">ECalUserCharacterizer</a> (object)
<i>chan</i>	(Long) Channel number of a calibrated S-parameter channel.
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>userchar.Initialize(2)</code>
<b>C++ Syntax</b>	HRESULT put_Initialize(long chanNum);
<b>Interface</b>	IECalUserCharacterizer

---

Last Modified:

4-Nov-2008    New topic (8.33)

## Item Method

---

**Description** Returns an object from the collection of objects.

**Notes**

- The order of objects within a collection cannot be assumed.
- Most, but not all, PNA Collections are '1-based'

**VB Syntax** *Object* [.Item](*n*)

**Variable** [\(Type\)](#) - Description

*Object* Any of the following (**collections**):

- [CalFactorSegments collection](#)
- [CalFactorSegmentsPMAR Collection](#)
- [Cal Sets collection](#)
- [Channels collection](#)
- [E5091Testset collection](#)
- [ECalModules Collection](#)
- [ExternalDevices Collection](#)
- [ExternalTestsets collection](#)
- [FOM collection](#)
- [LimitTest collection](#)
- [Measurements collection](#)
- [NaWindows collection](#)
- [PowerLossSegments collection](#)
- [PowerLossSegmentsPMAR Collection](#)
- [PowerSensors collection](#)
- [Segments collection](#)
- [Traces collection](#)
- [PowerMeterInterfaces Collection](#)

[Learn more about collections in the PNA](#)

**.Item** Optional - Item is the default property of a collections object and therefore can be called implicitly. For example, the following two commands are equivalent:

```
Channels.Item(3).Averaging = 1  
Channels(3).Averaging = 1
```

*n*    **(variant)** - number of the item in the collection.

- The **Measurements, Traces, and FOM** collections allow you to specify the name of the measurement as a string. For example:  
`measCollection("CH_S11_1").InterpolateMarkers`
- The [Cal Sets collection](#) allows you to specify the name of the cal set. For example:  
`Calsets("MyCalSet").Description = "New Description"`
- The [ExternalDevices Collection](#) allows you to specify the name of the external device. For example:  
`Set extDev = externalDevices.Item("NewPMAR")`

**Return Type**    **(Object)**

**Default**    Not Applicable

**Examples**

```
For i = 1 to Traces.Count 1
    Traces.Item(i).YScale = .5dB
Next i
```

**C++ Syntax**    HRESULT Item(VARIANT index, <interface>\*\* pItem)

**Interfaces**    All listed above.

---

Last Modified:

10-Sep-2008    Added CalSets name note

## LANConfigurationInitialize Method

---

<b>Description</b>	Performs an initialization (reset) of the PNA's LAN configuration, as dictated by Section 8.14 of the LAN eXtensions for Instrumentation (LXI) standard (Version 1.1). This performs the same operation as pressing the LAN Reset button on the PNA <a href="#">LAN Status dialog</a> .
<b>VB Syntax</b>	<i>app</i> . <b>LANConfigurationInitialize</b>
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>app</i>	An <a href="#">Application</a> (object)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.LANConfigurationInitialize</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT LANConfigurationInitialize ();
<b>Interface</b>	IApplication13

---

Last Modified:

2-Jun-2008    MX New topic

## LaunchCalWizard Method

---

**Description** Launches the S-parameter Cal Wizard on the PNA and does not return until the Cal Wizard is dismissed.

To launch the Cal Wizard for a PNA Application, use the [LaunchDialog Method](#).

**Note:** The Cal Wizard operates on the active measurement. Therefore, activate the measurement to be calibrated before launching the Cal Wizard.

**VB Syntax** *success* = *app*.LaunchCalWizard (*newCS*)

**Variable** [\(Type\)](#) - Description

*success* **(boolean)** - variable to store the returned value

**True** - The Cal was completed

**False** - The Cal was canceled without completing the calibration.

*app* An [Application](#) (object)

*newCS* **(boolean)**

**True** - Cal will be performed on a new Cal Set.

**False** - Cal will be performed using the existing Cal Set assigned to the channel. If no Cal Set is found, a new Cal Set will be created.

**Return Type** Boolean

**Default** Not Applicable

**Example**

```
dim bSuccess as boolean
dim bNewCalset as boolean
bNewCalSet = false
bSuccess = app.LaunchCalWizard( bNewCalSet)
```

**C++ Syntax** HRESULT LaunchCalWizard(VARIANT\_BOOL bCalsuccess)

**Interface** IApplication

---

Last Modified:

5-Sep-2008    S-parameter only



## LaunchDialog Method

---

**Description** Launches the specified dialog box.

The Calibration Wizard dialog that appears depends on the active channel. For example, if a Gain Compression channel is active, then the GCA Cal Wizard appears. Use [meas.Activate](#) to activate a measurement and channel.

Remote operation returns after the dialog is dismissed.

To invoke the Cal Wizard and have it return immediately, then use [Syst:Corr:Wiz](#) with the [SCPI Parser object](#).

**VB Syntax** `app.LaunchDialog dialog, [data]`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*dialog* **(String)** Dialog box to launch. Choose from:

"SourcePowerCal" [See this dialog.](#)

"PowerMeterSettings" [See this dialog.](#)

"PathConfiguration" [See this dialog.](#)

"CalibrationWizard" Depends on the channel

"CalibrationSelection" [See this dialog.](#)

*[data]* (Optional argument) Reserved for future use.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.LaunchDialog "SourcePowerCal"`

**C++ Syntax** `LaunchDialog( BSTR dialog, [defaultvalue(0)] VARIANT dialogData)`

**Interface** IApplication11

[See the PNA Object Model](#)

---

### Last Modified:

4-Sep-2008 Added new dialogs

1-Jun-2007 Added optional argument

16-Feb-2007 MX New topic



## LaunchPowerMeterSettingsDialog Method

---

**Description** Launches the [Power Meter Settings dialog](#) on the PNA. Changing certain values from that dialog will change values of the corresponding properties on this COM object.

**VB Syntax** *pwrCal*.**LaunchPowerMeterSettingsDialog**

**Variable**    [\(Type\)](#) - Description

*pwrCal*    A SourcePowerCalibrator (**object**)

**Return Type**    None

**Default**    Not Applicable

**Examples**    `powerCalibrator.LaunchPowerMeterSettingsDialog`

**C++ Syntax**    HRESULT put\_LaunchPowerMeterSettingsDialog();

**Interface**    ISourcePowerCalibrator2

## LoadConfiguration Method

---

<b>Description</b>	Loads the named configuration onto the specified channel. Use <a href="#">Configurations Method</a> to return the configuration names that are stored on the PNA.
<b>VB Syntax</b>	<i>pathMgr.LoadConfiguration ch, name</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>pathMgr</i>	<a href="#">PathConfigurationManager</a> ( <b>object</b> )
<i>ch</i>	<b>(Long)</b> Channel number of the configuration to be saved.
<i>name</i>	<b>(String)</b> Configuration name. "Default" is the default factory configuration.
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>path.LoadConfiguration 2, "myMixer"</code>
<b>C++ Syntax</b>	HRESULT LoadConfiguration (long channelNum, BSTR configName );
<b>Interface</b>	IPathConfigurationManager

---

Last Modified:

14-Dec-2006    MX New topic

## LoadENRFile Method

---

**Description** Loads an ENR file from disk into PNA memory. This file is typically provided by the manufacturer of the noise source.

**VB Syntax** *enr*.LoadENRFile (*filename*)

**Variable** [\(Type\)](#) - Description

*enr* An [ENRFile](#) (object)

*filename* (String) - Absolute path and filename of the ENR file.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See example program](#)

**C++ Syntax** HRESULT LoadENRFile(BSTR filename);

**Interface** IENRFile

---

Last Modified:

2-Aug-2007    MX New topic

Write only

## LoadFile Method

---

**Description** Loads a previously-configured mixer attributes file (.mxr)

**VB Syntax** *obj*.LoadFile (*filename*)

**Variable** [\(Type\)](#) - Description

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*filename* (String) Full path, file name, and .mxr extension of the mixer attributes file.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents".

**Return Type** String

**Default** Not Applicable

**Examples** `mixer.LoadFile ("C:/Program Files/Agilent/Network Analyzer/Documents/myMixer.mxr")`

**C++ Syntax** HRESULT LoadFile(BSTR newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009    Added converter

## ManualTrigger Method

---

**Description** Triggers the analyzer when [TriggerSetup.Source](#) = `naTriggerManual`.

**Note:** An **SMC Fixed Output** measurement cannot be triggered using this command. For more information, see the [example program](#).

**VB Syntax** `app.ManualTrigger [sync],[timeout]`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*[sync]* **(boolean)** - Optional argument.  
A variable set to either True or False.

**True** - The analyzer waits until the trigger is completed to process subsequent commands.

**False** - Subsequent commands are processed immediately (the default setting).

*timeout* **(long)** - Optional argument.  
If *sync* is true, *timeout* sets the amount of time the PNA will wait until continuing program execution. Units are milliseconds. A value of -1 (the default setting) causes the PNA to wait indefinitely.

If *sync* is False, the timeout setting is ignored.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
' After Manual trigger is executed, the PNA will wait 1 second
to continue program execution
Dim wait as Boolean
wait = True
app.ManualTrigger wait, 1000
```

**C++ Syntax** HRESULT ManualTrigger(VARIANT\_BOOL bSynchronize, long timeout)

**Interface** IApplication

---

Last Modified:

12-Jul-2007      [Modified link](#)

## MessageText Property

---

**Description** Returns text for the specified eventID

**VB Syntax** *app*.**MessageText**,*eventID*,*message*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (**object**)

*eventID* (**enum naEventID**) Choose from the list in [Working with the Analyzer's Events](#)

*message* (**string**) - variable to store the returned message

**Return Type** String

**Default** Not Applicable

**Examples** `RFNA.MessageText naEventID_ARRANGE_WINDOW_EXCEED_CAPACITY,  
message`

**C++ Syntax** HRESULT get\_MessageText( tagNAEventID msgID, BSTR\* message)

**Interface** IApplication



## NextIFBandwidth Method

---

**Description** A function that returns the Next higher IF Bandwidth value. Use to retrieve the list of available IFBandwidth settings.

**VB Syntax** *chan*.Next\_IFBandwidth *bw*

**Variable** [\(Type\)](#) - Description

*chan* A Channel (**object**)

*bw* (**double**) - The argument that you use to send an IFBandwidth. The function uses this argument to return the Next higher IFbandwidth.

**Return Type** Double

**Default** Not Applicable

**Examples**

```
Public pnbw As Double 'declare variable outside of procedure  
  
pnbw = chan.IFBandwidth 'put the current IFBW in pnbw  
chan.Next_IFBandwidth pnbw 'function returns the Next higher  
IFBandwidth.  
chan.IFBandwidth = pnbw 'set IFBW to the Next value
```

**C++ Syntax** HRESULT Next\_IFBandwidth (double \*pVal)

**Interface** IChannel

## NumberOfGroups Method

---

**Description** Sets the number of trigger signals the channel will receive. After the channel has received that number of trigger signals, the channel switches to Hold mode.

**VB Syntax** *chan*.**NumberOfGroups** *num*, *sync*

**Variable** [\(Type\)](#) - **Description**

*chan* A [Channel](#) (object)

*num* **(long integer)** Number of trigger signals the channel will receive. Choose any number between 1 and 2 million.

*sync* **(boolean)**

Variable set to either:

**True** - subsequent commands are not processed until the groups are complete. Do not use with manual trigger.

**False** - subsequent commands are processed immediately.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `chan.NumberOfGroups 5, False`

**C++ Syntax** HRESULT NumberOfGroups(long count, VARIANT\_BOOL bWait)

**Interface** IChannel

**OpenCalSet Method** **Superseded**

**Description** This command is no longer necessary. The CalSet.get... and put... commands that required this command have been replaced,

Open the Cal Set to read/write a particular **CalType**. Learn more about [reading and writing Cal Data using COM](#).

This method is a prerequisite to several other Cal Set methods.

A Cal Set can contain more than one CalType. This method opens the Cal Set and allows access to a particular set of terms. Subsequent commands like [getErrorTerm](#) use this information to access the correct error terms in the Cal Set. For example:

```
cset.OpenCalSet (naCalType_TwoPortSOLT, 3, 2)
cset.PutErrorTerm(naDirectivity, 1, 1, Buffer)
```

The directivity error term for port 1 could belong to any number of caltypes: Full1Port (S11), Full2Port (12), Full2Port (13) or Full3Port (123). The **CalType and port** specifiers in OpenCalSet directs the uploaded directivity term to the correct set of error terms.

To close the Cal Set, see [CloseCalSet](#).

**VB Syntax** *CalSet.OpenCalSet (CalType, p1, p2, p3)*

**Variable** [\(Type\) - Description](#)

*CalSet* **(object)** - A Cal Set object

*CalType* (enum as naCalType) - type of correction to be applied. Choose from:

Caltype	p arguments required
0 - naCalType_Response_Open	p1
1 - naCalType_Response_Short	p1
2 - *naCalType_Response_Thru	p1 (rcv), p2 (src)
3 - *naCalType_Response_Thru_And_Isol	p1 (rcv), p2 (src)
4 - naCalType_OnePort	p1
5 - naCalType_TwoPort_SOLT	p1, p2
6 - naCalType_TwoPort_TRL	p1, p2
7 - naCalType_None	N/A
8 - naCalType_ThreePort_SOLT	p1, p2, p3
9 - Custom	N/A

**10** - naCalType\_FourPort\_SOLT                      p1, p2, p3  
(port 4 is assumed)

\* order of port arguments is significant for these CalTypes

*p1* **(long)** - required. This argument must be specified.

This specifies either:

- the one significant port for an open/short response cal or a 1 port cal.
- or one of the ports involved in a 2 or 3 port cal
- or the **receive** port for a thru response / thru-isolation cal.

*p2* **(long)** - required for any caltype involving more than one port

This specifies either:

- one of the ports involved in a 2 or 3 port cal (order independent)
- or the **source** port for a thru response / thru-isolation cal

*p3* **(long)** - required only for 3 port cal

This specifies either:

- one of the ports involved in a 3 port cal (order independent)

**Return Type**    None

**Default**        Not Applicable

**Examples**        `CalSet.OpenCalSet naCalType_ThreePort_SOLT, 3,2,1`

**C++ Syntax**     HRESULT OpenCalSet ( naCalType, port1, [optional] port2, [optional]  
port3);

**Interface**        ICalSet

## Parse Method

---

**Description** Allows the use of COM to send a SCPI command. [See a C++ example](#) of how to return error information when using this command.

**Note:** The SCPIStringParser Methods can NOT be used with [SCPI Status Reporting](#). However, the \*OPC? will work.

**VB Syntax** *scpi*.Parse ("SCPI command")

**Variable** [\(Type\)](#) - Description

*scpi* A [ScpiStringParser](#) (object)

*SCPI command* **(string)** - Any valid SCPI command

**Return Type** String

**Default** Not Applicable

**Examples**

```
Dim scpi As ScpiStringParser
Set scpi = app.ScpiStringParser
Dim startfreq As Double
startfreq = 100e6
'
scpi.Parse "Sens:Freq:Start " & startfreq'Write

Dim str As String
str = scpi.Parse ("Sens:Freq:Start?") 'Read
```

**C++ Syntax** HRESULT Parse(BSTR SCPI\_Command, BSTR \*pQueryResponse)

**Interface** IScpiStringParser

---

Last Modified:

April 27, 2009    Added note

1-Jan-2007    Corrected example

## Preset Method

---

<b>Description</b>	<b>Application Object:</b> Deletes all traces and windows. In addition, resets the analyzer to factory defined default settings and creates an S11 measurement named "CH1_S11_1" in window 1.  <b>Channel Object:</b> Resets the channel (object) to factory defined default settings. Does NOT delete the current measurements or add a new measurement.
<b>VB Syntax</b>	<i>app</i> .Preset <i>chan</i> .Preset
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>app</i>	An <a href="#">Application</a> (object)
<i>chan</i>	A Channel (object)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.Preset</code>
<b>C++ Syntax</b>	HRESULT Preset()
<b>Interface</b>	IApplication IChannel

## PreviousIFBandwidth Method

---

**Description** A function that returns the previous IF Bandwidth value. Use to retrieve the list of available IFBandwidth settings.

**VB Syntax** *chan.Previous\_IFBandwidth bw*

**Variable** [\(Type\)](#) - Description

*chan* A Channel (**object**)

*bw* (**double**) - The argument that you use to send an IFBandwidth. The function uses this argument to return the previous IFbandwidth.

**Return Type** Double

**Default** Not Applicable

**Examples**

```
Public pnbw As Double 'declare variable outside of procedure
PreBW = chan.IFBandwidth 'put the current IFBW in PreBW
chan.Previous_IFBandwidth PreBW 'function returns the Previous
IFBandwidth of the current one.
chan.IFBandwidth = PreBW 'set IFBW to the previous value
```

**C++ Syntax** HRESULT Previous\_IFBandwidth (double \*pVal)

**Interface** IChannel

## PrintToFile Method

---

**Description** Saves the screen image to a bitmap file.

**VB Syntax** *app.PrintToFile filename*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

*filename* **(string)** Full path, file name, and extension of the screen image file.  
Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents".  
Use one of the following extensions:

- .bmp - not recommended due to large file size
- .jpg - not recommended due to poor quality
- **.png - recommended**

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.PrintToFile "C:/Program Files/Agilent/Network Analyzer/Documents/myfile.png"`

**[C++ Syntax](#)** HRESULT PrintToFile(BSTR bstrFile)

**Interface** IApplication



## PutComplex Method

---

**Description** Puts real and imaginary data into the specified location. This method forces the channel into Hold mode to prevent the input data from being overwritten. Learn more about [reading and writing Cal Data using COM](#).

Data put in the raw data store will be **re-processed** whenever a change is made to the measurement attributes such as format or correction.

Data put in the measurement results store will be **overwritten** by any measurement attribute changes.

See also [putNAComplex](#)

**VB Syntax** *measData.putComplex location, numPts, real(), imag(), [format]*

**Variable** [\(Type\) - Description](#)

*measData* An IArrayTransfer interface which supports the Measurement object

*location* **(enum NADataStore)** Where the Data will be put. Choose from:

0 - naRawData

1 - naCorrectedData

2 - naMeasResult

3 - naRawMemory

4 - naMemoryResult

5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

*numPts* **(long integer)** - Number of data points in the channel

*real()* **(single)** - Array containing real data values

*imag()* **(single)** - Array containing imaginary data values

*format* **(enum NADataFormat)** optional argument - display format of the real and imaginary data. Only used if destination is naMeasResult or naMemoryResult buffer. If unspecified, data is assumed to be in naDataFormat\_Polar

0 - naDataFormat\_LinMag

1 - naDataFormat\_LogMag

2 - naDataFormat\_Phase

3 - naDataFormat\_Polar

4 - naDataFormat\_Smith

5 - naDataFormat\_Delay

6 - naDataFormat\_Real

- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

[Learn more about Data Format.](#)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim measData As IArrayTransfer
Set measData = app.ActiveMeasurement

measData.putComplex naMemoryResult, 201,
real(0), imag(0), naDataFormat_SWR
```

**C++ Syntax** HRESULT putComplex( tagNADataStore DataStore, long lNumValues, float\* pReal, float\* plmag, tagDataFormat displayFormat)

**Interface** IArrayTransfer

---

Last Modified:

1-Oct-2007    Added temperature formats

## PutDataComplex Method

---

**Description** Puts complex data into the specified location. This method forces the channel into Hold mode to prevent the input data from being overwritten.

**VB Syntax** *meas.putDataComplex location, data*

**Variable** [\(Type\)](#) - Description

*meas* A measurement (**object**)

*location* (**enum NADataStore**) Where the Data will be put. Choose from:

0 - naRawData

1 - naCorrectedData

2 - naMeasResult

3 - naRawMemory

4 - naMemoryResult

5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

- Data put in 0 - naRawData will be re-processed whenever a change is made to the measurement attributes such as format or correction.
- Data put in 2 - naMeasResult will be overwritten by any measurement attribute changes.
- When putting data into 3 - naRawMemory:
  1. Put the analyzer in hold mode
  2. Call [DataToMemory](#) to initialize a memory buffer
  3. Call putDataComplex(naRawMemory, data)

This ensures that the memory buffer is appropriately initialized before receiving new data.

*data* (**variant**) - A **two-dimensional** variant array.

**Note:** All buffers except naMeasResult and naMemoryResult require Complex data

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
' Put 201 points worth of raw (complex) data into the measurement
' Note that an array of complex numbers is represented by a 2-D
```

array where the first rank is the number of points, and the 2nd rank is always size 2 (max index 1) representing the Real and Imag parts of the complex number.

```
' complex array of data (2nd dimension of size 2 represents  
Re/Im  
Dim data(200,1) )  
For i = 0 to 200  
' Set Real part of data point i  
data(i,0) = i/200;  
' Set Imag part of data point i  
data(i,1) = i/200;  
Next  
app.ActiveMeasurement.putDataComplex naRawData, data
```

**C++ Syntax** HRESULT putDataComplex(tagNADataStore DataStore, VARIANT complexData)

**Interface** IMeasurement

## PutENRData Method

---

**Description** Write ENR calibration data to PNA memory. All of the frequency and ENR data must be sent at the same time.

**VB Syntax** *enr.PutENRData (vData)*

**Variable** [\(Type\)](#) - Description

*enr* An [ENRFile](#) (object)

*vData* (Variant array) -ENR data. Frequency value in Hz, followed by corresponding ENR value in dB. Enter as many data pairs as necessary.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See example program](#)

**C++ Syntax** HRESULT PutENRData(VARIANT vdata);

**Interface** IENRFile

---

Last Modified:

2-Aug-2007    MX New topic

**PutErrorTerm Method - Superseded**

**Description** **Note:** This command is replaced by [PutErrorTermByString](#)

Puts variant error term data into the error-correction buffer.

[Learn about reading and writing Calibration data.](#)

**VB Syntax** `cal.putErrorTerm(term,rcv, src, data)`

**Variable** [\(Type\)](#) - Description

*cal* A Calibrator **(object)**

*term* **(enum As NaErrorTerm)**  
 naErrorTerm\_Directivity\_Isolation  
 naErrorTerm\_Match  
 naErrorTerm\_Tracking

*rcv* **(long integer)** - Receiver Port

*src* **(long integer)** - Source Port

*data* **(variant)** Error term data in a two-dimensional array (0:1, 0:numpts-1).

To get this	Specify these parameters:		
Error Term	<i>term</i>	<i>rcv</i>	<i>src</i>
Fwd Directivity	naET_Directivity Isolation	1	1
Rev Directivity	naET_Directivity Isolation	2	2
Fwd Isolation	naET_Directivity Isolation	2	1
Rev Isolation	naET_Directivity Isolation	1	2
Fwd Source Match	naErrorTerm_Match	1	1
Rev Source Match	naErrorTerm_Match	2	2
Fwd Load Match	naErrorTerm_Match	2	1
Rev Load Match	naErrorTerm_Match	1	2
Fwd Reflection Tracking	naErrorTerm_Tracking	1	1
Rev Reflection Tracking	naErrorTerm_Tracking	2	2
Fwd Trans Tracking	naErrorTerm_Tracking	2	1
Rev Trans Tracking	naErrorTerm_Tracking	1	2

Fwd Trans Tracking	naErrorTerm_Tracking	2	1
<b>Return Type</b>	Not Applicable		
<b>Default</b>	Not Applicable		
<b>Examples</b>	<pre>Dim varError As Variant varError = cal.putErrorTerm (naErrorTerm_Tracking, 2, 1, VarData)</pre>		
<b><u>C++ Syntax</u></b>	HRESULT putErrorTerm(tagNAErrorTerm ETerm, long ReceivePort, long SourcePort, VARIANT varData)		
<b>Interface</b>	ICalibrator		

**PutErrorTerm Method** **Superseded**

<b>Description</b>	<p>This command is <a href="#">replaced</a> with <a href="#">PutErrorTermByString</a></p> <p>Puts error term data into the Cal Set.</p> <p>Learn more about <a href="#">Reading and Writing Cal Data</a></p> <p>See examples of <a href="#">Reading</a> and <a href="#">Writing</a> Cal Set Data</p>
<b>VB Syntax</b>	<i>CalSet.putErrorTerm (term, rcv, src, data)</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>CalSet</i>	<b>(Object)</b> A <a href="#">CalSet</a> Object
<i>term</i>	<b>(enum As NaErrorTerm2)</b> Error Term. Choose from: 0 - naET_Directivity (src = rcv) 1 - naET_SourceMatch (src = rcv) 2 - naET_ReflectionTracking (src = rcv) 3 - naET_TransmissionTracking (src rcv) 4 - naET_LoadMatch (src rcv) 5 - naET_Isolation (src rcv)
<i>rcv</i>	<b>(long integer)</b> - Receiver Port
<i>src</i>	<b>(long integer)</b> - Source Port
<i>data</i>	<b>(variant)</b> Error term data in a two-dimensional array (0:1, 0:numpts-1). The data must be complex pairs.  <b>Note:</b> See also <a href="#">PutErrorTermComplex</a> on the ICalData2 interface to avoid using the variant data type.
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<a href="#">See an Example</a>
<b>C++ Syntax</b>	HRESULT putErrorTerm(tagNAErrorTerm2 ETerm, long ReceivePort, long SourcePort, VARIANT varData)
<b>Interface</b>	ICalSet



## PutErrorTermByString

---

**Description** Puts error term data into the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** `calSet.PutErrorTermByString(errorName, vdata)`

**Variable** [\(Type\) - Description](#)

*calSet* **(Object)** A [CalSet](#) Object

*errorName* **(String)** The string name used to identify a particular error term in the Cal Set. An example string for port 3 directivity in a full 2 port cal might be "Directivity(3,3)". To determine the string names of error terms, see [GetErrorTermList2](#).

*vdata* **(Variant)** This data array is usually two dimensional. Each element is a type single. The two elements represent the real and imaginary parts of a complex pair.

**Note:** This structure is compatible with scripting clients who can only use variants. For alternative methods that use typed arrays, see ICalData3.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See an Example](#)

**C++ Syntax** HRESULT PutStandardByString( BSTR bufferName, VARIANT vardata)

**Interface** ICalSet2

**PutErrorTermComplex Method** **Superseded**

<b>Description</b>	<b>Note:</b> This command is replaced by <a href="#">PutErrorTermComplexByString</a> Puts error term data into the error-correction data buffer. Learn more about <a href="#">reading and writing Cal data using COM</a>
<b>VB Syntax</b>	<code>data.putErrorTermComplex term, rcv, src, numPts, real(), imag()</code>
<b>Variable</b>	<b>(Type) - Description</b>
<i>data</i>	An ICalData pointer to the Calibrator object
<i>term</i>	<b>(enum NAEErrorTerm)</b> - The error term to be retrieved. Choose from: <ul style="list-style-type: none"> <li>• <b>naErrorTerm_Directivity_Isolation</b></li> <li>• <b>naErrorTerm_Match</b></li> <li>• <b>naErrorTerm_Tracking</b></li> </ul>
<i>rcv</i>	<b>(long integer)</b> - Receiver Port
<i>src</i>	<b>(long integer)</b> - Source Port
<i>numPts</i>	<b>(long integer)</b> - number of data points in the array
<i>real()</i>	<b>(single)</b> - array containing the <b>real</b> part of the calibration data. One-dimensional: the number of data points.
<i>imag()</i>	<b>(single)</b> - array containing the <b>imaginary</b> part of the calibration data. One-dimensional: the number of data points.

To get this	Specify these parameters:		
Error Term	<i>term</i>	<i>rcv</i>	<i>src</i>
Fwd Directivity	naET_Directivity Isolation	1	1
Rev Directivity	naET_Directivity Isolation	2	2
Fwd Isolation	naET_Directivity Isolation	2	1
Rev Isolation	naET_Directivity Isolation	1	2
Fwd Source Match	naErrorTerm_Match	1	1
Rev Source Match	naErrorTerm_Match	2	2
Fwd Load Match	naErrorTerm_Match	2	1
Rev Load Match	naErrorTerm_Match	1	2

Fwd Reflection Tracking	naErrorTerm_Tracking	1	1
Rev Reflection Tracking	naErrorTerm_Tracking	2	2
Fwd Trans Tracking	naErrorTerm_Tracking	2	1
Rev Trans Tracking	naErrorTerm_Tracking	1	2
Fwd Trans Tracking	naErrorTerm_Tracking	2	1

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim eData As ICalData
Set eData = chan.Calibrator
eData.putErrorTermComplex naErrorTerm_Directivity_Isolation, 1,
1, 201, rel(0), img(0)
```

**C++ Syntax** HRESULT putErrorTermComplex(tagNAErrorTerm ETerm, long ReceivePort, long SourcePort, long\* pNumValues, float\* pReal, float\* plmag)

**Interface** ICalData

**PutErrorTermComplex Method** **Superseded**

**Description** This command is [replaced](#) with [PutErrorTermComplexByString](#)

Puts error term data into the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** `data.putErrorTermComplex term, rcv, src, numPts, real(), imag()`

**Variable** [\(Type\)](#) - Description

*data* An [ICalData2](#) pointer to a Cal Set object

*term* **(enum NAEErrorTerm2)** - The error term to be written. Choose from:

0 - naET\_Directivity

1 - naET\_SourceMatch

2 - naET\_ReflectionTracking

3 - naET\_TransmissionTracking

4 - naET\_LoadMatch

5 - naET\_Isolation

*rcv* **(long)** - Receiver Port

*src* **(long)** - Source Port

*numPts* **(long)** - number of data points in the real and imaginary arrays.

*real()* **(single)** - array containing the **real** part of the calibration data. One-dimensional: the number of data points.

*imag()* **(single)** - array containing the **imaginary** part of the calibration data. One-dimensional: the number of data points.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim eData As ICalData2
Set eData = app.GetCalManager.Cal Sets.Item(1)
eData.putErrorTermComplex naET_LoadMatch, 1, 2, numpts, rel(0),
img(0)
```

**C++ Syntax** HRESULT putErrorTermComplex(tagNAErrorTerm2 ETerm, long ReceivePort, long SourcePort, long\* pNumValues, float\* pReal, float\* plmag)

**Interface** ICalData2

## PutErrorTermComplexByString Method

---

**Description** Puts error term data into the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** *ICalData3.PutErrorTermComplexByString* *errorName, InumPoints, real(0), imag(0)*

**Variable** [\(Type\)](#) - Description

*ICalData3* An [ICalData3](#) pointer to a Cal Set object.

*errorName* **(String)** The string name used to identify a particular error term in the Cal Set. An example string for port 3 directivity in a full 2 port cal might be "Directivity(3,3)". To determine the string names of error terms, see [GetErrorTermList2](#).

*InumPoints* **(Long)** The number of data points in the real and imaginary arrays.

*real* **(Single)** The real component of the complex data.

*imag* **(Single)** The imaginary component of the complex data.

**Note:** The size of the real and imaginary arrays should be the same.

**Return Value** Not Applicable

**Default** Not Applicable

**Examples** [See example](#)

**C++ Syntax** HRESULT PutErrorTermComplexByString( BSTR bufferName, long InumPoints, float\* real, float\* imag);

**Interface** ICalData3

## PutScalar Method

---

**Description** Puts Scalar data in the Measurement Result buffer. The putScalar array is not processed by the analyzer; it is just displayed. Any change to the measurement state (changing the format, for example) will cause the putScalar data to be overwritten with the data processed from the raw data buffer.

**VB Syntax** *measData.putScalar, format, numPts, data*

**Variable** [\(Type\) - Description](#)

*measData* An IArrayTransfer interface which supports the Measurement object.

*format* **(enum NADDataFormat)** Format of the data. Choose from:

- 0 - naDataFormat\_LinMag
- 1 - naDataFormat\_LogMag
- 2 - naDataFormat\_Phase
- 3 - naDataFormat\_Polar
- 4 - naDataFormat\_Smith
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

[Learn more about Data Format.](#)

**Note: Smith, InverseSmith, and Polar** formats are not allowed.

*numPts* **(integer)** - Number of values. Usually the number of points in the trace (chan.NumberOfPoints).

*data* **(single)** - A one-dimensional array of Scalar data matching the number of points in the current measurement.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim measData As IArrayTransfer  
Set measData = app.ActiveMeasurement
```

```
measData.putScalar naDataFormat_LogMag, 201, dScalar(0)
```

**C++ Syntax**

```
HRESULT putScalar/(tagDataFormat eFormat, long INumValues, float* pArrayOfScalar)
```

**Interface**

```
IArrayTransfer
```

---

Last Modified:

1-Oct-2007    Added temperature formats



## PutNAComplex Method

---

**Description** Puts complex data into the specified location. This method forces the channel into Hold mode to prevent the input data from being overwritten. The data is processed and displayed.

Data put in the naRawData store will be **re-processed** whenever a change is made to the measurement attributes such as format or correction.

Data put in the naMeasResult store will be **overwritten** by any measurement attribute changes (such as moving a marker).

**Note:** This method uses NAComplex which is a user-defined data type. If you cannot or prefer not to use this data type, use the [putComplex](#) method.

**VB Syntax** *measData.putNAComplex location, numPts, data, [format]*

**Variable** [\(Type\)](#) - **Description**

*measData* An IArrayTransfer interface which supports the Measurement object

*location* **(enum NADataStore)** Where the Data will be put. Choose from:

0 - naRawData

1 - naCorrectedData

2 - naMeasResult

3 - naRawMemory

4 - naMemoryResult

5 - naDivisor - When reading data from, or writing data to, the normalization divisor, you must first create a divisor trace using [DataToDivisor Method](#).

*numPts* **(long integer)** - Number of data points in the channel

*data* **(NAComplex)** - A one-dimensional array of Complex data matching the number of points in the current measurement.

*format* **(enum NADataFormat)** - Optional argument. Format of the data. If unspecified, naDataFormat\_Polar is assumed. Only used when the destination store is naMeasResult or naMemoryResult.

0 - naDataFormat\_LinMag

1 - naDataFormat\_LogMag

2 - naDataFormat\_Phase

3 - naDataFormat\_Polar

4 - naDataFormat\_Smith

5 - naDataFormat\_Delay

6 - naDataFormat\_Real

- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

[Learn more about Data Format.](#)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim measData As IArrayTransfer
Set measData = app.ActiveMeasurement

measData.putNAComplex naMemoryResult, 201, dRawComplex(0)
```

**C++ Syntax** HRESULT putNAComplex(tagNADataStore DataStore, long INumValues, TsComplex\* pArrayOfComplex, tagDataFormat displayFormat)

**Interface** IArrayTransfer

---

Last Modified:

1-Oct-2007    Added temperature formats

## put\_Output Method

---

**Description** Writes a TTL HI or TTL Low to output pins 3 or 4 of the Material Handler IO connector. Each pin also has a latched output which is written to with USER. With the latched (USER) outputs, the value is not applied to the associated pin until a positive edge is detected at INPUT1 (pin 2).

**VB Syntax** *handlerIo.put\_Output (pin) = value*

**Variable** [\(Type\) - Description](#)

*handlerIo* **(object)** - A HandlerIO object

*pin* **(enum as NAMatHandlerOutput)** - pin to write data to. Choose from:

**naOutput1 - (0)** - pin3

**naOutput1User (1)** - pin3 latched (applied to pin 3 on positive edge of Input1-pin2)

**naOutput2 (2)** - pin4

**naOutput2User (3)** - pin4 latched (applied to pin 4 on positive edge of Input1-pin2)

*value* **(Variant)** Value to write to the selected pin. Choose from

**0** - TTL LOW

**1** - TTL HIGH

**Return Type** Not Applicable

**Default** 0

**Examples** `handlerIo.put_Output (naOutput1) = 1`

**C++ Syntax** HRESULT put\_Output ( tagNAMatHandlerOutput Output, VARIANT Data );

**Interface** IHWMaterialHandlerIO

## put\_OutputVoltage Method

---

**Description** **E836x and PNA-L:** Sets voltages on the DAC/Analog Output 1|2 of the Auxiliary IO connector.

**PNA-X:** Sets voltage on the [Power I/O connector](#) AnalogOut1|2.

Read output voltages using [get OutputVoltage Method](#).

**VB Syntax** *AuxIO.put\_OutputVoltage output, voltage*

**Variable** [\(Type\)](#) - Description

*AuxIO* **(object)** - A Hardware Auxiliary Input / Output object

*output* **(variant)** Number of the output DAC to write voltage to. Choose from:

**1** Output 1 (Aux I/O pin 3) and (Power I/O pin 3)

**2** Output 2 (Aux I/O pin 2) and (Power I/O pin 4)

*voltage* **(double)** Voltage to write to the output DAC. Choose a voltage from -10 to 10

**Return Type** None

**Default** None

**Examples** `HWAuxIO.put_OutputVoltage 1,9 'set Analog Out1 to +9v`

**C++ Syntax** HRESULT put\_OutputVoltage (VARIANT Output, double Voltage );

**Interface** IHWAuxIO

---

Last Modified:

10-Jul-2007    Added PNA-X capability

## put\_OutputVoltageMode Method

---

**Description** This command sets the mode of the selected "Analog Out" line on the [Auxiliary IO connector](#) and [Power I/O connector](#). The modes give the user the option to have the requested voltage applied immediately or not until the sweep is done. To read the mode on each output use [get\\_OutputVoltageMode Method](#).

**VB Syntax** *auxIo.put\_OutputVoltageMode (output, mode)*

**Variable** [\(Type\)](#) - Description

*auxIo* **(Object)** An AuxIO object

*output* Analog Output to receive mode setting. Choose from **1** or **2**

*mode* **(enum NAOOutputVoltageMode)**

**naWaitEOS** - While in this mode any voltage changes sent to the selected analog out will only get applied to the output between sweeps.

**naNoWait** - While in this mode any voltage changes sent to the selected analog out will occur right away without waiting until the end of a sweep, the voltage gets applied immediately.

**Return Type** NAOOutputVoltageMode

**Default** naWaitEOS

**Examples** `auxIo.put_OutputVoltageMode 1, naWaitEOS 'Write`

**C++ Syntax** HRESULT put\_OutputVoltageMode(VARIANT Output, tagNAOutputVoltageMode dNewMode);

**Interface** IHWAuxIO

---

Last Modified:

10-Jul-2007    Added PNA-X capability

## put\_Port Method

**Description** Writes a value to the specified port. Use the [get\\_Port](#) Method to read the settings from the "readable" ports (C, D, E).

**VB Syntax** *handlerIo.put\_Port (port, value)*

**Variable** [\(Type\) - Description](#)

*handlerIo* **(object)** - A HandlerIO object

*port* **(enum as NAmatHandlerPort)** - port to put data into. Choose from:

**naPortA** - (0)

**naPortB** - (1)

**naPortC** - (2)

**naPortD** - (3)

**naPortE** - (4)

**naPortF** - (5)

**naPortG** - (6)

**naPortH** - (7)

*value* The number of the data bits to set. The following table shows what the *value* represents:

**Note:** When writing to port G, port C must be set to output mode

When writing to port H, both port C and port D must be set to output mode. Use Port Mode Property

Port	Max allowable <num>	MSB.....LSB 23.....0	
A	255	A7...A0	Write-only
B	255	B7...B0	Write-only
C	15	C3...C0	Read-Write
D	15	D3...D0	Read-Write
E	255	D3...D0 + C3...C0	Read-Write
F	65535	B7...B0 + A7...A0	Write-only

G	1048575	C3...C0 + B7...B0 + A7...A0	Write-only
H	16777215	D3...D0 + C3...C0 + B7...B0 + A7...A0	Write-only

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `handlerIo.put_Port (naPortB, 1)`

**C++ Syntax** `HRESULT put_Port ( tagNAMatHandlerPort Port, VARIANT Data );`

**Interface** IHWMaterialHandlerIO

---

Last Modified:

21-Mar-2008 Fixed syntax

## put\_PortCData Method

---

**Description** Writes a 4-bit value to Port C on the Aux I/O connector (pins 22-25) and the Material Handler IO (pins 21-24 Anritsu) - (pins 22-25 Avantest).

**Note:** These lines are connected to both the Handler IO and Aux IO in the PNA. Therefore, this command will affect both of these connectors in the same way.

**VB Syntax** *AuxIO.put\_PortCData num*

**Variable** [\(Type\)](#) - Description

*AuxIO* **(object)** - A Hardware Auxiliary Input / Output object

*num* **(variant)** - 4 bit binary value. Choose from 0-15

**Return Type** None

**Default** None

**Examples** `HWAuxIO.put_PortCData 15 'If Positive Logic, Port C lines C0, C1, C2, C3 go High. If Negative Logic, they go Low.`

**C++ Syntax** HRESULT put\_PortCData( VARIANT Data );

**Interface** IHWAuxIO



## PutDataScalar Method

---

**Description** Puts formatted variant scalar data into the measurement result buffer. The data will be immediately processed and displayed. Subsequent changes to the measurement state will be reflected on the display.

Always precede this command by setting the format on the measurement to be consistent with the format of the data being sent to the analyzer. In this way, the display annotation will be correct.

Execution of this command does not change the display format.

**VB Syntax** *meas.putDataScalar format, data*

**Variable** [\(Type\) - Description](#)

*meas* A measurement (**object**)

*format* (**enum NADDataFormat**) Format of the data. **This value is presently ignored by the PNA.** Data is always presented in the current format.

Choose from:

- 0 - naDataFormat\_LinMag
- 1 - naDataFormat\_LogMag
- 2 - naDataFormat\_Phase
- 3 - naDataFormat\_Polar
- 4 - naDataFormat\_Smith
- 5 - naDataFormat\_Delay
- 6 - naDataFormat\_Real
- 7 - naDataFormat\_Imaginary
- 8 - naDataFormat\_SWR
- 9 - naDataFormat\_PhaseUnwrapped
- 10 - naDataFormat\_InverseSmith
- 11 - naDataFormat\_Kelvin
- 12 - naDataFormat\_Fahrenheit
- 13 - naDataFormat\_Centigrade

### Notes:

- The [getData](#) (variant) method includes a "format" argument, which allows scalar (one-dimensional) data. To put data back into the "raw" data buffer using this (putDataComplex) method, specify **Polar** format when using the getData method.
- **Phase** format accepts data in radians (not degrees) and displays in degrees. To convert to degrees:  $\text{radians} * (57.29577951308233) = \text{degrees}$ . The getData method returns degrees if the request is for phase data.

*data* **(variant)** - A 1-dimension array of single precision floating point numbers.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
' Put 201 points worth of scalar data into the measurement
' 200 is max index, so 0 to 200 is 201 points
Dim data(200) ' array of 201 (scalar) data points
' Fill the array
For i = 0 to 200
    data(i) = i/200;
Next
app.ActiveMeasurement.putDataScalar 0, data
```

**C++ Syntax** HRESULT putDataScalar(tagNADataStore DataStore, VARIANT scalarArray)

**Interface** IMeasurement

---

Last Modified:

1-Oct-2007    Added temperature formats

## PutShortcut Method

---

**Description** Defines a Macro (shortcut) file in the analyzer. This command links a file name and path to the Macro file. The file must be put in the PNA at the location indicated by this command.

**VB Syntax** *app.PutShortcut index,title,path, arguments*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*index* **(long)** - Number of the macro to be stored in the analyzer. If the index number already exists, the existing macro is replaced with the new macro.

*title* **(string)** - The name to be assigned to the macro

*path* **(string)** - Full path, file name, and extension of the existing macro "executable" file.

*arguments* **(string)** - Arguments that may be required for the specified macro to run.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.PutShortcut 1, "Test", "C:/Automation/MyTest.vbs", ""`

**C++ Syntax** HRESULT PutShortcut(long Number, BSTR title, BSTR pathname,BSTR arguments)

**Interface** IApplication

---

Last Modified:

25-Jun-2009    Added arguments

## putSourcePowerCalDataEx Method

<b>Description</b>	<p><b>Note:</b> This method replaces <a href="#">putSourcePowerCalData Method</a></p> <p>Inputs source power calibration data (as variant data type) to this channel for a specific source port.</p> <p>If the channel is sweeping the source backwards, then the first data point is the highest frequency value; the last data point is the lowest. Use the <a href="#">Get X-Axis Values</a> command to return the X-axis values in the displayed order.</p> <p>The calibration is not valid if the current number of points on the channel is not equal to the number of values that were input.</p> <p><b>Note:</b> This method sends variant data which is less efficient than methods available on the ISourcePowerCalData interface.</p>
<b>VB Syntax</b>	<i>chan.putSourcePowerCalDataEx buffer, srcPort, data</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>chan</i>	<b>(object)</b> – A Channel object
<i>buffer</i>	<b>(enum NASourcePowerCalBuffer)</b> - The source power cal data buffer to write to. <b>0 - naCorrectionValues</b> This is the only data buffer currently available.
<i>srcPort</i>	<b>(long integer)</b> – The source port for which calibration data is being requested. <b>Note:</b> If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use <a href="#">chan.getPortNumber</a> to translate the string into a port number. To learn more see <a href="#">Remotely Specifying a Source Port</a> .
<i>data</i>	<b>(variant)</b> – Array of source power cal data being input.
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>chan.putSourcePowerCalDataEx naCorrectionValues, 1, varData</code>
<b>C++ Syntax</b>	HRESULT putSourcePowerCalDataEx(tagNASourcePowerCalBuffer bufSelect, long sourcePort, VARIANT varData);
<b>Interface</b>	IChannel4

Last Modified:

24-Apr-2008    Added string names note

27-Jun-2007    Updated for PNA-X source port names

## putSourcePowerCalDataScalarEx Method

<b>Description</b>	<p><b>Note:</b> This method replaces <a href="#">putSourcePowerCalDataScalar Method</a></p> <p>Inputs source power calibration data (as scalar values) to this channel for a specific source port.</p> <p>If the channel is sweeping the source backwards, then the first data point is the highest frequency value; the last data point is the lowest. Use the <a href="#">Get X-Axis Values2</a> command to return the X-axis values in the displayed order.</p>
<b>VB Syntax</b>	<i>chanData.putSourcePowerCalDataScalarEx buffer, srcPort, numValues, data</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>chanData</i>	<b>(interface)</b> – An <a href="#">ISourcePowerCalData2</a> interface on the Channel (object)
<i>buffer</i>	<b>(enum NASourcePowerCalBuffer)</b> - The source power cal data buffer to write to. <b>0 - naCorrectionValues</b> This is the only buffer currently available.
<i>srcPort</i>	<b>(long integer)</b> – The source port for which calibration data is being input. <b>Note:</b> If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use <a href="#">chan.getPortNumber</a> to translate the string into a port number. To learn more see <a href="#">Remotely Specifying a Source Port</a> .
<i>numValues</i>	<b>(long integer)</b> – Number of data values being input. <b>Note:</b> If this does not equal the current number of points on the channel, the calibration will not be valid.
<i>data</i>	<b>(single)</b> – Array of source power cal data being input.
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim chanData As ISourcePowerCalData2 Set chanData = app.ActiveChannel chanData.putSourcePowerCalDataScalarEx naCorrectionValues, 1, 201, scalarCalValues(0)</pre>
<b>C++ Syntax</b>	HRESULT putSourcePowerCalDataScalarEx(tagNASourcePowerCalBuffer bufSelect, long sourcePort, long numValues, float *pData);
<b>Interface</b>	ISourcePowerCalData2

Last Modified:

24-Apr-2008    Added string name note

**PutStandard Method** **Superseded**

---

**Description** This command is [replaced](#) with [PutStandardByString](#)

Puts standard acquisition data into the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data.

**VB Syntax** *CalSet.putStandard class, rcv, src, data*

**Variable** [\(Type\)](#) - Description

*CalSet* **(object)** - A [Cal Set](#) object

*class* **(enum NACalClass)** Standard. Choose from:

- 1 - naClassA
- 2 - naClassB
- 3 - naClassC
- 4 - naClassD
- 5 - naClassE
- 6 - naReferenceRatioLine
- 7 - naReferenceRatioThru

**SOLT Standards**

- 1 - naSOLT\_Open
- 2 - naSOLT\_Short
- 3 - naSOLT\_Load
- 4 - naSOLT\_Thru
- 5 - naSOLT\_Isolation

**TRL Standards**

- 1 - naTRL\_Reflection
- 2 - naTRL\_Line\_Reflection
- 3 - naTRL\_Line\_Tracking
- 4 - naTRL\_Thru
- 5 - naTRL\_Isolation



*rcv*    **(long)** - Receiver Port

*src*    **(long)** - Source Port

*data*    **(variant)** Error term data in a two-dimensional array (0:1, 0:numpts-1). The data must be complex pairs.

**Note:** See also [Put Standard Complex](#) on the ICalData2 interface to avoid using the variant data type.

**Return Type**    Not Applicable

**Default**    Not Applicable

**Examples**    [See an Example](#)

**C++ Syntax**    HRESULT putStandard(tagNACalClass stdclass, long ReceivePort, long SourcePort, VARIANT varData)

**Interface**    ICalSet

## PutStandardByString

---

**Description** Puts standard acquisition data into the Cal Set.  
Learn more about [Reading and Writing Cal Data](#)  
See examples of [Reading](#) and [Writing](#) Cal Set Data.

**VB Syntax** **PutStandardByString**(*stdName*, *vdata*)

**Variable** [\(Type\)](#) - **Description**

*stdName* **(String)** The string used to identify a particular standard in the Cal Set. An example string requesting the data for the Load standard in a full 2 port cal might be "S11C(3,3)".

*vdata* **(Variant)** The variant containing a safearray of variants. This data is usually two dimensional.

**Note:** The vardata array is a safearray of variants wrapped in a variant. This structure is compatible with scripting clients who can only use variants. For alternative methods that used typed arrays, see ICalData3.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See an Example](#)

**C++ Syntax** HRESULT PutStandardByString( BSTR bufferName, VARIANT vardata);

**Interface** ICalSet2

**PutStandardComplex Method** **Superseded**

**Description** This command is replaced with [PutStandardComplexByString](#)

Puts standards acquisition data into the Cal Set.

Learn more about [Reading and Writing Cal Data](#)

See examples of [Reading](#) and [Writing](#) Cal Set Data

**VB Syntax** *ICalData2.putStandardComplex class, rcv, src, numPts, real(), imag()*

**Variable** [\(Type\)](#) - Description

*ICalData2* An [ICalData2](#) pointer to the Cal Set object

*class* **(enum NACalClass)** Standard. Choose from:

1 - naClassA

2 - naClassB

3 - naClassC

4 - naClassD

5 - naClassE

6 - naReferenceRatioLine

7 - naReferenceRatioThru

**SOLT Standards**

1 - naSOLT\_Open

2 - naSOLT\_Short

3 - naSOLT\_Load

4 - naSOLT\_Thru

5 - naSOLT\_Isolation

**TRL Standards**

1 - naTRL\_Reflection

2 - naTRL\_Line\_Reflection

3 - naTRL\_Line\_Tracking

4 - naTRL\_Thru

5 - naTRL\_Isolation

<i>rcv</i>	<b>(long)</b> - Receiver Port
<i>src</i>	<b>(long)</b> - Source Port
<i>numPts</i>	<b>(long)</b> - The number of data points in the real and imaginary arrays.
<i>real()</i>	<b>(single)</b> - one-dimensional array containing the <b>real</b> part of the acquisition data. (0:points-1)
<i>imag()</i>	<b>(single)</b> - one-dimensional array containing the <b>imaginary</b> part of the acquisition data. (0:points-1)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Dim sdata As ICalData2 Set sdata = calmanager.CreateCal Set( 1 ) sdata.putStandardComplex naSOLT_Open, 1, 1, numpts, rel(0), img(0)</pre>
<b><u>C++ Syntax</u></b>	HRESULT putStandardComplex(tagNACalClass stdclass, long ReceivePort, long SourcePort, long INumValues, float* pReal, float* plmag)
<b>Interface</b>	ICalData2

## PutStandardComplexByString

---

**Description** Puts standard acquisition data into the Cal Set.  
Learn more about [Reading and Writing Cal Data](#)  
See examples of [Reading](#) and [Writing](#) Cal Set Data.

**VB Syntax** *ICalData3.PutStandardComplexByString(stdName, InumPoints , real(o) , imag(0))*

**Variable** [\(Type\)](#) - Description

*ICalData3* An [ICalData3](#) pointer to a Cal Set object.

*stdName* **(String)** The string used to identify a particular standard in the Cal Set. An example string requesting the data for the Load standard in a full 2 port cal might be "S11C(3,3)".

*Inumpoints* **(long)** - The number of data points in the real and imaginary arrays.

*real* **(Single)** The real component of the complex data.

*imag* **(Single)** The imaginary component of the complex data.

**Return Value** Single

**Default** Not Applicable

**Examples** [See an Example](#)

**C++ Syntax** HRESULT PutStandardComplexByString( BSTR bufferName, long InumPoints, float\* real, float\* imag);

**Interface** ICalData3

## Quit Method

---

**Description** Terminates the Network Analyzer application.

**VB Syntax** *app*.Quit

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.Quit`

**[C++ Syntax](#)** HRESULT Quit()

**Interface** IApplication

**Remarks** Under the rules of COM, the server should not exit until all references to it have been released. This method is a brute force way of terminating the application. Be sure to release all references (or terminate the client program) before attempting to restart the Network Analyzer application.

An alternate approach to terminating the application is to make the application invisible (`app.Visible = False`) and release all references. The server will shutdown.

## ReadData Method

---

<b>Description</b>	Reads a 13-bit data word from the specified address. Data is read using the AD0 through AD12 lines of the external test set connector. The instrument generates the appropriate timing signals. It automatically controls timing signals LDS, LAS and RLW to strobe the address, and then read the data, from the external test set. See the <a href="#">timing diagram</a> for Address and Data I/O read.
<b>VB Syntax</b>	<i>value</i> = <b>ExtIO.ReadData</b> ( <i>address</i> )
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>value</i>	<b>(variant)</b> - Variable to store the returned data
<i>ExtIO</i>	<b>(object)</b> - An ExternalTestSetIO object
<i>address</i>	<b>(variant)</b> - address to read data from.
<b>Return Type</b>	Variant
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>value = ExtIO.ReadData (15)</pre>
<b><a href="#">C++ Syntax</a></b>	HRESULT ReadData (VARIANT Address, VARIANT* Data);
<b>Interface</b>	IHWExternaTestSetIO

## ReadRaw Method

---

**Description** Reads a 16-bit value from the external test set. The 16-bit value is comprised of lines AD0 - AD12, Sweep Holdoff In and Interrupt In (inverted).

When this command is used the analyzer does NOT generate the appropriate timing signals; it simply reads the lines. The user needs to first use the [WriteRaw](#) method to do the initial setup. The RLW line (pin25) must be set to the appropriate level in order to read the test set connected.

**Below is the format of data that is read with ReadRaw:**

Pin	Bit	Signal name
22	0	AD0*
23	1	AD1*
11	2	AD2*
10	3	AD3*
9	4	AD4*
21	5	AD5*
20	6	AD6*
19	7	AD7*
6	8	AD8*
5	9	AD9*
4	10	AD10*
17	11	AD11*
3	12	AD12*
2	13	Sweep Holdoff In
13	14	Interrupt In (inverted internally)
na	15	Always Zero, grounded internally



\*These lines are dependent on the state of RLW (pin25).  
Writing a 0(low) to RLW will set lines AD0-AD12 to write mode.  
Writing a 1(high) to RLW will set lines AD0-AD12 to read mode.

**VB Syntax**    *value* = *ExtIO.ReadRaw* (*address*)

**Variable**    [\(Type\)](#) - Description

*value*    **(variant)** - Variable to store the returned data

*ExtIO*    **(object)** - An External IO object

*address*    **(variant)** - Address to read data from

**Return Type**    Real

**Default**    Not Applicable

**Examples**    `value = ExtIO.ReadRaw (address)`

**C++ Syntax**    HRESULT ReadRaw( VARIANT\* Input );

**Interface**    IHWExternalTestSetIO

## Recall Method

---

<b>Description</b>	Recalls a measurement state, calibration state, or both, from the hard drive into the analyzer. Use app. <a href="#">Save</a> to save files.
<b>VB Syntax</b>	<code>app.Recall (filename.ext)</code>
<b>Variable</b>	<b><a href="#">(Type) - Description</a></b>
<code>app</code>	An <a href="#">Application</a> ( <b>object</b> )
<code>filename.ext</code>	<b>(string)</b> - Full path, file name, and extension, of the file. Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents" Use one of the following extensions: <ul style="list-style-type: none"> <li>• .sta - Instrument State</li> <li>• .cal - Calibration file</li> <li>• .cst - Both Instrument State and Calibration reference</li> <li>• .cti - Citifile (data will always be formatted. See <a href="#">Recalling Citifiles Using the PNA</a>)</li> <li>• .csa - Instrument state and calibration data (not a reference pointer).</li> </ul>
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.Recall ("C:/Program Files/Agilent/Network Analyzer/Documents/MyState.cst")</code> 'Recalls "mystate.cst" from the specified folder
<b><a href="#">C++ Syntax</a></b>	<code>HRESULT Recall(BSTR bstrFile)</code>
<b>Interface</b>	IApplication

## Recall Kits Method

---

**Description**    Recalls the calibration kits definitions that were stored with the SaveKits command.

**VB Syntax**    *app*.**RecallKits**

**Variable**    [\(Type\)](#) - Description

*app*    An [Application](#) (**object**)

**Return Type**    Not Applicable

**Default**    Not Applicable

**Examples**    `app.RecallKits`

**C++ Syntax**    HRESULT RecallKits()

**Interface**    IApplication

## Remove Method

---

**Description** Removes an item from a collection of objects.

**VB Syntax** *Object*.**Remove** *item*

**Variable** [\(Type\)](#) - **Description**

*Object* Any of the following (**objects**)

- [CalFactorSegments collection](#)
- [CalFactorSegmentsPMAR Collection](#)
- [Cal Sets collection](#)
- [ExternalDevices Collection](#)
- [Measurements collection](#)
- [NAWindows collection](#)
- [PowerLossSegments collection](#)
- [PowerLossSegmentsPMAR Collection](#)
- [Segments collection](#)

**Note: Segments, CalFactorSegments, and PowerLossSegments** have an OPTIONAL argument [size] referring to the number of segments to remove, starting with the *item* parameter.

**Note: Segments** - When ALL segments are deleted, [SweepType](#) is automatically set to Linear because there are no segments to sweep.

*item* (**variant**) - Item number to be removed.

Note: The ExternalDevices Collection requires that you specify *item* as the string name of the device. For example:

```
extDevices.Remove ('mySource')
```

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Measurements.Remove 3 'Removes measurement 3
segments.Remove 2,20 'Removes 20 segments (2 - 21)
```

**C++ Syntax** HRESULT Remove(VARIANT index); //Measurements  
 HRESULT Remove(VARIANT index); //Cal Sets

HRESULT Remove(long windowNumber); //NAWindows  
HRESULT Remove(VARIANT index, long size); //Segments  
HRESULT Remove(VARIANT index, long size); //CalFactorSegments(PMAR)  
HRESULT Remove(VARIANT index, long size); //PowerLossSegments(PMAR)  
HRESULT Remove(BSTR name) //ExternalDevices

**Interface** All listed above

---

Last Modified:

31-Jul-2009 Added External Devices (9.0)

## Reset Method

---

<b>Description</b>	Removes all existing windows and measurements from the application. (Unlike <a href="#">Preset</a> , does not create a new measurement.)
<b>VB Syntax</b>	<i>app</i> .Reset
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>app</i>	An <a href="#">Application</a> (object)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.Reset</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT Reset()
<b>Interface</b>	IApplication

## ResetLOFrequency Method

---

**Description** Resets the LO Delta Frequency to 0 (zero) Hz.

**VB Syntax** *obj*.ResetLOFrequency

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (**object**) or  
A [ConverterEmbeddedLO](#) (**object**)

**Default** Not Applicable

**Examples** `embedLO.ResetLOFrequency 'write`

**C++ Syntax** HRESULT ResetLOFrequency();

**Interface** IEmbeddedLO

---

### Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic

## ResetTuningParameters Method

---

**Description** Resets the tuning parameters to their default values.

**VB Syntax** *obj*.ResetTuningParameters

**Variable** [\(Type\)](#) - Description

*obj* An [EmbeddedLO](#) (**object**) or  
A [ConverterEmbeddedLO](#) (**object**)

**Default** Not Applicable

**Examples** `embedLO.ResetTuningParameters 'write`

**C++ Syntax** HRESULT ResetTuningParameters();

**Interface** IEmbeddedLO

---

### Last Modified:

12-Aug-2009 Added ConvEmbedLO object

18-Apr-2007 MX New topic



## RestoreCalKitDefaults Method

---

**Description** Restores the original properties of the specified Cal Kit, overwriting the last definition with the factory defaults.

**NOTE:** ONLY works with PNA releases 1.0 through 1.6.

**VB Syntax** `app.RestoreCalKitDefaults (calKit)`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*calKit* (**enum NACalKit**) - Calibration Kit to restore. Choose from:

1 - naCalKit\_85032F\_N50  
2 - naCalKit\_85033E\_3\_5  
3 - naCalKit\_85032B\_N50  
4 - naCalKit\_85033D\_3\_5  
5 - naCalKit\_85038A\_7\_16  
6 - naCalKit\_85052C\_3\_5\_TRL  
7 - naCalKit\_User7  
8 - naCalKit\_User8  
9 - naCalKit\_User9  
10 - naCalKit\_User10

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.RestoreCalKitDefaults naCalKit_MechKit10`

**C++ Syntax** HRESULT RestoreCalKitDefaults/(tagNACalKit kit)

**Interface** IApplication

## RestoreCalKitDefaultsAll Method

---

**Description** Restores the original properties of ALL of the Cal Kits, overwriting the last definitions with the factory defaults.

**NOTE:** ONLY works with PNA releases 1.0 through 1.6.

**VB Syntax** `app.RestoreCalKitDefaultsAll`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.RestoreCalKitDefaultsAll`

**C++ Syntax** HRESULT RestoreCalKitDefaultsAll()

**Interface** IApplication

## RestoreDefaults Method

---

**Description** Resets the PNA preferences to their factory default settings.

**VB Syntax** *pref*.**RestoreDefaults**

**Variable** [\(Type\)](#) - **Description**

*pref* A [Preferences](#) (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `pref.RestoreDefaults`

**C++ Syntax** HRESULT RestoreDefaults()

**Interface** IPreferences9

---

Last Modified:

24-Apr-2008    MX New topic

## Resume Method

---

<b>Description</b>	Resumes the trigger mode of all channels that was in effect before sending the <a href="#">channels.Hold</a> method. Channels.Hold must be sent before channels.Resume, using the same instance of the Channels object.
<b>VB Syntax</b>	<i>chans</i> .Resume
<b>Variable</b>	<a href="#">(Type)</a> - Description
<i>chans</i>	A <a href="#">Channel collection</a> (object)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>chans . Resume</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT Resume();
<b>Interface</b>	Channels2

## Save Method

---

**Description** Saves the appropriate content to the hard drive depending on the extension that is provided.

Some saved files can be recalled using app.[Recall](#). depending on the content.

**VB Syntax** `app.Save(filename.ext)`

**Variable** [\(Type\)](#) - Description

<i>app</i>	An <a href="#">Application</a> (object)
<i>filename.ext</i>	<p>(string) - Full path, file name, and extension of the file.</p> <p>Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents"</p> <p>Use one of the following extensions:</p> <ul style="list-style-type: none"> <li>• <b>.cst</b> - Saves both Instrument State and Cal Set reference - Recalls a calibrated measurement. (Recallable)</li> <li>• <b>.sta</b> - Saves Instrument State only - recalls the instrument state without calibration. (Recallable)</li> <li>• <b>.cal</b> - Calibration file – saves the active Cal Sets currently in use by any channel. Use this mode for archival purposes only. All Cal Sets are saved to a Cal Set data file. This mode provides a method of safeguarding calibration data. This data can be restored to the list of Cal Sets available in the instrument. (Recallable)</li> <li>• <b>.csa</b> - Saves both instrument state AND actual calibration data, not a reference pointer to the Cal Set.</li> <li>• <b>.prn</b> - Saves active trace in comma-separated format (not recallable)</li> <li>• <b>.bmp</b> - Saves a Bitmap of the screen (not recallable)</li> <li>• <b>.s1p</b> - Saves 1-port measurement data (not recallable)</li> <li>• <b>.s2p</b> - Saves 2-port measurement data (not recallable)</li> <li>• <b>.s3p</b> - Saves 3-port measurement data (not recallable)</li> <li>• <b>.s4p</b> - Saves 4-port measurement data (not recallable)</li> </ul>

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
app.Save("C:/Program Files/Agilent/Network
Analyzer/Documents/Newfolder/MyState.cst") 'Saves "mystate.cst"
to the specified folder
```

[C++ Syntax](#) HRESULT Save(BSTR bstrFile)

[Interface](#) IApplication

---

Last Modified:

26-Jun-2007    Corrected example

## Save Method

---

**Description** Saves the current Cal Set to disk. This is the recommended method for saving a Cal Set.

Learn more about [reading and writing Cal data using COM](#)

**VB Syntax** *CalSet*.**Save**

**Variable** [\(Type\)](#) - Description

*CalSet* **(object)** - A [CalSet](#) object

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `myCalSet.Save`

See [Copy Method](#) for an example application of this command.

**C++ Syntax** HRESULT Save();

**Interface** ICalSet

## SaveCalSets Method **Superseded**

---

**Description** This command is replaced by [ICalSet::Save](#) which saves the data for **only** the current Cal Set to the disk.

Writes new or changed Cal Sets to disk. All Cal Sets are saved in a single file. This file is updated at the following times:

- When a Cal Set has been deleted.
- When a calibration has been performed through the front panel interface.
- When this method is called.
- When [ICalSet::Save](#) is called.

Learn more about [reading and writing Cal data using COM](#)

**VB Syntax** *object*.SaveCalSets

**Variable** [\(Type\)](#) - Description

*object* **(object)** - A CalManager object or a Calibrator object

**Return Type** None

**Default** Not Applicable

**Example** `calMgr.SaveCalSets`

**C++ Syntax** HRESULT SaveCalSets( );

**Interface** ICalManager  
ICalibrator



## SaveCitiDataData Method - Superseded

---

<b>Description</b>	This command is replaced with <a href="#">SaveData Method</a> Saves UNFORMATTED trace data to .cti file. <a href="#">Learn more about citifiles.</a>
<b>VB Syntax</b>	<code>app.SaveCitiDataData(filename.cti)</code>
<b>Variable</b>	<b>(Type) - Description</b>
<i>app</i>	An <a href="#">Application</a> (object)
<i>filename.cti</i>	<b>(string)</b> - Full path, file name, and .cti extension of the file. Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents".
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>app.SaveCitiDataData ("C:/Program Files/Agilent/Network Analyzer/Documents/myDDCitifile.cti") 'Saves "myDDCitifile.cti" to the specified folder</pre>
<b>C++ Syntax</b>	HRESULT SaveCitiDataData (BSTR bstrFile)
<b>Interface</b>	IApplication5

## SaveCitiFormattedData Method - Superseded

---

<b>Description</b>	This command is replaced with <a href="#">SaveData Method</a> Saves FORMATTED trace data to .cti file. <a href="#">Learn more about citifiles.</a>
<b>VB Syntax</b>	<code>app.SaveCitiFormattedData(filename.cti)</code>
<b>Variable</b>	<b>(Type) - Description</b>
<i>app</i>	An <a href="#">Application</a> (object)
<i>filename.cti</i>	<b>(string)</b> - Full path, file name, and .cti extension of the file. Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents"
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.SaveCitiFormattedData ("C:/Program Files/Agilent/Network Analyzer/Documents/Newfolder/myFDCitifile.cti") 'Saves "myFDCitifile.cti" to the specified folder</code>
<b>C++ Syntax</b>	HRESULT SaveCitiFormattedData (BSTR bstrFile)
<b>Interface</b>	IApplication5

## SaveENRFile Method

---

**Description** Saves an ENR table to disk.

**VB Syntax** *enr*.**SaveENRFile** (*filename*)

**Variable**    [\(Type\)](#) - Description

*enr*    An [ENRFile](#) (**object**)

*filename*    (String) - Absolute path and filename of the ENR file.

**Return Type**    Not Applicable

**Default**    Not Applicable

**Examples**    [See example program](#)

**C++ Syntax**    HRESULT SaveENRFile(BSTR filename);

**Interface**    IENRFile

---

Last Modified:

2-Aug-2007    MX New topic

Write only

## SaveFile Method

---

**Description** Saves the mixer/converter test setup to a mixer attributes (.mxr) file.

**VB Syntax** *obj*.**SaveFile** (*filename*)

**Variable** [\(Type\)](#) - **Description**

*obj* A [Mixer Interface](#) pointer to the [Measurement](#) (object)

Or

A [Converter Object](#)

*filename* (String) Full path, file name, and .mxr extension of the file.

Files are typically stored in "C:/Program Files/Agilent/Network Analyzer/Documents".

**Return Type** String

**Default** Not Applicable

**Examples** `mixer.SaveFile ("C:/Program Files/Agilent/Network Analyzer/Documents/myMixer.mxr")`

**C++ Syntax** HRESULT SaveFile(BSTR newVal)

**Interface** IMixer  
IConverter

---

Last Modified:

2-Feb-2009    Added Converter

## SaveToECal Method

---

**Description** Saves the User Characterization to the ECal module. This can take several minutes to complete. To save to PNA disk memory, use [SaveToDiskMemory Method](#). User Characterization can be saved to both PNA disk memory and ECal module memory.

**VB Syntax** *userChar*.**SaveToECal**

**Variable** [\(Type\)](#) - **Description**

*userChar* An [ECalUserCharacterizer](#) (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `userChar . SaveToECal`

**C++ Syntax** HRESULT SaveToECal();

**Interface** IECalUserCharacterizer

---

Last Modified:

2-Nov-2008    New topic (8.33)

## SaveKits Method

---

**Description** Saves the cal kits, typically after modifying a calibration kit. To load a cal kit into the analyzer from the hard drive, use [app.RecallKits](#).

**VB Syntax** *app*.SaveKits

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.SaveKits`

**[C++ Syntax](#)** HRESULT SaveKits()

**Interface** IApplication

## SearchCompressionPoint Method

---

**Description** Searches the markers domain for the specified [compression level](#).

**VB Syntax** *mkr*.SearchCompressionPoint

**Variable** [\(Type\)](#) - Description

*mkr* A [Marker](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mkr.SearchCompressionPoint 'Read`

[See example program](#)

**C++ Syntax** HRESULT SearchCompressionPoint()

**Interface** IMarker4

---

Last Modified:

8-Feb-2009 MX New topic

## SearchFilterBandwidth Method

---

<b>Description</b>	<p>Searches the measurement data with the current <a href="#">BandwidthTarget</a> (default is -3). To continually track the filter bandwidth, use <a href="#">BandwidthTracking</a>.</p> <p>This feature uses markers 1-4. If not already, they are activated. To turn off these markers, either turn them off individually or <a href="#">DeleteAllMarkers</a>.</p> <p>The bandwidth statistics are displayed on the analyzer screen. To get the bandwidth statistics, use either <a href="#">GetFilterStatistics</a> or <a href="#">FilterBW</a>, <a href="#">FilterCF</a>, <a href="#">FilterLoss</a>, or <a href="#">FilterQ</a>.</p> <p>The analyzer screen will show either Bandwidth statistics OR Trace statistics; not both.</p> <p>To search a <a href="#">UserRange</a> with the bandwidth search, first activate marker 1 and set the desired UserRange. Then send the SearchFilterBandwidth command. The user range used with bandwidth search only applies to marker 1 searching for the max value. The other markers may fall outside the user range.</p>
<b>VB Syntax</b>	<i>meas</i> . <b>SearchFilterBandwidth</b>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>meas</i>	A Measurement ( <b>object</b> )
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>meas.SearchFilterBandwidth</code>
<b>C++ Syntax</b>	HRESULT SearchFilterBandwidth()
<b>Interface</b>	IMeasurement



## SearchMax Method

---

**Description** Searches the marker domain for the maximum value.

**VB Syntax** *mark*.**SearchMax**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchMax`

**[C++ Syntax](#)** HRESULT SearchMax()

**Interface** IMarker

## SearchMin Method

---

**Description** Searches the marker domain for the minimum value.

**VB Syntax** *mark*.**SearchMin**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchMin`

**C++ Syntax** HRESULT SearchMin()

**Interface** IMarker

## SearchNextPeak Method

---

**Description** Searches the marker's domain for the next peak value.

**VB Syntax** *mark*.SearchNextPeak

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchNextPeak`

**[C++ Syntax](#)** HRESULT SearchNextPeak()

**Interface** IMarker

## SearchPeakLeft Method

---

**Description** Searches the marker's domain for the next [VALID](#) peak to the left of the marker.

**VB Syntax** *mark*.SearchPeakLeft

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchPeakLeft`

**[C++ Syntax](#)** HRESULT SearchPeakLeft()

**Interface** IMarker

## SearchPeakRight Method

---

**Description** Searches the marker's domain for the next [VALID](#) peak to the right of the marker.

**VB Syntax** *mark*.SearchPeakRight

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchPeakRight`

**[C++ Syntax](#)** HRESULT SearchPeakRight()

**Interface** IMarker

## SearchTarget Method

---

**Description** Searches the marker's domain for the target value (specified with [mark.TargetValue](#)). Searches to the right; then at the end of the search domain, begins again at the start of the search domain.

**VB Syntax** *mark*.**SearchTarget**

**Variable** [\(Type\)](#) - **Description**

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchTarget`

**C++ Syntax** HRESULT SearchTarget()

**Interface** IMarker

## SearchTargetLeft Method

---

**Description** Moving to the left of the marker position, searches the marker's domain for the target value (specified with [mark.TargetValue](#)).

**VB Syntax** *mark*.**SearchTargetLeft**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchTargetLeft`

**C++ Syntax** HRESULT SearchTargetLeft()

**Interface** IMarker

## SearchTargetRight Method

---

**Description** Moving to the right of the marker position, searches the marker's domain for the target value (specified with [mark.TargetValue](#)).

**VB Syntax** *mark*.**SearchTargetRight**

**Variable** [\(Type\)](#) - **Description**

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SearchTargetRight`

**C++ Syntax** HRESULT SearchTargetRight()

**Interface** IMarker



## SelectCalSet Method

---

**Description** Selects and applies a Cal Set to the specified channel.

**Note:** Error Correction is not automatically applied as a result of this command being issued. If there is more than one Cal Type in the Cal Set, you must explicitly choose the Cal Type you want to apply. (See [meas.Caltype](#))

**VB Syntax** *channel*.**SelectCalSet** *calSet*, *restore*

**Variable** [\(Type\)](#) - **Description**

*channel* **(object)** - A [Channel](#) object

*calSet* **(string)** - Cal Set to make active. Specify the Cal Set by GUID or Name. Use [EnumerateCalSets](#) to list the available Cal Sets.

*restore* **(boolean)** -

**True (1)** - The stimulus stored with the cal set will be applied to the channel.

**False (0)** - If a conflict is detected between the existing channel settings and the Cal Set stimulus settings, then the following will occur:

If interpolation is ON, then interpolation will be attempted. This may fail if the channel frequency is outside the range of the Cal Set.

If interpolation is OFF, the selection will be abandoned and an error is returned:  
E\_NA\_CAL\_STIMULUS\_VALUES\_EXCEEDED

**Return Type** Not Applicable

**Default** Not Applicable

**Example** `channel.SelectCalSet GUID, 1`

`chan.SelectCalSet "MyCalSet", 0`

**C++ Syntax** HRESULT SelectCalSet (BSTR strCset, bool bRestore);

**Interface** IChannel

---

Last Modified:

29-Nov-2007    Modified to accept name

## SetAllSegments Method

---

**Description** Uploads a segment table to the PNA replacing any existing segment table.

Segments must be ascending in frequency and non-overlapping. If they are not, the segments are 'adjusted' as they are from the User Interface control. The total number of points for all segments cannot exceed the PNA [maximum number of points](#) for a sweep.

**VB Syntax** *Segs.SetAllSegments (segdata)*

**Variable** [\(Type\)](#) - **Description**

*segs* A [Segments](#) (Collection)

*segdata* (Variant) A 2-dimensional array of Segment data:

- dimension 0 is the number of elements in each segment.
- dimension 1 is the number of segments that will be used.

The following is a list of dimension 0 elements for each segment:

**Note:** All elements must be **dimensioned** as either ALL Double or ALL Variant.

- 0 = Segment state (Boolean True or False)
- 1 = Number of Points in this segment (Integer)
- 2 = Start Freq (Double)
- 3 = Stop Freq (Double)
- 4 = IFBW (Double) optional
- 5 = Dwell Time (Double) optional
- 6 + = Power (Double) optional; see table below.

The first four data elements must always be supplied. After those values, data must be supplied for successive optional elements. For example, to set dwell time values, you must also supply IFBW values, because IFBW (#4) precedes dwell time (#5) in the array order.

The [IFBandwidthOption](#), [SweepTimeOption](#), and [SourcePowerOption](#) settings do NOT affect the order in which elements are interpreted.

The number of elements to supply for Power depends on the following two settings:

1. [SourcePowerOption](#) = True allows segments to have independent power levels.
2. [CouplePorts](#) = False allows different power levels for each test port.

CouplePorts	SourcePowerOption	Number of Elements
False	False	Each port has its own channel-wide power setting, which is set using <a href="#">TestPortPower</a> . Provide exactly 7 elements per segment. The last element (power) is ignored.
False	True	Provide 6 elements + total number of ports. The first 7 elements are still interpreted the same.  The remaining elements (in-order) are interpreted as the power levels to set on that segment for Ports 2 through N, where N is the total number of ports currently enabled for the PNA or for a PNA with multiport external test set.
True	False	Provide exactly 7 elements per segment. The last element (power) is ignored.
True	True	Provide exactly 7 elements per segment. The last element (power) is honored.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** [See a VB example using this command](#)  
[See a C++ example using this command](#)

**C++ Syntax** HRESULT SetAllSegments (VARIANT Segments );

**Interface** ISegments2

---

Last Modified:

28-Apr-2009    More edits

15-Oct-2007    Major edits and link to C++ example

## SetBBPorts Method

---

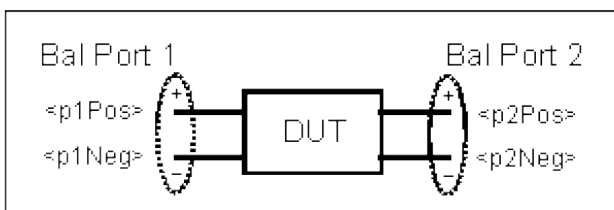
**Description** For a Balanced - Balanced device type, maps the PNA ports to the DUT ports.  
Set the Balanced device type using the [DUTTopology Property](#)

**VB Syntax** *balTopology.SetBBPorts p1Pos, p1Neg, p2Pos, p2Neg*

**Variable** [\(Type\)](#) - Description

*balTopology* A [BalancedTopology](#) (object)

*p1Pos, p1Neg, p2Pos, p2Neg* **(Long Integer)** PNA port number that connects to each of the following DUT ports:



**Return Type** Not applicable - To read port mappings, use the [BalancedTopology](#) properties.

**Default** Not Applicable

**Examples** `balTop.SetBBPorts 1,2,3,4`

**C++ Syntax** HRESULT SetBBPorts (long p1Pos, long p1Neg, long p2Pos, long p2Neg)

**Interface** IBalancedTopology

## SetCalInfo Method

---

**Description** Specifies the type of Unguided calibration. This method should be the first method called on the calibrator object. It prepares the internal state for the rest of the calibration.

**Note:** You can NOT perform a 3 or 4-port cal using SetCalInfo even though there is enumCalTypes. You must use the [GuidedCalibration](#) object.

Learn more about [reading and writing Cal data using COM](#)

The analyzer can measure both ports simultaneously, assuming you have two of each standard type. For a 2-port cal, See [cal.Simultaneous2PortAcquisition](#)

**VB Syntax** *cal.SetCalInfo (type,rcvPort,srcPort)*

**Variable** [\(Type\)](#) - **Description**

*cal* A Calibrator (**object**)

*type* (**enum NACalType**) - Calibration type. Choose from:

- 0 - naCalType\_Response\_Open
- 1 - naCalType\_Response\_Short
- 2 - naCalType\_Response\_Thru
- 3 - naCalType\_Response\_Thru\_And\_Isol
- 4 - naCalType\_OnePort
- 5 - naCalType\_TwoPort\_SOLT
- 6 - naCalType\_TwoPort\_TRL
- 7 - naCalType\_None
- 8 - naCalType\_ThreePort\_SOLT
- 9 - Custom
- 10 - naCalType\_FourPort\_SOLT

**Note:** For 1-port cals, the source port = receiver port. For 2, 3,4-port SOLT and TRL, it doesn't matter which port is specified as source and receiver

*rcvPort* (**long integer**) - Receiver Port

*srcPort* (**long integer**) - Source Port

**Return Type** NACalType

**Default** 7-naCalType\_None

**Examples** `cal.setCalInfo(naCalType_Response_Open, 1, 1)`

**C++ Syntax** HRESULT SetCallInfo(tagNACalType calType, long portA, long portB)

**Interface** ICalibrator

**SetCalInfoEx Method (for source power cal)**

<b>Description</b>	This command replaces <a href="#">SetCalInfo2 Method</a> . Specifies the channel and the source port to be used for the source power calibration about to be performed.
<b>VB Syntax</b>	<code>powerCalibrator.SetCalInfoEx channel, srcPort, [powerOffset,] [display]</code>
<b>Variable</b>	<b>(Type) - Description</b>
<code>powerCalibrator</code>	<b>(object)</b> - A <a href="#">SourcePowerCalibrator</a> object
<code>channel</code>	<b>(long integer)</b> - Number of the PNA channel (not power meter channel) on which the source power cal will be performed. If the channel does not already exist, it will be created.
<code>srcPort</code>	<b>(long integer)</b> - Port number on which the source power cal will be performed. <b>Note:</b> If the port is defined by a string name, such as an external source, a balanced port, or one of the Source 2 outputs on the 2-port 2-source PNA-x model, then you must use <a href="#">chan.getPortNumber</a> to translate the string into a port number. To learn more see <a href="#">Remotely Specifying a Source Port</a> .
<code>[powerOffset]</code>	<b>(double)</b> - Optional argument. Sets or returns a power level offset from the PNA test port power. This can be a gain or loss value (in dB) to account for components you connect between the source and the reference plane of your measurement. For example, specify 10 dB to account for a 10 dB amplifier at the input of your DUT. Following the calibration, the PNA power readouts are adjusted by this value. This argument performs the same function as <a href="#">chan.SourcePowerCalPowerOffset Property</a>
<code>[display]</code>	<b>(boolean)</b> Optional argument. Enables and disables the display of power readings on the PNA screen. After the source power cal data is acquired, this setting is reset to ON. If unspecified, value is set to ON. <b>True</b> - Display of power readings is ON <b>False</b> - Display of power readings is OFF
<b>Return Type</b>	None
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>powerCalibrator.SetCalInfoEx 1, 1, -10, True</code>
<b>C++ Syntax</b>	<code>HRESULT SetCalInfoEx( long Channel, long SourcePort, double PowerOffset = 0., VARIANT_BOOL bDisplay = VARIANT_TRUE);</code>
<b>Interface</b>	ISourcePowerCalibrator4

Last Modified:

24-Apr-2008    Added note for string names

30-Apr-2007    Edited for src strings



## SetCenter Method

---

<b>Description</b>	Changes the center stimulus to the stimulus value of the marker. The start stimulus stays the same and the stop is adjusted.  This command does not work with channels that are in <a href="#">CW</a> or <a href="#">Segment Sweep</a> mode.
<b>VB Syntax</b>	<i>mark</i> . <b>SetCenter</b>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>mark</i>	A Marker ( <b>object</b> )
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>mark.SetCenter</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT SetCenter()
<b>Interface</b>	IMarker

## SetCW Method

---

**Description** Changes the analyzer to sweep type CW mode and sets the CW frequency to the marker's frequency. Does not change anything if current sweep type is other than a frequency sweep.

**VB Syntax** *mark*.**SetCW**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetCW`

**C++ Syntax** HRESULT SetCW()

**Interface** IMarker

## SetCWFreq Method

---

**Description** Sets the CW frequency to the frequency of the active marker. Does NOT change sweep type.

Use ONLY when the current sweep type is sweeping frequency - NOT available in CW or Power Sweep.

Use this command to first set the CW Frequency to a value that is known to be within the current calibrated range, THEN set [Sweep Type](#) to naPowerSweep or naCWTimeSweep.

**VB Syntax** *mark*.SetCWFreq

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetCWFreq`

**C++ Syntax** HRESULT SetCWFreq()

**Interface** IMarker3

---

Last Modified:

3-Oct-2008    MX New topic

## SetDutPorts Method

---

<b>Description</b>	Sets the PNA to DUT port map for FCA measurements. Use <a href="#">DeviceInputPort</a> and <a href="#">DeviceOutputPort</a> to read these values.  Changing the ports may limit your ability to use an internal second source. If a selected port is shared by one of the sources, then that source will not be available as an LO source. <a href="#">Learn more about Internal second sources.</a>
<b>VB Syntax</b>	<i>mixer</i> . <b>SetDUTPorts</b> ( <i>inputPort</i> , <i>outputPort</i> )
<b>Variable</b>	<a href="#">(Type)</a> - <a href="#">Description</a>
<i>mixer</i>	A <a href="#">IMixer</a> Interface pointer to the <a href="#">Meas</a> (object)
<i>inputPort</i>	( <b>Long</b> ) PNA port to be connected to the DUT input.  <ul style="list-style-type: none"> <li>For SMC, choose any unused PNA port.</li> <li>For VMC, set to 1</li> </ul>
<i>outputPort</i>	( <b>Long</b> ) PNA port to be connected to the DUT output. Choose any unused port for SMC and VMC.
<b>Return Type</b>	Not Applicable
<b>Default</b>	1,2
<b>Examples</b>	<code>mixer.SetDUTPorts =2,1</code>
<b>C++ Syntax</b>	HRESULT SetDutPorts(long inputPort,long OutputPort);
<b>Interface</b>	IMixer8

---

Last Modified:

23-Apr-2008    New topic.

## SetElectricalDelay Method

---

**Description** Changes the measurement's electrical delay to the marker's delay value.

**VB Syntax** *mark*.SetElectricalDelay

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetElectricalDelay`

**C++ Syntax** HRESULT SetElectricalDelay()

**Interface** IMarker

## SetFailOnOverRange Method

---

**Description** When set TRUE, configures the analyzer to report outOfRange conditions with an **error** code. Any overrange error will return **E\_NA\_LIMIT\_OUTOFRANGE\_ERROR**.

**Note:** This method is for the benefit of VB clients. The analyzer automatically adjusts overrange conditions to the closest acceptable setting. The VB user will not see that an overrange occurred because the HRESULT is not returned if it has a success code. For more information, see [Events/OverRange](#).

**VB Syntax** *app.SetFailOnOverRange state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* (**boolean**) -  
**True (1)** - Overrange conditions report an error code  
**False (0)** - Overrange conditions report a success code

**Return Type** Not Applicable

**Default** False (0)

**VB Example**

```
app.SetFailOnOverRange TRUE
On Error Goto ERRHANDLER

'the following overrange will cause ERRHANDLER to be invoked

channel.StartFrequency = 9.9 GHZ
exit

ERRHANDLER:
    print "something failed"
```

**C++ Syntax** HRESULT put\_SetFailOnOverRange(VARIANT\_BOOL mode)

**Interface** IApplication

## SetIsolationPaths Method

---

**Description** Adjusts the list of paths (port pairings) for which isolation standards will be measured during calibration.

**VB Syntax** *guidedCal.SetIsolationPaths specifier, pathList*

**Variable** [\(Type\)](#) - Description

*obj* Any of the following:  
[GuidedCalibration](#) (object)

*specifier* **(Enum)** - Choose from:

**0 - naPathsAll** - Measure isolation on all pairings of the ports that are to be calibrated.

**1 - naPathsNone** - Do not measure isolation on any pairing of the ports to be calibrated.

**2 - naPathsAdd** - Add one or more specific pairings of ports to the list of port pairings for which isolation will be measured.

**3 - naPathsRemove** - Remove one or more specific pairings of ports from the list of port pairings for which isolation will be measured.

*pathlist* **(Variant)** port numbers in pairs. One-dimensional array of Long Integers.

**Note:** *pathList* is evaluated only when *specifier* is **naPathsAdd** or **naPathsRemove**. For **naPathsAll** and **naPathsNone**, *pathList* is ignored.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Dim pathList
'selecting to measure isolation on all possible paths for the
ports about to be calibrated
guidedCal.SetIsolationPaths naPathsAll, pathList

'now removing the paths 1-to-2, 2-to-3 and 2-to-4 from the set
of all paths
pathList = Array(1,2,2,3,2,4)
guidedCal.SetIsolationPaths naPathsRemove, pathList
```

**C++ Syntax** HRESULT SetIsolationPaths(enum NAPortPathSpecifier specifier, VARIANT pathList);

**Interface** IGuidedCalibration3

---

Last Modified:

16-Apr-2007    MX New topic





## SetIPConfiguration Method

---

**Description**    Modifies settings of the PNA computer networking configuration.

**VB Syntax**    *app.SetIPConfiguration AutoIPAddress, DNSServer1, DNSServer2, HostName, DomainName, IPAddress, SubNet, Gateway, DNSSuffix1, DNSSuffix2*  
                          or  
                          *retStr = app.SetIPConfiguration (AutoIPAddress, DNSServer1, DNSServer2, HostName, DomainName, IPAddress, SubNet, Gateway, DNSSuffix1, DNSSuffix2)*

### Variable    [\(Type\)](#) - Description

<i>app</i>	An <a href="#">Application</a> (object)
<i>AutoIPAddress</i>	<b>(boolean)</b> -Choose either: <b>True</b> - PNA is assigned an IP address by a DHCP server, or will use AutoIP (Dynamic Link-Local Addressing) if DHCP server not found. <b>False</b> – PNA will use the static IP address value specified by IPAddress.
<i>DNSServer1</i>	<b>(string)</b> IP address of primary DNS server. When AutoIPAddress = True and an empty string is specified for DNSServer1, the PNA will attempt to obtain the addresses of primary and secondary DNS servers automatically. When AutoIPAddress = False, an IP address must be specified for DNSServer1 and/or DNSServer2 or else the PNA's host name will not be resolvable on the computer network.
<i>DNSServer2</i>	<b>(string)</b> IP address of secondary DNS server. When specifying an empty string for DNSServer1, then specify an empty string here also.
<i>HostName</i>	<b>(string)</b> DNS host name (computer name) to be assigned to this PNA. <b>Note:</b> If specifying a name different than the PNA's current host name, the change will not take effect until after you reboot the PNA.
<i>DomainName</i>	<b>(string)</b> DNS domain name associated with this PNA.
<i>IPAddress</i>	<b>(string)</b> Static IP address to assign to this PNA when AutoIPAddress = False. When AutoIPAddress = True, the value of IPAddress is ignored.
<i>SubNet</i>	<b>(string)</b> Subnet mask value to assign to the PNA network configuration.
<i>Gateway</i>	<b>(string)</b> Gateway address to assign to the PNA network configuration.
<i>DNSSuffix1</i>	<b>(string)</b> Primary suffix to set in the PNA DNS suffix search order. An empty string is allowed.

*DNSSuffix2* **(string)** Secondary suffix to set in the PNA DNS suffix search order. An empty string is allowed.

*retStr* **(string)** String returned by this method should be ignored. It is intended for Agilent diagnostic use.

**Return Type** String

**Default** Not Applicable

**Examples** `app.SetIPConfiguration True, "", "", "MyHostName",  
"MyRegion.MyCompany.com", "", "255.255.255.0", "123.45.67.890",  
"", ""`

```
app.SetIPConfiguration False, "123.456.78.90", "234.56.78.901",  
"MyHostName", "MyRegion.MyCompany.com", "123.456.789.0",  
"255.255.255.0", "123.45.67.890", "MyCompany.com", ""
```

**C++ Syntax** HRESULT SetIPConfiguration(VARIANT\_BOOL AutoIPAddress, BSTR DNSServer1, BSTR DNSServer2, BSTR HostName, BSTR DomainName, BSTR IPAddress, BSTR SubNet, BSTR Gateway, BSTR DNSSuffix1, BSTR DNSSuffix2, BSTR \*pRetStr);

**Interface** IApplication14

---

Last Modified:

2-Jun-2008    MX New topic

## SetPowerAcquisitionDevice Method

---

**Description** Sets the power sensor channel (A or B) to be used. This performs the same function as the **Use this sensor only** checkbox in the Power Sensor Settings dialog.

**Note:** This method is only necessary when performing an SMC calibration.

**VB Syntax** *pwrCal* . **SetPowerAcquisitionDevice** *sensor*

**Variable (Type) - Description**

*pwrCal* (Object) A SourcePowerCalibrator object

*sensor* (**enum NAPowerAcquisitionDevice** ) The power sensor channel. Choose from:

0 – naPowerSensor\_A

1 – naPowerSensor\_B

**Default** Not Applicable

**Examples** `pwrCal.PowerAcquisitionDevice naPowerSensor_A`

**C++ Syntax** HRESULT SetPowerAcquisitionDevice( tagNAPowerAcquisitionDevice enumAcqDevice);

**Interface** ISourcePowerCalibrator3

## SetFrequencyLowPass Method

---

**Description** Set the start frequencies when **trans.Mode = LowPass**.

**VB Syntax** *trans*.**SetFrequencyLowPass**

**Variable** [\(Type\)](#) - **Description**

*trans* A Transform (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `trans.SetFrequencyLowPass`

**C++ Syntax** HRESULT SetFrequencyLowPass(void)

**Interface** ITransform

## SetPortMap Method

---

**Description** Set the DUT-to-PNA port mapping for the [Noise Figure](#), [Gain Compression](#), [IMD](#), [IMDx](#), [IMS](#), or [IMSx](#) measurement.

### For Noise Figure:

Port mapping is allowed ONLY when [NoiseReceiver](#) is set to **naStandardReceiver**.

### When setting IMD and IMS channels:

- When input is 1, output can be 2 or 4.
- When input is 3, output must be 4.
- This setting is necessary only when using the **limited port mapping feature**. [Learn more.](#)

**VB Syntax** *obj.SetPortMap in,out*

### Variable [\(Type\)](#) - Description

*obj* A [GainCompression](#) (**object**) or  
A [SweptIMD](#) (**object**) or  
An [IMSSpectrum](#) (**object**)  
A [NoiseFigure](#) (**object**) - [See example program](#)

*in* PNA port which is connected to the DUT input.

*out* PNA port which is connected to the DUT output.

**Return Type** Not Applicable  
To read port map, use:  
[DeviceInputPort Property](#)  
[DeviceOutputPort Property](#)

**Default** 1,2

**Examples** `gca.SetPortMap 2,1`

**C++ Syntax** HRESULT SetPortMap(long input\_port,long output\_port);

**Interface** IGainCompression  
ISweptIMD  
ISpectrum  
INoiseFigure6

---

Last Modified:

30-Apr-2010 Added NF Opt 028  
11-Aug-2009 Added IMD and IMSpectrum (9.0)  
29-Nov-2007 MX New topic

## SetReferenceLevel Method

---

**Description** Changes the measurement's reference level to the marker's Y-axis value.

**VB Syntax** *mark*.SetReferenceLevel

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetReferenceLevel`

**C++ Syntax** HRESULT SetReferenceLevel()

**Interface** IMarker

## SetSBPorts Method

---

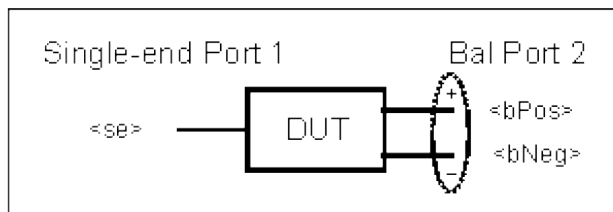
**Description** For a Single-ended - Balanced device type, maps the PNA ports to the DUT ports.  
Set the Single-ended - Balanced device type using the [DUTTopology Property](#)

**VB Syntax** *balTopology*.**SetSBPorts** *se, bPos, bNeg*

**Variable** [\(Type\)](#) - **Description**

*balTopology* A [BalancedTopology](#) (**object**)

*se, bPos, bNeg* PNA port number that connects to each of the following DUT ports:



**Return Type** Not applicable - To read port mappings, use the [BalancedTopology](#) properties.

**Default** Not Applicable

**Examples** `balTop.SetSBPorts 1,2,3`

**C++ Syntax** HRESULT SetSBPorts (long se, long bPos, long bNeg)

**Interface** IBalancedTopology



## SetSSBPorts Method

---

**Description** For a Single-ended - Single-ended - Balanced device type, maps the PNA ports to the DUT ports.

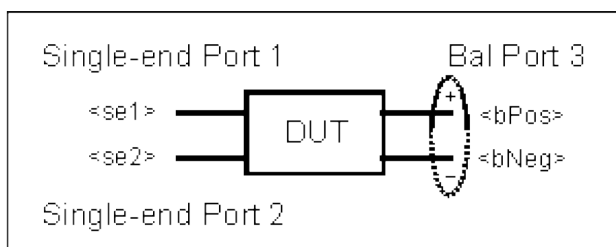
Set the Single-ended - Single-ended - Balanced device type using the [DUTTopology Property](#)

**VB Syntax** *balTopology.SetSSBPorts se, se2, bPos, bNeg*

**Variable** [\(Type\)](#) - Description

*balTopology* A [BalancedTopology](#) (object)

*se, se2, bPos, bNeg* PNA port number that connects to each of the following DUT ports:



**Return Type** Not applicable - To read port mappings, use the [BalancedTopology](#) properties.

**Default** Not Applicable

**Examples** `balTop.SetSSBPorts 1,2,3,4`

**C++ Syntax** HRESULT SetSSBPorts (long se, long se2, long bPos, long bNeg)

**Interface** IBalancedTopology

## SetupMeasurementsForStep Method

---

**Description** Show the Cal Window, and optionally one or more other specific windows, before acquiring a Cal standard. This command will cause the Cal Window to display the specific measurements that are to be made for that particular Cal standard.

[See custom Cal window commands.](#)

**VB Syntax** *guidedCal*.SetupMeasurementsForStep (*n*)

**Variable** [\(Type\)](#) - Description

*guidedCal* A [GuidedCalibration](#) (object)

*n* Step number in the calibration process.

Use [GenerateSteps](#) to determine the total number of steps.

Use [GetStepDescription](#) to read the description of each step.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `guidedCal.SetupMeasurementsForStep 3`

[See example using this command](#)

**C++ Syntax** HRESULT SetupMeasurementsForStep(long step);

**Interface** IGuidedCalibration4

---

Last Modified:

8-Nov-2007    MX New topic

## SetStandardsForClass Method

---

**Description** Set the calibration standard numbers for a specified calibration class. To read the cal standard numbers use [GetStandardsForClass Method](#)

**VB Syntax** `calKit.SetStandardsForClass (calclassorder, std1, std2, std3, std4, std5, std6, std7)`

**Variable** [\(Type\)](#) - Description

*calKit* A CalKit (**object**)

*calclassorder* (**enum NACalClassOrder**) Cal Class. Choose from:

- 0 - naRefl\_1\_S11
- 1 - naRefl\_2\_S11
- 2 - naRefl\_3\_S11
- 3 - naTran\_1\_S21
- 4 - naRefl\_1\_S22
- 5 - naRefl\_2\_S22
- 6 - naRefl\_3\_S22
- 7 - naTran\_1\_S12
- 8 - naRefl\_1\_S33
- 9 - naRefl\_2\_S33
- 10 - naRefl\_3\_S33
- 11 - naTran\_1\_S32
- 12 - naTran\_1\_S23
- 13 - naTran\_1\_S31
- 14 - naTran\_1\_S13
- 15 - naTRL\_T
- 16 - naTRL\_R
- 17 - naTRL\_L

*std1...std7* (**long**) Calibration Standard Number. Choose from **1** through **30**. Std2 through Std7 are optional

**Return Type** Not applicable

**Default** Not applicable

**Examples** **calkit.SetStandardsForClass** naRefl\_3\_S11, 3, 5, 6  
**calkit.SetStandardsForClass** naTran\_1\_S21, 4

**C++ Syntax** HRESULT SetStandardsForClass(NACalClassOrder calclassorder, long std1, long std2, long std3, long std4, long std5, long std6, long std7)

**Interface** ICalkKit

## SetStart Method

---

**Description** Changes the start stimulus to the stimulus value of the marker. The stop stimulus stays the same and the span is adjusted.

This command does not work with channels that are in [CW](#) or [Segment Sweep](#) mode.

**VB Syntax** *mark*.**SetStart**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetStart`

**C++ Syntax** HRESULT SetStart()

**Interface** IMarker

## SetStop Method

---

**Description** Changes the stop stimulus to the stimulus value of the marker. The start stimulus stays the same and the span is adjusted.

This command does not work with channels that are in [CW](#) or [Segment Sweep](#) mode.

**VB Syntax** *mark*.**SetStop**

**Variable** [\(Type\)](#) - Description

*mark* A Marker (**object**)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `mark.SetStop`

**C++ Syntax** HRESULT SetStop()

**Interface** IMarker

## ShowMarkerReadout Method

---

**Description** Shows and Hides the Marker readout for the active marker in the upper-right corner of the window.

**VB Syntax** *win.ShowMarkerReadout state*

**Variable** [\(Type\)](#) - Description

*win* A NAWindow (**object**)

*state* (**boolean**) -  
**True (1)** - Show the Marker readout  
**False (0)** - Hide the Marker readout

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `win.ShowMarkerReadout True`

**[C++ Syntax](#)** HRESULT ShowMarkerReadout(VARIANT\_BOOL bState)

**Interface** INAWindow

## ShowStatusBar Method

---

**Description** Shows and Hides the Status Bar. The Status Bar is located across the bottom of the display. The following information is shown for the active measurement:

- Channel number
- Parameter
- Correction On or Off
- Remote or Local operation

**VB Syntax** *app.ShowStatusBar state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* **(boolean)** -  
**True (1)** - Show the Status Bar  
**False (0)** - Hide the Status Bar

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.ShowStatusBar True`

**C++ Syntax** HRESULT ShowStatusBar (VARIANT\_BOOL bState)

**Interface** IApplication



## ShowStimulus Method

---

**Description** Shows and Hides the Stimulus (X-axis) information located at the bottom of the display. The start and stop stimulus values are shown for the active measurement.

**VB Syntax** *app.ShowStimulus state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* **(boolean)** -  
**True (1)** - Show the Stimulus information  
**False (0)** - Hide the Stimulus information

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.ShowStimulus True`

**[C++ Syntax](#)** HRESULT ShowStimulus(VARIANT\_BOOL bState)

**Interface** IApplication

## ShowTable Method

---

**Description** Shows or Hides the specified table for the window's active measurement in the lower part of the window.

**VB Syntax** *win.ShowTable value*

**Variable** [\(Type\)](#) - Description

*win* A NAWindow (**object**)

*value* (**enum naTable**) - The table to show or hide. Choose from:

0 - naTable\_None

1 - naTable\_Marker

2 - naTable\_Segment

3 - naTable\_Limit

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `win.ShowTable naTable_limit`

**C++ Syntax** HRESULT ShowTable (tagNATableType table)

**Interface** INAWindow

## ShowTitleBars Method

---

**Description** Shows and Hides the Title Bars. The Title Bars are across the top of the Network Analyzer Window and each of the measurement windows. The Window name is shown in the Title Bar.

**VB Syntax** *app.ShowTitleBars state*

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*state* **(boolean)**  
**True (1)** - Show the Title Bars  
**False (0)** - Hide the Title Bars

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.ShowTitleBars True`

**C++ Syntax** HRESULT ShowTitleBars/(VARIANT\_BOOL bState)

**Interface** IApplication

## ShowToolbar Method

---

**Description** Shows and Hides the specified Toolbar.

**VB Syntax** *app.ShowToolbar toolbar,state*

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (**object**)

*toolbar* (**enum NAToolbarType**) - The toolbar to show or hide. Choose from:

0 - naToolbar\_None  
1 - naToolbar\_ActiveEntry  
2 - naToolbar\_Markers  
3 - naToolbar\_Measurement  
4 - naToolbar\_Stimulus  
5 - naToolbar\_SweepControl

*state* (**boolean**) -  
**True (1)** - Show the specified toolbar  
**False (0)** - Hide the specified toolbar

**Return Type** Not Applicable

**Default** 1 - naToolbar\_ActiveEntry showing; all others hiding.

**Examples** `app.ShowToolbar 1,1 'shows the active entry toolbar`

**[C++ Syntax](#)** HRESULT ShowToolbar(tagNAToolbarType toolbar, VARIANT\_BOOL bState)

**Interface** IApplication

## Single Method

---

**Description** Sets the trigger count to 1 which will cause the channel to respond once to the [trigger source](#).

**How** the channel responds to a single trigger depends on the [trigger mode](#) (point, trace, and so forth.)

With the exception of the 'sync' argument, this command behaves like the [channel 'single' setting](#) from the user interface.

This setting has implications on Calibration. [Learn more.](#)

**VB Syntax** `chan.Single [sync]`

**Variable** [\(Type\)](#) - **Description**

*chan* A Channel (**object**)

*[sync]* (**boolean**) -Optional argument.

- **True** - The PNA waits (blocks execution) until the entire acquisition process is completed.
- **False** - The PNA returns immediately - does NOT wait for acquisition to complete (non-blocking). Default setting.

When trigger source is set to Manual:

- with *sync* = **True**, trigger source automatically changes to Internal which sends AND allows one trigger signal, then changes back to Manual.
- with *sync* = **False**, a trigger signal must also be sent using [app.ManualTrigger Method](#).

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
sync = True
chan.Single sync
```

**C++ Syntax** HRESULT Single(VARIANT\_BOOL bWait)

**Interface** IChannel

---

Last Modified:

10-Jun-2009    Minor edits

10-Jul-2007    Corrected for manual mode with sync = True



## Store Method

---

<b>Description</b>	Saves the path configuration currently associated with channel (ch) to the specified configuration name.  This command is identical to PathConfigurationManager. <a href="#">StoreConfiguration Method</a>
<b>VB Syntax</b>	<i>pathMgr.StoreConfiguration ch, name</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>pathMgr</i>	<a href="#">PathConfigurationManager</a> (object)
<i>ch</i>	<b>(Long)</b> Channel number of the configuration to be saved.
<i>name</i>	<b>(String)</b> Configuration name. Factory configurations can NOT be overwritten. Specifying the name of a pre-defined factory configuration will result in an error.
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>path.StoreConfiguration(2) "myMixer"</pre>
<b>C++ Syntax</b>	HRESULT StoreConfiguration( long channelNum, BSTR configName );
<b>Interface</b>	IPathConfigurationManager

---

Last modified:

Dec.12, 2006    MX New Command

## StoreConfiguration Method

---

**Description** Saves the path configuration currently associated with channel (ch) to the specified configuration name.

**VB Syntax** *pathMgr.StoreConfiguration ch, name*

**Variable** [\(Type\)](#) - Description

*pathMgr* [PathConfigurationManager](#) (object)

*ch* **(Long)** Channel number of the configuration to be saved.

*name* **(String)** Configuration name. Factory configurations can NOT be overwritten. Specifying the name of a pre-defined factory configuration will result in an error.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `path.StoreConfiguration(2) "myMixer"`

**C++ Syntax** HRESULT StoreConfiguration( long channelNum, BSTR configName );

**Interface** IPathConfigurationManager

---

Last Modified:

14-Dec-2006    MX New topic



## StringToNACalClass Method

---

**Description** Converts the returned strings from [GetStandardsList](#) into the enumeration (NACalClass) and the port numbers required for [PutStandard](#) and [GetStandard](#) methods that transmit data in and out of the Cal Set.

Learn more about [reading and writing Cal data using COM](#)

**VB Syntax** *CalSet*.**StringToNACalClass** (*list*, *std*, *rcv*, *src*)

**Variable** [\(Type\)](#) - **Description**

*CalSet* **(object)** - A Cal Set object

*list* **(string)** - a string containing the textual description of the standard.

*std* **(enum NACalClass)** Choose from:

1 - naClassA

2 - naClassB

3 - naClassC

4 - naClassD

5 - naClassE

6 - naReferenceRatioLine

7 - naReferenceRatioThru

### SOLT Standards

1 - naSOLT\_Open

2 - naSOLT\_Short

3 - naSOLT\_Load

4 - naSOLT\_Thru

5 - naSOLT\_Isolation

### TRL Standards

1 - naTRL\_Reflection

2 - naTRL\_Line\_Reflection

3 - naTRL\_Line\_Tracking

4 - naTRL\_Thru

5 - naTRL\_Isolation

*rcv* **(long)** - port number of the receiver

*src* **(long)** - port number of the source

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `guid = CalSet.StringToNACalClass(list, std, rcv, src)`

**C++ Syntax** HRESULT StringtoNACalClass ( BSTR\* str, NACalClass\* item, long \*rcv, long \*src);

**Interface** ICalSet

## StringToNAErrorTerm2 Method

---

**Description** Converts the returned strings from [GetErrorTermList](#) into the enumeration (NAErrorTerm2) and the port numbers required for [PutErrorTerm](#) and [GetErrorTerm](#) methods that transmit data in and out of the Cal Set.

Learn more about [reading and writing Cal data using COM](#)

**VB Syntax** *Cal Set*.StringToNAErrorTerm2 (*list*, *eterm*, *rcv*, *src*)

**Variable** [\(Type\) - Description](#)

*Cal Set* **(object)** - A Cal Set object

*list* **(string)** - a string containing the textual description of the error term.

*eterm* **(enum As NAErrorTerm2)**. Choose from:

0 - naET\_Directivity (rcv = src)  
 1 - naET\_SourceMatch (rcv = src)  
 2 - naET\_ReflectionTracking (rcv = src)  
 3 - naET\_TransmissionTracking (rcv != src)  
 4 - naET\_LoadMatch (rcv != src)  
 5 - naET\_Isolation (rcv != src)

*rcv* **(long)** - port number of the receiver

*src* **(long)** - port number of the source

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `CalSet.StringToNAErrorTerm2 str, term, rcv, src`

**C++ Syntax** HRESULT StringToNAErrorTerm2 (BSTR\* str, NAErrorTerm2\* item, long \*rcv, long \*src);

**Interface** ICalSet

## SweepOnlyCalChannelDuringCalAcquisition Method

---

**Description** Clears ALL flags for channels to sweep during calibration except the Cal channel. To flag a channel, see [AllowChannelToSweepDuringCalAcquisition Method](#)

**VB Syntax** *calMgr*.**SweepOnlyCalChannelDuringCalAcquisition**

**Variable** [\(Type\)](#) - [Description](#)

*calMgr* **(object)** - A [CalManager](#) object

**Return Type** Not Applicable

**Default** Not Applicable

**Example** `calMgr.SweepOnlyCalChannelDuringCalAcquisition`

[See example using this command](#)

**C++ Syntax** HRESULT SweepOnlyCalChannelDuringCalAcquisition()

**Interface** ICalManager5

---

Last Modified:

8-Nov-2007    MX New topic

## TestsetCatalog Method

---

**Description** Returns a list of supported testsets.

**VB Syntax** `data = Tsets.TestsetCatalog`

**Variable** [\(Type\)](#) - Description

*data* **(variant array)** - Variable to store the returned data.

*Tsets* **(object)** - An [ExternalTestSets](#) collection

**Return Type** Variant

**Default** Not Applicable

**Examples** `value = Tsets.TestsetCatalog`

**C++ Syntax** HRESULT TestsetCatalog (VARIANT\* Data);

**Interface** IExternaTestSets

## UserPreset Method

---

<b>Description</b>	Performs a User Preset. There must be an active User Preset state file (see <a href="#">UserPresetLoadFile</a> and <a href="#">UserPresetSaveState</a> ) or an error will be returned.  Regardless of the state of the <a href="#">User Preset Enable</a> checkbox, the <a href="#">app.Preset</a> command will always preset the PNA to the factory preset settings, and <a href="#">app.UserPreset</a> will always perform a User Preset.
<b>VB Syntax</b>	<i>app</i> .UserPreset
<b>Variable</b>	<b>(Type) - Description</b>
<i>app</i>	An <a href="#">Application</a> (object)
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>app.UserPreset</code>
<b>C++ Syntax</b>	HRESULT UserPreset()
<b>Interface</b>	IApplication6

## UserPresetLoadFile Method

---

**Description** Loads an existing instrument state file (.sta or .cst) to be used for User Preset. Subsequent execution of [app.UserPreset](#) will cause the PNA to assume this instrument state.

Regardless of the state of the [User Preset Enable](#) checkbox, the [app.Preset](#) command will always preset the PNA to the factory preset settings, and [app.UserPreset](#) will always perform a User Preset.

**VB Syntax** `app.UserPresetLoadFile (file)`

**Variable** [\(Type\)](#) - Description

*app* An [Application](#) (object)

*file* **(String)** Full path, name, and extension of the file to be loaded.

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.UserPresetLoadFile ("C:/Program Files/Agilent/Network Analyzer/Documents/10MHzto20GHz.sta")`

**C++ Syntax** HRESULT UserPresetLoadFile (BSTR bstrFile)

**Interface** IApplication6

## UserPresetSaveState Method

---

**Description** Saves the current instrument settings as UserPreset.sta. Subsequent execution of [app.UserPreset](#) will cause the PNA to assume this instrument state.

Regardless of the state of the [User Preset Enable](#) checkbox, the [app.Preset](#) command will always preset the PNA to the factory preset settings, and [app.UserPreset](#) will always perform a User Preset.

**VB Syntax** `app.UserPresetSaveState`

**Variable** [\(Type\)](#) - **Description**

*app* An [Application](#) (object)

**Return Type** Not Applicable

**Default** Not Applicable

**Examples** `app.UserPresetSaveState`

**C++ Syntax** HRESULT UserPresetSaveState()

**Interface** IApplication6



## WriteData Method

---

<b>Description</b>	Writes a 13-bit value to the specified address using the AD0 through AD12 lines of the external test set connector. The PNA generates the appropriate timing signals. It automatically controls timing signals LDS, LAS and RLW to strobe the address, then the data, to the external test set. See the <a href="#">timing diagram</a> for Address and Data I/O read.
<b>VB Syntax</b>	<i>ExtIO.WriteData</i> ( <i>address</i> ) = <i>value</i>
<b>Variable</b>	<a href="#">(Type)</a> - <b>Description</b>
<i>ExtIO</i>	<b>(object)</b> - An <a href="#">External IO object</a>
<i>address</i>	<b>(variant)</b> - Address to be written to.
<i>value</i>	<b>(variant)</b> - 13-bit word to write
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<code>ExtIO.WriteData (15) = 12</code>
<b><a href="#">C++ Syntax</a></b>	HRESULT WriteData(VARIANT Address, VARIANT Data);
<b>Interface</b>	IHWExternalTestSetIO

## WriteRaw Method

**Description** Writes a 16-bit value to the external test set connector lines AD0 - AD12, RLW, LAS and LDS. The analyzer does NOT generate the appropriate timing signals. The user has control of all 16 lines using this write method.

**Note:** When RLW (pin25) is set to 1 (high) it causes lines AD0 - AD12 to float. It disables their output latches and sets the hardware for reading. LDS and LAS are not affected by this behavior.

**Below is the format of data that is written with WriteRaw:**

\* This Output will float if RLW (bit-13) is set high

Pin	Bit	Signal name
22	0	AD0*
23	1	AD1*
11	2	AD2*
10	3	AD3*
9	4	AD4*
21	5	AD5*
20	6	AD6*
19	7	AD7*
6	8	AD8*
5	9	AD9*
4	10	AD10*
17	11	AD11*
3	12	AD12*
25	13	RLW
24	14	LDS
8	15	LAS

**VB Syntax**    *ExtIO.WriteRaw value*

**Variable**    [\(Type\)](#) - Description

*ExtIO*    **(object)** - An External IO object

*value*    **(variant)** - Data to be written

**Return Type**    Not Applicable

**Default**    Not Applicable

**Examples**    `ExtIO.WriteRaw 12`

**[C++ Syntax](#)**    HRESULT WriteRaw( VARIANT Output );

**Interface**    IHWExternalTestSetIO

## WriteSnpFileWithSpecifiedPorts Method

---

<b>Description</b>	<p><b>Note:</b> This command replaces <a href="#">app.Save (.snp)</a>. This command is more explicit regarding the data to be saved, and works for PNAs with multiport test sets.</p> <p>Saves SnP data to the specified file. <a href="#">Learn more about SnP data.</a></p>
<b>VB Syntax</b>	<i>data</i> = <i>meas</i> .WriteSnpFileWithSpecifiedPorts <i>ports</i> , <i>filename</i>
<b>Variable</b>	<b>(Type) - Description</b>
<i>data</i>	<b>(Variant)</b> array to store the data.
<i>meas</i>	A <a href="#">Measurement</a> (object)
<i>ports</i>	<b>(Variant Array)</b> One dimensional array containing a list of port numbers for which snp data is requested.
<i>filename</i>	<p><b>(string)</b> - Full path, filename, and suffix to store the data.</p> <p>The suffix is not checked for accuracy. If saving 2 ports, specify "filename.s2p"; If saving 3 ports, specify "filename.s3p." and so forth.</p> <p>SnP data can be output using several data formatting options. See <a href="#">SnPFormat Property</a></p>
<b>Return Type</b>	Variant array - automatically dimensioned to the size of the data.
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>'This VBScript example can be pasted into a notepad file and run on the PNA as a macro. <a href="#">Learn how.</a>  Set pna = CreateObject("AgilentPnA835x.application") Set meas = pna.ActiveMeasurement  'List the port numbers for required data ports = Array(1,2,4)  'specify where to save the data filename="C:/Program Files/Agilent/Network Analyzer/Documents/MyData.s3p" meas.WriteSnpFileWithSpecifiedPorts ports,filename</pre>
<b>C++ Syntax</b>	HRESULT WriteSnpFileWithSpecifiedPorts(VARIANT portsToMeasure,BSTR filename);
<b>Interface</b>	IMeasurement7

---

Last modified:

9/18/06 MQ Added for multiport

## OnCalEvent

---

**Description** Triggered by a calibration event. See a list of [CAL Events](#).

**Note:** Some Severe Events are also used as Error Messages

**VB Syntax** Sub *app\_OnCalEvent*(ByVal *eventID* As Variant, ByVal *chanNum* As Variant, ByVal *measNum* As Variant)

**Variable** [\(Type\)](#) - Description

*app* An Application (**object**)

*eventID* Code number of the event which occurred

*chanNum* Channel Number of the event

*measNum* Measurement Number of the event

**Return Type** Not Applicable

**Default** Not Applicable

**Examples**

```
Sub pna_OnCalEvent (ByVal eventID As Variant, ByVal  
channelNumber As Variant, ByVal measurementNumber As Variant)  
,  
MsgBox ("A Calibration event has occurred")  
End Sub
```

---

**C++ Syntax** HRESULT OnCalEvent(VARIANT eventID, VARIANT channelNumber, VARIANT measurementNumber)

**Interface** IApplication

### Selected Cal Events

512 [naEventID\\_CAL\\_QUESTIONABLE](#)

513 [naEventID\\_CAL\\_STD\\_NEEDED](#)

514 [naEventID\\_CAL\\_STATE\\_NOT\\_HW\\_COMPATIBLE](#)

515 [naEventID\\_CAL\\_REQUIRED](#)

516 [naEventID\\_CAL\\_CORRECTION\\_TURNED\\_OFF](#)

517 [naEventID\\_CAL\\_CORRECTION\\_TURNED\\_OFF\\_INTERPOLATION\\_OFF](#)

518 [naEventID\\_CAL\\_CORRECTION\\_RESTORED](#)

519 [naEventID\\_CAL\\_CORRECTION\\_TURNED\\_OFF\\_FREQRANGE\\_EXCEEDED](#)

520 [naEventID\\_CAL\\_CALTYPE\\_SET\\_TO\\_NONE](#)  
521 [naEventID\\_CAL\\_CORRECTION\\_TURNED\\_OFF\\_NOT\\_AN\\_SPARAM](#)  
524 [naEventID\\_SOURCE\\_POWER\\_CAL\\_COMPLETED](#)  
592 [naEventID\\_SOURCE\\_POWER\\_CAL\\_NOT\\_PRESENT](#)  
593 [naEventID\\_SOURCE\\_POWER\\_CAL\\_NOT\\_COMPLETE](#)  
594 [naEventID\\_SOURCE\\_POWER\\_CAL\\_REMOVE\\_TRACE](#)  
595 [naEventID\\_SOURCE\\_POWER\\_CAL\\_REMOVE\\_MEAS](#)  
596 [naEventID\\_SOURCE\\_POWER\\_CAL\\_POWER\\_CHANGED](#)  
598 [naEventID\\_INSUFFICIENT\\_SLIDE\\_MOVEMENT](#)  
613 [naEventID\\_CALSET\\_NOT\\_FOUND](#)  
615 [naEventID\\_CALSET\\_CREATED](#)  
617 [naEventID\\_CALSET\\_FILE\\_NOT\\_VALID](#)  
634 [naEventID\\_CALSET\\_LOAD\\_FAILED](#)  
635 [naEventID\\_CALSET\\_SAVE\\_FAILED](#)  
636 [naEventID\\_CALSET\\_DELETED](#)  
637 [naEventID\\_CALSET\\_FILE\\_NOT\\_COMPATIBLE](#)  
639 [naEventID\\_NEW\\_CALSET\\_FILE\\_CREATED](#)  
640 [naEventID\\_CAL\\_SET\\_IN\\_USE](#)  
644 [naEventID\\_CAL\\_COULD\\_NOT\\_TURN\\_ON](#)  
693 [naEventID\\_ERROR\\_FIXTURING\\_S2PFILE\\_CANNOT\\_OPEN](#)  
696 [naEventID\\_ERROR\\_FIXTURING\\_TURNED\\_OFF](#)  
701 [naEventID\\_MORE\\_THRU\\_PATHS\\_NEEDED](#)

#### **See Also**

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events

## OnChannelEvent

---

<b>Description</b>	Triggered by a channel event.
<b>VB Syntax</b>	Sub <i>app_OnChannelEvent</i> (ByVal <i>eventID</i> As Variant, ByVal <i>chanNum</i> As Variant)
<b>Variable</b>	<b>(Type) - Description</b>
<i>app</i>	An Application ( <b>object</b> )
<i>eventID</i>	Code number of the event which occurred
<i>chanNum</i>	Channel Number of the event
<b>Return Type</b>	Not Applicable
<b>Default</b>	Not Applicable
<b>Examples</b>	<pre>Sub pna_OnChannelEvent (ByVal eventID As Variant, ByVal channelNumber As Variant)     If eventID=naEventID_CHANNEL_CREATED then         MsgBox "Channel" + channelNumber + "was created"     End If End Sub</pre>
<b>C++ Syntax</b>	HRESULT OnChannelEvent(VARIANT eventID, VARIANT channelNumber)
<b>Interface</b>	IApplication

### Selected Channel Events

- 1792 [naEventID\\_CHANNEL\\_SWEEP\\_COMPLETE](#)
- 1793 [naEventID\\_CHANNEL\\_TRIGGER\\_COMPLETE](#)
- 1796 [naEventID\\_SET\\_CHANNEL\\_DIRTY](#)
- 1797 [naEventID\\_CLEAR\\_CHANNEL\\_DIRTY](#)
- 1801 [naEventID\\_ALL\\_SWEEPS\\_COMPLETED\\_AND\\_PROCESSED](#)
- 1805 [naEventID\\_CHANNEL\\_CREATED](#)
- 1806 [naEventID\\_CHANNEL\\_DELETED](#)
- 1876 [naEventID\\_NO\\_SOURCE\\_ATTEN](#)
- 1879 [naEventID\\_FREQ\\_OFFSET\\_OVERRANGE\\_SO\\_TURNED\\_OFF](#)
- 1883 [naEventID\\_PORT\\_NUMBER\\_OUT\\_OF\\_RANGE](#)

### See Also

[Errors and the SCPIStringParser Object](#)

---



Last modified:

March 2, 2007    Added channel create and delete

Nov. 6, 2006    Added events

## OnDisplayEvent

---

**Description** Triggered by a display event.

**VB Syntax** Sub *app\_OnDisplayEvent*(ByVal *eventID* As Variant, ByVal *winNum* As Variant, ByVal *traceNum* As Variant)

---

**Variable**    [\(Type\)](#) - Description

*app*    An Application (**object**)

*eventID*    Code number of the event which occurred

*winNum*    Window Number of the event

*traceNum*    Trace Number of the event

**Return Type**    Not Applicable

**Default**    Not Applicable

---

**Examples**

```
Sub pna_OnDisplayEvent(ByVal eventID As Variant, ByVal  
windowNumber As Variant, ByVal traceNumber As Variant)  
MsgBox ("A Display event has occurred")  
End Sub
```

---

**C++ Syntax** HRESULT OnDisplayEvent(VARIANT eventID, VARIANT windowNumber, VARIANT traceNumber)

**Interface**    IApplication

### Selected Display Events

1541 [naEventID\\_PRINT\\_SETUP\\_FAILURE](#)

1542 [naEventID\\_PRINT\\_CANCELED](#)

### See Also

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events



## OnHardwareEvent

---

**Description** Triggered by a hardware event. See a list of [Hardware Events](#)

**Note:** Some Severe Events are also used as Error Messages

**VB Syntax** Sub *app*\_**OnHardwareEvent**(ByVal *eventID* As Variant)

---

**Variable**    [\(Type\)](#) - Description

*app*    An Application (**object**)

*eventID*    Code number of the event which occurred

**Return Type**    Not Applicable

**Default**    Not Applicable

---

**Examples**

```
Private Sub pna_OnHardwareEvent (ByVal eventID As Variant)
MsgBox ("A Hardware event has occurred")
End Sub
```

---

**C++ Syntax**    HRESULT OnHardwareEvent(VARIANT eventID)

**Interface**    IApplication

### Selected Hardware Events

848 [naEventID\\_PHASELOCK](#)

852 [naEventID\\_RFPOWEROFF](#)

853 [naEventID\\_RFPOWERON](#)

855 [naEventID\\_UNLEVELED](#)

857 [naEventID\\_OVERLOAD](#)

914 [naEventID\\_TRIGGER\\_REQUIRES\\_EDGE\\_LEVEL\\_TRIGGER](#)

915 [naEventID\\_TRIGGER\\_REQUIRES\\_TRIGGER\\_OUT](#)

### See Also

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events

## OnMeasurementEvent

---

**Description** Triggered by a measurement event.

**VB Syntax** Sub *app\_OnMeasurementEvent*(ByVal *eventID* As Variant, ByVal *measNum* As Variant)

---

**Variable**    [\(Type\)](#) - Description

*app*    An Application (**object**)

*eventID*    Code number of the event which occurred

*measNum*    Measurement Number of the event

**Return Type**    Not Applicable

**Default**    Not Applicable

---

**Examples**

```
Private Sub pna_OnMeasurementEvent (ByVal eventID As Variant, ByVal
measurementNumber As Variant)

MsgBox ("A Measurement event has occurred")

End Sub
```

---

**C++ Syntax**    HRESULT OnMeasurementEvent(VARIANT eventID, VARIANT measurementNumber)

**Interface**    IApplication

### Selected Measurement Events

1024 [naEventID\\_NO\\_VALID\\_MEMORY\\_TRACE](#)

1028 [naEventID\\_LIMIT\\_FAILED](#)

1029 [naEventID\\_LIMIT\\_PASSED](#)

1034 [naEventID\\_MEMORY\\_NOT\\_SAVED](#)

1035 [naEventID\\_SET\\_AVERAGE\\_COMPLETE](#)

1036 [naEventID\\_CLEAR\\_AVERAGE\\_COMPLETE](#)

1111 [naEventID\\_MARKER\\_BANDWIDTH\\_NOT\\_FOUND](#)

1112 [naEventID\\_PEAK\\_NOT\\_FOUND](#)

1113 [naEventID\\_TARGET\\_VALUE\\_NOT\\_FOUND](#)

## See Also

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events

## OnSCPIEvent

---

**Description** Triggered by a SCPI event.

**Note:** Some Severe Events are also used as Error Messages

**VB Syntax** Sub *app*\_OnSCPIEvent(ByVal *eventID* As Variant)

---

**Variable**    [\(Type\)](#) - Description

*app*    An Application (**object**)

*eventID*    Code number of the event which occurred

**Return Type**    Not Applicable

**Default**    Not Applicable

---

**Examples**

```
Private Sub pna_OnSCPIEvent(ByVal eventID As Variant)
MsgBox ("A SCPI event has occurred")
End Sub
```

---

**C++ Syntax**    HRESULT OnSCPIEvent(VARIANT eventID )

**Interface**    IApplication

### Selected SCPI Parser Events

1281 [naEventID\\_NOTHING\\_TO\\_SAY](#)

1284 [naEventID\\_SCPI\\_STATUS\\_BYTE\\_CHANGE](#)

1360 [naEventID\\_BAD\\_SCPI\\_EXECUTE](#)

1375 [naEventID\\_CALC\\_MEASUREMENT\\_SET\\_TO\\_NONE](#)

### See Also

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events





## OnSystemEvent

---

**Description** Triggered by a system event. See a list of [System Events](#), also known as general events.

See also [EnableSourceUnleveledEvents Property](#)

**Note:** Some Severe Events are also used as Error Messages

**VB Syntax** Sub *app*\_**OnSystemEvent**(ByVal *eventID* As Variant)

---

**Variable**    [\(Type\)](#) - Description

*app*    An Application (**object**)

*eventID*    Code number of the event which occurred

*chanNum*    Channel Number of the event

**Return Type**    Not Applicable

**Default**    Not Applicable

---

**Examples**

```
Private Sub pna_OnSystemEvent (ByVal eventID As Variant)
MsgBox ("A System event has occured")
End Sub
```

---

**C++ Syntax**    HRESULT OnSystemEvent(VARIANT eventID)

**Interface**    IApplication

### Selected System Events

2048 [naEventID\\_OPTION\\_NOT\\_INSTALLED](#)

2049 [naEventID\\_FEATURE\\_NOT\\_AVAILABLE](#)

2050 [naEventID\\_FEATURE\\_NOT\\_VALID](#)

2051 [naEventID\\_SAVEFILE\\_OK](#)

2063 [naEventID\\_RECALLFILE\\_SUCCESS](#)

2130 [naEventID\\_PRINTER\\_TROUBLE](#)

2133 [naEventID\\_TRIGGERDENIED](#)

2134 [naEventID\\_MACRO\\_FAILED](#)

2144 [naEventID\\_NO\\_LICENSE](#)  
2163 [naEventID\\_PRESET](#)  
2166 [naEventID\\_TRIGGERFAILED](#)

#### **See Also**

[Errors and the SCPIStringParser Object](#)

---

Last modified:

Nov. 6, 2006    Added events

## OnUserEvent

---

**Description** Reserved for future use.

**VB Syntax** Sub app\_**OnUserEvent**

## C++ Example

The following example uses the smart pointer created by Microsoft Visual Studio. The calls to CoInitialize and CoUninitialize open and close the COM libraries.

Also notice that the pointers local to the main routine are explicitly released. When smart pointers go out of scope, they will perform this duty implicitly. However, we are calling CoUninitialize before they have the chance to be destroyed, so we are obliged to release them.

```
// An example program to illustrate the use of #import to bind to the
// PNA type library.
//

#ifdef _UNICODE
#define _UNICODE
#endif

#include "stdafx.h"
#include "stdio.h"
#include "math.h"

////////////////////////////////////
// import the network analyzer type library
////////////////////////////////////
#import "C:/Program Files/Common Files/Agilent/Pna/835x.tlb" no_namespace,
named_guids
////////////////////////////////////
// include the error definitions for the PNA so we can implement
// error handling.
////////////////////////////////////
#include "C:/Program Files/Common Files/Agilent/Pna/errorsystemmessage.h"

IApplicationPtr pNA; // top level application pointer
float fScalarData [1601]; // global buffer for data retrieval
float fScalarData2[1601];

DWORD dwCookie;

////////////////////////////////////
// SetupChannel:
//
// input: pointer to the channel
//
// function: sets properties on the channel
////////////////////////////////////
void SetupChannel(IChannelPtr pChannel)
{
    pChannel->put_StartFrequency( 1.2E9 );
    pChannel->put_StopFrequency ( 4.2E9 );
    pChannel->put_NumberOfPoints ( 201);
```

```

}

////////////////////////////////////
// AcquireData:
//
// input: pointer to the channel
//
// function: single sweeps the channel
////////////////////////////////////
void AcquireData( IChannelPtr pChannel )
{
    pChannel->Single( TRUE );
}

////////////////////////////////////
// ReadData:
//
// input: pointer to the Measurement object
//
// function: reads data from the measurment's formatted
// result data buffer
////////////////////////////////////
void ReadScalarData( IMeasurementPtr pMeas )
{
    IArrayTransferPtr pDataTransfer;
    pDataTransfer = pMeas;
    long numVals = 1601;
    float* pData = fScalarData;

    if(pDataTransfer) {

        pDataTransfer->getScalar( naMeasResult, naDataFormat_LogMag, &numVals, pData);

        for (int i = 0; i < numVals; i++)
            printf("%d\t%f\n",i,pData[i]);
        }
    TCHAR msg[100];
    BSTR param;
    pMeas->get_Parameter(&param);
    swprintf(msg,L"Review %s data",param);
    MessageBox(NULL,msg,L"User Message",0);
    ::SysFreeString(param);
}

void ReadComplexData( IMeasurementPtr pMeas )
{
    IArrayTransferPtr pDataTransfer;
    pDataTransfer = pMeas;
    long numVals = 1601;
    float* pReal= fScalarData;
    float* pImag = fScalarData2;

```

```

if(pDataTransfer){

pDataTransfer->getPairedData( naRawData, naRealImaginary, &numVals, pReal, pImag);

for (int i = 0; i < numVals; i++)
printf("%d/t%f/t%f/n",i,pReal[i], pImag[i]);
}
TCHAR msg[100];
BSTR param;
pMeas->get_Parameter(&param);
swprintf(msg,L"Review %s data",param);
MessageBox(NULL,msg,L"User Message",0);
::SysFreeString(param);
}
////////////////////////////////////
// PutData:
//
// input: pointer to the Measurement object
//
// function: writes data to the measurment's raw data
// buffer
////////////////////////////////////
void PutData( IMeasurementPtr pMeas )
{
    IArrayTransferPtr pDataTransfer;
    pDataTransfer = pMeas;
    long numVals = 201;

    if(pDataTransfer){

NAComplex* pComplex = new NAComplex[numVals];

pComplex[0].Im = 0;
pComplex[0].Re = 1;
for (int i = 1; i < numVals; i++)
{
pComplex[i].Im = (float)sin(i)/i;
pComplex[i].Re = (float)cos(i)/i;
}

pDataTransfer->putNAComplex( naRawData, numVals, pComplex, naDataFormat_Polar);
delete [] pComplex;
}
}

////////////////////////////////////
// printError
////////////////////////////////////
void printError( HRESULT hr)
{
    BSTR text;

```

```

    hr = pNA->get_MessageText ((NAEventID) hr, &text);
    MessageBox(NULL,text,L"Network Analyzer error",0);
    ::SysFreeString(text);
}

////////////////////////////////////
// main
////////////////////////////////////
int main(int argc, char* argv[])
{
    HRESULT hr;
    const long channel1 = 1;
    const long window1 = 1;
    const long srcport = 1;
    IMeasurementPtr pMeasurement;
    IChannelPtr pChannel;

    // initialize COM libraries
    CoInitialize(NULL);

    try {
        pNA = IApplicationPtr("AgilentPNA835x.Application.1");

        pNA->put_Visible(TRUE);
        pNA->Reset();

        pNA->CreateMeasurement (channel1, "S21",srcport, 3);
        hr = pNA->get_ActiveChannel( &pChannel);

        if (SUCCEEDED (hr))
        {
            SetupChannel( pChannel);
            AcquireData(pChannel);
        }

        hr= pNA->get_ActiveMeasurement( &pMeasurement);
        if (SUCCEEDED(hr))
        {
            pMeasurement->put_Format( naDataFormat_Polar);
            ReadScalarData( pMeasurement);
            ReadComplexData( pMeasurement);
            PutData(pMeasurement);
        }
        if (FAILED(hr))
        {
            printError(hr);
        }

        // make sure to release the remaining pointers

```



```
// before calling CoUninitialize

pMeasurement.Release();
pChannel.Release();
pNA.Release();
}
catch (_com_error err)
{
    printError( err.Error() );
}

CoUninitialize();
return 0;
}
```

---

## Reading Cal Set Data using COM

---

This example iterates over the entire collection of Cal Sets that currently reside in the PNA. It reads the entire list of error term strings from each Cal Set and queries the data for each term. It then does the same for the standards data.

Learn more about [Reading and Writing Calibration data using COM](#).

Learn more about [Cal Sets](#).

---

### See Other COM Example Programs

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as CalSets.vbs. [Learn how to setup and run the macro](#).

```
Dim pna
Dim cset
Dim calsets

' create the pna object
' to run on a remote PC, substitute 'name' for the full computer name of your PNA
' to run as a macro on the PNA, remove , "name"
Set pna = CreateObject("AgilentPNA835x.Application", "name")
wscript.echo pna.IDString
' obtain the calset collection
Set calsets = pna.GetCalManager.calsets

' loop thru the calsets
Dim c
For c = 1 To calsets.count
Set cset = calsets.Item(c)

' wscript.echo prints values to a message box
wscript.echo "calset = ", cset.GetGUID, cset.Description

' iterate through error terms data
Dim vterms
vterms = cset.GetErrorTermList2(0, "")
if (Not IsEmpty(vterms)) then
For i = LBound(vterms) To UBound(vterms)
wscript.echo vterms(i)
vdata = cset.GetErrorTermByString(0,vterms(i) )
wscript.echo vdata(1,0) , vdata(1,1)
Next
end if

' iterate through standards data
vterms = cset.GetStandardList2("")
if (Not IsEmpty(vterms)) then
For i = LBound(vterms) To UBound(vterms)
```

```
wscript.echo vterms(i)
vdata = cset.GetStandardByString( vterms(i) )
wscript.echo vdata(1,0), vdata(1,1)
Next
end if
Next
```

## Getting Trace Data from the Analyzer

This Visual Basic program:

- Retrieves Scalar Data from the Analyzer and plots it.
- Retrieves Paired Data from the Analyzer and plots it.
- Retrieves Complex Data from the Analyzer and plots it.

To use this code, prepare a form with the following:

- Two MSCharts named **MSChart1** and **MSChart2**
- Three buttons named **GetScalar**, **GetPaired**, **GetComplex**

**Note:** You can get MSChart in Visual Basic by clicking **Project / Components / Microsoft Chart Control**

```
'Put this in a module
Public dlocation As NADataStore
Public numpts As Long
Public fmt As NADataFormat
Public app As Application
Public measData As IArrayTransfer
Public chan As Channel

Sub Form_Load()
'Change analyzerName to your analyzer's full computer name
Set app = CreateObject("AgilentPNA835x.Application", "analyzerName")

Set measData = app.ActiveMeasurement
Set chan = app.ActiveChannel

'To pick a location to get the data from remove the comment from one of these
dlocation = naRawData
'dlocation = naCorrectedData
'dlocation = naMeasResult
'dlocation = naRawMemory
'dlocation = naMemoryResult

'setup MSChart1 and MSChart2
'right click on the chart and select:
' - line chart
' - series in rows
End Sub

Sub GetComplex_Click()
ReDim Data(numpts) As NAComplex
Dim Real(201) As Single
Dim Imag(201) As Single
numpts = chan.NumberOfPoints
```

```

'You cannot change the format of Complex Data
Call trigger
'get data
measData.GetNAComplex dlocation, numpts, Data(0)
'plot data
Dim i As Integer

For i = 0 To numpts - 1
    Real(i) = Data(i).Re
    Imag(i) = Data(i).Im
Next i
MSChart1 = Real()
MSChart2.Visible = True
MSChart2 = Imag()
Call Sweep
End Sub

```

```

Sub GetPaired_Click()
ReDim Real(numpts) As Single
ReDim Imag(numpts) As Single
numpts = chan.NumberOfPoints

```

```

' To pick a format, remove the comment from one of these
fmt = naLogMagPhase
'fmt = naLinMagPhase
Call trigger
'Get data
measData.getPairedData dlocation, fmt, numpts, Real(0), Imag(0)
'Plot Scalar
MSChart1 = Real()
MSChart2.Visible = True
MSChart2 = Imag()
Call Sweep
End Sub

```

```

Sub GetScalar_Click()
ReDim Data(numpts) As Single

```

```

numpts = chan.NumberOfPoints
'To pick a format remove the comment from one of these
fmt = naDataFormat_LogMag
'fmt = naDataFormat_LinMag
'fmt = naDataFormat_Phase
'fmt = naDataFormat_Delay
'fmt = naDataFormat_Real
'fmt = naDataFormat_Imaginary
Call trigger
'Get data
measData.GetScalar dlocation, fmt, numpts, Data(0)
'Plot Data

```

```
MSChart1 = Data()  
MSChart2.Visible = False  
Call Sweep  
End Sub
```

```
Sub trigger()
```

```
'The analyzer sends continuous trigger signals  
app.TriggerSignal = naTriggerInternal  
'The channel will only accept one, then go into hold  
'Sync true will wait for the sweep to complete
```

```
sync=True
```

```
chan.Single sync  
End Sub
```

```
Sub Sweep()  
'The channel goes back to accepting all triggers  
chan.Continuous  
End Sub
```

## Perform a Guided Calibration using COM

---

This example uses the [GuidedCalibration](#) interface to perform either a 2-port or 4-port calibration.

Learn more about [Reading and Writing Calibration data using COM](#).

---

### [See Other COM Example Programs](#)

---

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as Calibrate.vbs. [Learn how to setup and run the macro](#).

```
Set pna = CreateObject("AgilentPNA835x.Application")
Set calMgr = pna.GetCalManager
Set guidedCal = calMgr.GuidedCalibration
Set chan = pna.ActiveChannel
chanNum = chan.ChannelNumber
' Initialize guided cal to be performed on the active channel.
' The boolean argument of True indicates to store the cal only
' in the channel's calibration register. If instead you wish
' to create a new calset that the new cal will get stored to,
' comment out this next line and uncomment the three lines below it.
guidedCal.Initialize chanNum, True
' Set calset = calMgr.CreateCalSet(chanNum)
' chan.SelectCalSet calset.GetGUID, True
' guidedCal.Initialize chanNum, False

' To perform 2-port cal, Uncomment the following
' Then comment the 4-port cal

' Do 2-port cal
' TwoPortGuidedCal

' Do 4-port cal
FourPortGuidedCal

Sub TwoPortGuidedCal()
' Select the connectors
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 male"
For i = 3 To pna.NumberOfPorts
    guidedCal.ConnectorType(i) = "Not used"
Next
value = MsgBox("Connectors defined for Ports 1 and 2")
' Select the Cal Kit for each port being calibrated.
guidedCal.CalKitType(1) = "85052D"
guidedCal.CalKitType(2) = "85052D"
```

```

' To use an ECal module instead, comment out the above two lines
' and uncomment the following two lines.
' Replace N4691-60004 with your own ECAL model followed by 'ECal'.
' Your ECal module must already be connected to a PNA USB port.
'guidedCal.CalKitType(1) = "N4691-60004 ECal"
'guidedCal.CalKitType(2) = "N4691-60004 ECal"
value = MsgBox("Cal kits defined for Ports 1 and 2")
' Initiate the calibration and query the number of steps
numSteps = guidedCal.GenerateSteps
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub

Sub FourPortGuidedCal()
' Select the connectors
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 female"
guidedCal.ConnectorType(3) = "APC 3.5 female"
guidedCal.ConnectorType(4) = "APC 3.5 female"
' If a PNA which has more than 4 ports
For i = 5 To pna.NumberOfPorts
    guidedCal.ConnectorType(i) = "Not used"
Next
value = MsgBox("Connectors defined for Ports 1 to 4")
' Select the Cal Kit for each port being calibrated.
guidedCal.CalKitType(1) = "85052D"
guidedCal.CalKitType(2) = "85052D"
guidedCal.CalKitType(3) = "85052D"
guidedCal.CalKitType(4) = "85052D"
' To use an ECal module instead, comment out the above four lines
' and uncomment the following four lines.
' Replace N4691-60003 with your own ECAL model followed by 'ECal'.
' Your ECal module must already be connected to a PNA USB port.
'guidedCal.CalKitType(1) = "N4431-60003 ECal"
'guidedCal.CalKitType(2) = "N4431-60003 ECal"
'guidedCal.CalKitType(3) = "N4431-60003 ECal"
'guidedCal.CalKitType(4) = "N4431-60003 ECal"
value = MsgBox("Cal kits defined for Ports 1 to 4")
' Initiate the calibration
guidedCal.GenerateSteps
' If your selected cal kit is not a 4-port ECal module which can
' mate to all 4 ports at once, then you may want to choose which
' thru connections to measure for the cal. You must measure at
' least 3 different thru paths for a 4-port cal (for greatest
' accuracy you can choose to measure a thru connection for all 6
' pairings of the 4 ports). If you omit this command, the default
' is to measure from port 1 to port 2, port 1 to port 3, and
' port 1 to port 4. For this example we select to measure
' from port 1 to port 2, port 2 to port 3, and port 2 to port 4.
portList = Array(1,2,2,3,2,4)
guidedCal.ThruPortList = portList
' Re-generate the connection steps to account for the thru changes

```



```

numSteps = guidedCal.GenerateSteps
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub

Sub MeasureAndComplete(ByVal numSteps)
value = MsgBox("Number of steps is " + CStr(numSteps))

' Measure the standards
'The following series of commands shows that standards
'can be measured in any order. These steps acquire
'measurement of standards in reverse order.
'It is easiest to iterate through standards using
'a For-Next Loop.
For i = NumSteps To 1
    step = "Step " + CStr(i) + " of " + CStr(numSteps)
    strPrompt = guidedCal.GetStepDescription(i)
    value = MsgBox(strPrompt, vbOKOnly, step)
    guidedCal.AcquireStep i
Next

' Conclude the calibration
guidedCal.GenerateErrorTerms
MsgBox ("Cal is done!")
End Sub

```

---

Last Modified:

20-Jan-2007    Added any order to steps.

## Perform a Source Power Cal using COM

---

This program can be run in either Visual Basic 6 or as a VBScript program. The PNA can run \*.vbs programs as [macros](#).

This program demonstrates:

- Performing a source power calibration of Port 2 for Channel 1.
- Reading the calibration data.

Learn more about [Power Calibrations](#)

See an example that [Uploads a Source Power Cal](#)

### [See Other COM Example Programs](#)

To run this program, you need:

- One of the following power meters connected to the PNA through GPIB: E4416A, E4417A, E4418A/B, E4419A/B, 437B, 438A, EPM-441A, EPM-442A

**Note:** If your power meter is other than these, you can [create your own Power Meter Driver](#) using our template.

- Your PC and PNA both connected to a LAN (for communicating with each other).

To make this program work in VBS, save the following code in a text editor file such as Notepad and save as \*.vbs.

To make this program work in Visual Basic 6:

1. Create a new project
2. Click **Project, Add New Module**, click **Open**.
3. Paste the following code into the code window.
4. Delete the first two lines (comment and Main)
5. Click **Project, Properties**. Under **Startup Object**, select **Sub Main**
6. Click **Project, References**, and select the Agilent PNA Series Type Library.

```
' Run the Main subroutine
Main
Public Sub Main()
Dim PNA, chan, pwrcal                                ' PNA COM objects
Const naPowerMeter = 0, naPowerMeterAndReceiver = 1  ' enum NASourcePowerCalMethod
Const naPowerSensor_A = 0                             ' enum NAPowerAcquisitionDevice
Const naCorrectionValues = 0                          ' enum NASourcePowerCalBuffer
```

```

Const port = 2                ' PNA port #2 as source port
Const offset = 0              ' cal power offset value
Const bDisplay = True         ' whether to display data during
acquire
Dim stimulus, calvalues, strResult

' Instantiate our PNA COM objects
Set PNA = CreateObject("AgilentPNA835x.Application")
Set chan = PNA.Channels(1)
Set pwrCal = PNA.SourcePowerCalibrator

' Set the number of sweep points to 21 on Channel 1.
chan.NumberOfPoints = 21

' Specify the GPIB address of the power meter
' that will be used in performing the calibration.
pwrCal.PowerMeterGPIBAddress = 13

' Turn use of the loss table OFF (this assumes there is
' virtually no loss in the RF path to the power sensor
' due to a splitter, coupler or adapter).
pwrCal.UsePowerLossSegments = False

' Turn frequency checking OFF (so one power sensor is used for the entire cal
' acquisition sweep regardless of frequency span).
pwrCal.UsePowerSensorFrequencyLimits = False

' Specify a nominal power accuracy tolerance (IterationsTolerance) in dB for the
' calibration, and the maximum number of iterations to adjust power at each point,
' attempting to achieve within tolerance of the desired power. If at any stimulus
' point the power fails to reach within the set tolerance of the desired power
' after the maximum number of iterations, the power at that point will be set to the
' value determined by the last iteration (the Source Power Cal dialog box will
' indicate the FAIL, but we can still apply the cal if desired when it's complete).
' Each iteration is based upon a SETTLED power reading (see comments preceding the
' next two properties below).
pwrCal.IterationsTolerance = 0.1
pwrCal.MaximumIterationsPerPoint = 3

' The worst-case window of power uncertainty (for a calibration which meets
' tolerance) is the sum of the iteration tolerance and the power meter settling
' tolerance (which is described below).
' At each stimulus point, the PNA takes power meter readings and determines when
' they have settled by comparing the magnitude difference between consecutive
' readings versus a nominal dB tolerance limit (ReadingsTolerance) on that magnitude
' difference. When consecutive readings are within tolerance of each other, or
' if they are not within tolerance but we've taken a maximum number of readings
' (ReadingsPerPoint), the PNA does a weighted average of the readings taken at that
' stimulus point and that is considered our settled power reading.
pwrCal.ReadingsTolerance = 0.1
pwrCal.ReadingsPerPoint = 5

```

```

' Setup of information pertaining to this specific cal acquisition. Includes the
' method (type of devices) that will be used to perform the cal -- choose either
' naPowerMeter or naPowerMeterAndReceiver. naPowerMeterAndReceiver uses the power
' meter for the first iteration of each point and the PNA's reference receiver for
' subsequent iterations, so is much faster than using power meter only
naPowerMeter).
' But the power meter accounts for compression when calibrating at the output of an
' active device, whereas the reference receiver cannot unless it is coupled to the
' cal reference plane (on a PNA which allows direct access to the receivers).
' 'offset' specifies if the cal power level is offset (positive value for a gain,
' negative value for a loss) from the PNA port power setting on the channel when
' no source power cal is active. This is to account for components between the PNA
' test port and cal reference plane. In this example, we will calibrate at the PNA
' test port, so there is no offset (it is zero).
' 'bDisplay' indicates whether to display the source power cal dialog during the
' source power cal acquisition (the dialog will chart the corrected power readings).
pwrCal.SetCalInfo2 naPowerMeter, chan.ChannelNumber, port, offset, bDisplay

' Perform synchronous source power cal acquisition sweep using the sensor attached
' to Channel A of the power meter. This assumes that the power sensor is already
' connected to Port 2 of the PNA.
pwrCal.AcquirePowerReadings naPowerSensor_A, True

' Conclude the calibration. This applies the cal data to PNA channel memory,
' and turns the correction ON for Port 2 on Channel 1, but does NOT save the
' calibration.
pwrCal.ApplyPowerCorrectionValues

' At this point, if you choose to save the instrument state as a ".CST" file,
' the calibration will be saved with the instrument state in that file.
' Read the stimulus values from Channel 1.
stimulus = chan.GetXAxisValues

' Read the source power correction data.
calvalues = chan.getSourcePowerCalDataEx(naCorrectionValues, port)

' Print the data using a message box (here, Chr returns the ASCII characters
' for Tab (9) and Linefeed (10)).
strResult = "Stimulus" & Chr(9) & Chr(9) & "Cal Value" & Chr(10)
For i = 0 To UBound(stimulus)
strResult = strResult & stimulus(i) & Chr(9) & calvalues(i) & Chr(10)
Next
MsgBox strResult
End Sub

```

## Perform an Unguided Cal using COM

---

This example uses the [ICalibrator](#) interface to do the following:

- perform a two port calibration
- retrieve the error term data
- retrieve the standard data (cal acquisition data)

Learn more about [Reading and Writing Calibration data using COM](#).

### See Other COM Example Programs

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as Calibrate.vbs. [Learn how to setup and run the macro.](#)

```
dim pna
' To run from an external PC, substitute your PNA Name and use the following
command.
' set pna = CreateObject("AgilentPNA835x.Application", "PNA Name")
set pna = CreateObject("AgilentPNA835x.Application")
dim calibrator
set calibrator=pna.activechannel.calibrator

wscript.echo "setcalinfo for two port cal"
calibrator.setcalinfo 5,1, 2

' only have one set of standards
calibrator.Simultaneous2PortAcquisition = false

'first acquire forward reflection standards, then reverse
dim p
for p = 1 to 2

if (p = 1) then
calibrator.AcquisitionDirection = 0
else
calibrator.AcquisitionDirection = 1
end if

wscript.echo "connect open to port ", p
calibrator.acquirecalstandard 1

wscript.echo "connect short to port ", p
calibrator.acquirecalstandard 2
```

```

wscript.echo "connect load to port ", p
calibrator.acquirecalstandard 3

next

wscript.echo "connect a thru1"
calibrator.acquirecalstandard 4

'Optional - perform isolation
wscript.echo "connect loads to both ports"
calibrator.acquirecalstandard 5

wscript.echo "calculating"
calibrator.CalculateErrorCoefficients

'Calibration complete
' Now read error terms and standard data

dim termName
termName= Array("Directivity","SourceMatch","ReflectionTracking")
dim vardata

' iterate over error terms
dim t
for t = 0 to 2    ' per error term
for p = 1 to 2    ' per port
wscript.echo "Requesting ",termName(t),p,p
vardata = calibrator.GetErrorTerm( t, p, p)
next
next

' now get the path terms:  iterator each one request
termName = Array("Isolation", "LoadMatch", "TransmissionTracking")
for t = 0 to 2

wscript.echo "Requesting Forward term",termName(t),1,2
vardata = calibrator.GetErrorTerm( t, 1,2)
wscript.echo "Requesting Reverse Term",termName(t),2,1
vardata = calibrator.GetErrorTerm( t, 2,1)
next

dim stdname
stdname= Array("", "Open", "Short", "Load", "Thru", "Isolation")

' iterate over the port standards
for t = 1 to 3
for p = 1 to 2
' request the standard term for each port of interest
wscript.echo "Requesting",stdname(t),p,p

vardata = calibrator.GetStandard( t, p, p)

```

```
next
next

' now get the path standards: iterator each one request
for t = 4 to 5

wscript.echo "Requesting Forward",stdname(t),1,2
vardata = calibrator.GetStandard( t, 1,2)

wscript.echo "Requesting Reverse",stdname(t),2,1
vardata = calibrator.GetStandard( t, 2,1)

next
```

## Perform an Unknown Thru or TRL Cal

---

The following program performs either a 2-port SOLT Unknown Thru Cal or a 2-port TRL Cal. The 85052C Cal Kit used in this program contains both types of standards. This program can be run on 2-port or 4-port PNAs. When run on a multiport (4 or more ports) PNA, which does not have a reference receiver per port, a [Delta Match Cal](#) is required. See example of Delta Match Cal.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unknown.vbs. [Learn how to setup and run the macro.](#)

```
PerformUnknownThruOrTRLCal
Sub PerformUnknownThruOrTRLCal()
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
' Get the cal manager object
Set calMgr = app.GetCalManager
' Get the guided cal object
Set guidedCal = calMgr.GuidedCalibration
Set chan = app.ActiveChannel
chanNum = chan.ChannelNumber

' Initialize guided cal to be performed on the active channel.
' The boolean argument of True specifies the creation of a new calset
' for storing the new calibration.
guidedCal.Initialize chanNum, True

' Specify connectors for Ports 1 and 2
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 male"

' If your PNA has more than 2 ports, uncomment one or both of
' these next two lines, to explicitly specify this is
' just a 2-port cal.
'guidedCal.ConnectorType(3) = "Not used"
'guidedCal.ConnectorType(4) = "Not used"

' Specify cal kit for Ports 1 and 2
guidedCal.CalKitType(1) = "85052C"
guidedCal.CalKitType(2) = "85052C"

' Since the 85052C cal kit contains SOLT standards and also TRL
' standards, these next lines determine whether the cal becomes
' unknown thru (SOLT), or TRL.
' Specify cal and Thru method
guidedCal.PathCalMethod (1,2) = "SOLT"
guidedCal.PathThruMethod (1,2) = "Undefined Thru"

' To set up the cal as TRL, comment the previous line and uncomment this next line.
' The Thru method is set by default.
guidedCal.PathCalMethod (1,2) = "TRL"
```



```

' Initiate the calibration and query the number of steps
numSteps = guidedCal.GenerateSteps
MsgBox "Number of steps is " + CStr(numSteps)

' Query the list of ports that need delta match
portList = guidedCal.PortsNeedingDeltaMatch
' If portList contains just one element and it's value is 0, then that indicates
' none of the ports being calibrated require delta match data.
' If each testport on the PNA has it's own reference receiver (R channel),
' then delta match is never needed, so portList will always be just 0.
lowerBound = LBound(portList)
If (UBound(portList) <> lowerBound) Or (portList(lowerBound) <> 0) Then
    ' Delta match data is required for at least one port.
    ' For this example, we assume a Global Delta Match Cal has previously been
    ' performed so the Global Delta Match CalSet exists.
    ' Supplying an empty string to ApplyDeltaMatchFromCalSet indicates to use
    ' the Global Delta Match CalSet.
    MsgBox "here"
    guidedCal.ApplyDeltaMatchFromCalSet ""
End If

' Measure the standards
For i = 1 To numSteps
    step = "Step " + CStr(i) + " of " + CStr(numSteps)
    strPrompt = guidedCal.GetStepDescription(i)
    retVal = MsgBox(strPrompt, vbOKCancel, step)
    If retVal = vbCancel Then Exit Sub
    guidedCal.AcquireStep i
Next

' Conclude the calibration
guidedCal.GenerateErrorTerms
MsgBox "Cal is done!"

End Sub

```

## Perform Global Delta Match Cal

---

The following program performs a [Global Delta Match Calibration](#). This is required when performing an Unknown Thru cal or TRL cal on PNAs without a reference receiver for each test port. [See example](#).

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Delta.vbs. [Learn how to setup and run the macro](#).

```
Sub PerformGlobalDeltaMatchCal()  
    ' Create / Get the PNA application.  
    Set app = CreateObject("AgilentPNA835x.Application")  
    ' Get cal manager object  
    Set calMgr = app.GetCalManager  
    ' Get guided cal object  
    Set guidedCal = calMgr.GuidedCalibration  
  
    ' Initiate a Global Delta Match calibration, choosing connector and cal kit  
    numSteps = guidedCal.GenerateGlobalDeltaMatchSequence("APC 3.5 female", "85033D/E")  
    MsgBox "Number of steps is " + CStr(numSteps)  
  
    ' Measure the standards  
    For i = 1 To numSteps  
        step = "Step " + CStr(i) + " of " + CStr(numSteps)  
        strPrompt = guidedCal.GetStepDescription(i)  
        retVal = MsgBox(strPrompt, vbOKCancel, step)  
        If retVal = vbCancel Then Exit Sub  
        guidedCal.AcquireStep i  
    Next  
  
    ' Conclude the calibration  
    guidedCal.GenerateErrorTerms  
    MsgBox "Cal is done!"  
  
End Sub
```

## Perform a Guided Cal with C#

The following example performs a 2-port or 4-port Guided Cal using C#.

**Note:** Replace **<remote host name>** with the full computer name of your PNA.

[Learn more about using .NET with the PNA](#)

```
AgilentPNA835x.IApplication pna;
    AgilentPNA835x.ICalManager3 calMgr;
    AgilentPNA835x.IGuidedCalibration guidedCal;
    AgilentPNA835x.IChannel chan;
    int chanNum;
    AgilentPNA835x.ICalSet calset;
    void PerformGuidedCal()
    {
        Type pnaType = Type.GetTypeFromProgID("AgilentPNA835x.Application",
"<remote host name>");
        pna = (AgilentPNA835x.IApplication)
            Activator.CreateInstance(pnaType);
        calMgr = (AgilentPNA835x.ICalManager3)pna.GetCalManager();
        guidedCal =
(AgilentPNA835x.IGuidedCalibration)calMgr.GuidedCalibration;
        chan = pna.ActiveChannel;
        chanNum = chan.channelNumber;

        // Initialize guided cal to be performed on the active channel.
        // The boolean argument of True indicates to store the cal only
        // in the channel's calibration register. If instead you wish
        // to create a new calset that the new cal will get stored to,
        // comment out this next line and uncomment the three lines below it.
        guidedCal.Initialize(chanNum, true);
        calset = calMgr.CreateCalSet(chanNum);
        chan.SelectCalSet(calset.GetGUID(), true);
        guidedCal.Initialize(chanNum, false);

        // To perform 2-port cal, Uncomment the following
        // Then comment the 4-port cal
```

```

        // Do 2-port cal
        // TwoPortGuidedCal();

        // Do 4-port cal
        FourPortGuidedCal();
    }

    void TwoPortGuidedCal()
    {
        // Select the connectors
        guidedCal.set_ConnectorType(1, "APC 3.5 female");
        guidedCal.set_ConnectorType(2, "APC 3.5 male");
        for (int i = 3; i <= pna.NumberOfPorts; i++)
            guidedCal.set_ConnectorType(i, "Not used");
        MessageBox.Show("Connectors defined for Ports 1 and 2");
        // Select the Cal Kit for each port being calibrated.
        guidedCal.set_CalKitType(1, "85052D");
        guidedCal.set_CalKitType(2, "85052D");
        // To use an ECal module instead, comment out the above two lines
        // and uncomment the following two lines.
        // Replace N4691-60004 with your own ECAL model followed by 'ECal'.
        // Your ECal module must already be connected to a PNA USB port.
        // guidedCal.CalKitType(1) = "N4691-60004 ECal"
        // guidedCal.CalKitType(2) = "N4691-60004 ECal"
        MessageBox.Show("Cal kits defined for Ports 1 and 2");
        // Initiate the calibration and query the number of steps
        int numSteps = guidedCal.GenerateSteps();
        // Measure the standards, compute and apply the cal
        MeasureAndComplete(numSteps);
    }

    void FourPortGuidedCal()
    {
        //Select the connectors
        guidedCal.set_ConnectorType(1, "APC 3.5 female");
        guidedCal.set_ConnectorType(2, "APC 3.5 female");
    }

```

```

guidedCal.set_ConnectorType(3,"APC 3.5 female");
guidedCal.set_ConnectorType(4,"APC 3.5 female");
// If a PNA which has more than 4 ports
for (int i = 5;i<=pna.NumberOfPorts;++i)
    guidedCal.set_ConnectorType(i,"Not used");

MessageBox.Show("Connectors defined for Ports 1 to 4");
// Select the Cal Kit for each port being calibrated.
guidedCal.set_CalKitType(1,"85052D");
guidedCal.set_CalKitType(2,"85052D");
guidedCal.set_CalKitType(3,"85052D");
guidedCal.set_CalKitType(4,"85052D");
// To use an ECal module instead, comment out the above four lines
// and uncomment the following four lines.
// Replace N4691-60003 with your own ECAL model followed by 'ECal'.
// Your ECal module must already be connected to a PNA USB port.
//guidedCal.CalKitType(1) = "N4431-60003 ECal";
//guidedCal.CalKitType(2) = "N4431-60003 ECal";
//guidedCal.CalKitType(3) = "N4431-60003 ECal";
//guidedCal.CalKitType(4) = "N4431-60003 ECal";
MessageBox.Show("Cal kits defined for Ports 1 to 4");
// Initiate the calibration
guidedCal.GenerateSteps();
// If your selected cal kit is not a 4-port ECal module which can
// mate to all 4 ports at once, then you may want to choose which
// thru connections to measure for the cal. You must measure at
// least 3 different thru paths for a 4-port cal (for greatest
// accuracy you can choose to measure a thru connection for all 6
// pairings of the 4 ports). If you omit this command, the default
// is to measure from port 1 to port 2, port 1 to port 3, and
// port 1 to port 4. For this example we select to measure
// from port 1 to port 2, port 2 to port 3, and port 2 to port 4.
long[] portList = new long[6]{1,2,2,3,2,4};
guidedCal.ThruPortList = portList;
// Re-generate the connection steps to account for the thru changes
int numSteps = guidedCal.GenerateSteps();

```

```

        // Measure the standards, compute and apply the cal
        MeasureAndComplete(numSteps);
    }

    void MeasureAndComplete(int numSteps)
    {
        MessageBox.Show("Number of steps is " + numSteps.ToString());

        // Measure the standards
        // The following series of commands shows that standards
        // can be measured in any order. These steps acquire
        // measurement of standards in reverse order.
        // It is easiest to iterate through standards using
        // a For-Next Loop.
        for (int i = numSteps; i >= 1; --i)
        {
            string strPrompt = guidedCal.GetStepDescription(i);
            MessageBox.Show(strPrompt);
            guidedCal.AcquireStep(i);
        }

        // Conclude the calibration
        guidedCal.GenerateErrorTerms();
        MessageBox.Show("Cal is done!");
    }

```

Last Modified:

27-Jan-2009    New topic.

## Perform an ECal using COM

---

This example uses the [GuidedCalibration](#) interface to perform a 2-port ECal calibration.

Learn more about [Reading and Writing Calibration data using COM](#).

### [See Other COM Example Programs](#)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as ECal.vbs. [Learn how to setup and run the macro](#).

```
Set pna = CreateObject("AgilentPNA835x.Application")
Set calMgr = pna.GetCalManager
Set guidedCal = calMgr.GuidedCalibration
Set chan = pna.ActiveChannel
chanNum = chan.ChannelNumber
' Initialize guided cal to be performed on the active channel.
' The boolean argument of True indicates to create a new calset
guidedCal.Initialize chanNum, True
' To perform 3-port cal, Uncomment the following
' Then comment the 2-port cal

' Do 2-port cal
TwoPortGuidedCal

' Do 3-port cal
' ThreePortGuidedCal

Sub TwoPortGuidedCal()
'Change the following to match the connectors on your ECal module
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 female"
For i = 3 To pna.NumberOfPorts
guidedCal.ConnectorType(i) = "Not used"
Next
value = MsgBox("Connectors defined for Ports 1 and 2")
' Select the ECal module for each port being calibrated.
' Replace N4691-60004 with your own ECAL model followed by 'ECal'.
' Your ECal module must already be connected to a PNA USB port.

guidedCal.CalKitType(1) = "85092-60010 ECal"
guidedCal.CalKitType(2) = "85092-60010 ECal"

value = MsgBox("ECal Module defined for Ports 1 and 2")
' Initiate the calibration and query the number of steps
numSteps = guidedCal.GenerateSteps
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub
```

```

Sub ThreePortGuidedCal()
'Change the following to match the connectors on your ECal module
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 female"
guidedCal.ConnectorType(3) = "APC 3.5 female"

' Select the ECal module for each port being calibrated.
' Replace N4691-60003 with your own ECAL model followed by 'ECal'.
' Your ECal module must already be connected to a PNA USB port.
guidedCal.CalKitType(1) = "N4431-60003 ECal"
guidedCal.CalKitType(2) = "N4431-60003 ECal"
guidedCal.CalKitType(3) = "N4431-60003 ECal"

value = MsgBox("Cal kits defined for Ports 1 to 3")
' Initiate the calibration
numSteps = guidedCal.GenerateSteps
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub

Sub MeasureAndComplete(ByVal numSteps)
value = MsgBox("Number of steps is " + CStr(numSteps))
' Measure the standards
For i = 1 To numSteps
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = guidedCal.GetStepDescription(i)
value = MsgBox(strPrompt, vbOKOnly, step)
guidedCal.AcquireStep i
Next

' Conclude the calibration
guidedCal.GenerateErrorTerms
MsgBox ("Cal is done!")
End Sub

```



## Perform an ECal User Characterization

---

This example performs a user-characterization and stores it to both the ECal module memory and PNA disk memory.

It then performs two 2-port cals: the first using the characterization from module memory, then using the characterization from disk memory.

**Note:** This example requires that channel 1 be already calibrated.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file, such as Notepad, and save it on the PNA hard drive as ECal.vbs.

### See Also

[How to setup and run the macro.](#)

[ECalUserCharacterizer Object](#)

[About User Characterization](#)

### See Other COM Example Programs

```
Option Explicit

Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")

Dim calMgr
Set calMgr = pna.GetCalManager

' Get ECal User Characterizer COM object
Dim ecalCharacterizer
Set ecalCharacterizer = calMgr.GetECalUserCharacterizer

' Substitute here the model number and serial number of your own ECal.
' Note that this example corresponds to a 4-port ECal module with
' serial number 00001. If instead you have a 2-port ECal module,
' their model numbers are '5x5' numbers -- for example, 'N4691-60001'.

Dim ecalModelNum
ecalModelNum = "N4433A"

Dim ecalSerialNum
ecalSerialNum = "00001"

ecalCharacterizer.ECalID = ecalModelNum & "," & ecalSerialNum

MsgBox "ECal module to be characterized is: " & ecalCharacterizer.ECalID

' Set which user characterization number (1-12) the new characterization
' will be stored to in the ECal module when it is done. If you intend to
```

```

' store your user characterization just to PNA Disk Memory and NOT the
' ECal module's memory, then omit the setting of this property.
ecalCharacterizer.CharacterizationNumber = 1

' The following commented-out lines of code show how you can access
' the list of connector type names you can set for the ports of an
' ECal when you user-characterize it. However, please note that if
' you are writing the user characterization to the ECal module's memory,
' as of yet only the Factory Defined set of connector choices will work
' properly (see the ValidConnectorType property).
' If you will be saving your characterization to just
' PNA Disk Memory only, then all connector names returned by this call
' will work, user-defined connector names as well as factory-defined.
'Dim connTypeArray
'connTypeArray = ecalCharacterizer.ValidConnectorTypes
'MsgBox connTypeArray(1)

' Access element 1 in the string array
' For each port of the ECal module, specify which connector type
' is at the end of the adapter (or cable or fixture) that is
' connected to that port of the ECal for the characterization
' (must be one of the connector types that is included in the
' list that the ValidConnectorTypes method returns). The default
' is "No adapter", which assumes you are characterizing that port
' of the ECal "as is" (nothing attached to it). So in this example,
' Ports C and D of the ECal are being characterized to just the
' ECal's connectors.
ecalCharacterizer.ConnectorType(1) = "APC 3.5 male" ' ECal Port A
ecalCharacterizer.ConnectorType(2) = "APC 3.5 male" ' ECal Port B

' As with the connector types, the information set in these next
' few properties also gets stored within the characterization.
' Set the name of the person and/or company that is producing
' this characterization.
ecalCharacterizer.UserName = "John Doe, Acme Inc."

' Set user-specified description of the PNA being used.
ecalCharacterizer.UserDescriptionOfPNA = "SN US12345678"

' Set descriptions of what you have connected to the ECal module's
' ports for the characterization.

```

```

ecalCharacterizer.PortDescription(1) = "3.5 mm adapter, SN 00001" ' Port A of the
ECal
ecalCharacterizer.PortDescription(2) = "3.5 mm adapter, SN 00002" ' Port B of the
ECal
' Begin a user characterization on Channel 1.
' If you will be storing this characterization to the ECal module's memory, then
' the boolean argument to this command should be set to True. If you will be
storing
' this characterization to PNA disk memory ONLY, then you should specify False
for
' that argument. In this example we will be storing the characterization to both
' module memory and PNA disk memory, so we use True.
ecalCharacterizer.InitializeEx 1, True
' Generate the measurement steps for the user characterization.
Dim numSteps
numSteps = ecalCharacterizer.GenerateSteps
' Measure the steps.
' You must ensure you have already applied the appropriate calibration to the
channel
' already, or else an error will be thrown indicating that.
Dim i
For i = 1 To numSteps
    MsgBox ecalCharacterizer.GetStepDescription(i)
    ecalCharacterizer.AcquireStep(i)
    MsgBox "Acquire is complete"
Next
MsgBox "Now the user characterization will be saved to the ECal module and to PNA
disk memory"
' Save the user characterization to the ECal module's memory.
' Note that this can take multiple minutes, depending on how
' many sweep points the channel has.
ecalCharacterizer.SaveToECal
' Save the user characterization to PNA Disk Memory.
Dim characterizationName
characterizationName = "test"
ecalCharacterizer.SaveToDiskMemory(characterizationName)
MsgBox "User characterization is complete. Now we will calibrate using it.
First we will use it from ECal module memory."

```

```

Dim moduleMemCalKitName
moduleMemCalKitName = GetCalKitName("User " &
CStr(ecalCharacterizer.CharacterizationNumber))
DoTwoPortCal moduleMemCalKitName
MsgBox "Now we will calibrate using the characterization from PNA Disk Memory."
Dim pnaDiskMemCalKitName
pnaDiskMemCalKitName = GetCalKitName(characterizationName)
DoTwoPortCal pnaDiskMemCalKitName
MsgBox "Example has completed"

Function GetCalKitName(characterizationName)
Dim calKitName
calKitName = ecalModelNum
If Len(characterizationName) > 0 Then calKitName = calKitName & " " &
characterizationName
calKitName = calKitName & " ECal " & ecalSerialNum
GetCalKitName = calKitName
End Function

Sub DoTwoPortCal(calKitName)
' Initialize guided cal to be performed on Channel 1.
Dim guidedCal
Set guidedCal = calMgr.GuidedCalibration
guidedCal.Initialize 1, True
' Specify the DUT connector for each PNA port to be calibrated (DUT connector =
ECal characterization's connector)
guidedCal.ConnectorType(1) = "APC 3.5 male"
guidedCal.ConnectorType(2) = "APC 3.5 male"
' Specify the "cal kit" for each of those ports
guidedCal.CalKitType(1) = calKitName
guidedCal.CalKitType(2) = calKitName
' We know this example will result in a calibration sequence of a single
"connection step"
Dim numSteps
numSteps = guidedCal.GenerateSteps
' Acquire the cal connection step
guidedCal.AcquireStep 1

```

```
' Conclude the cal and turn it on  
guidedCal.GenerateErrorTerms  
End Sub
```

---

Last Modified:

18-Sep-2009    Updated with new User Char commands (9.0)  
10-Nov-2008    New topic (8.33)

## Writing Cal Set Data using COM

---

This example creates a Cal Set and then writes data to the Cal Set.

Learn more about [Reading and Writing Calibration data using COM](#).

Learn more about [Cal Sets](#).

### [See Other COM Example Programs](#)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as CalSetsWrite.vbs. [Learn how to setup and run the macro](#).

```
dim pna
dim v
Set pna=CreateObject("AgilentPNA835x.Application")
InitPhonyData
PutPhonyData
' This sub creates phony data
Sub InitPhonyData()
Dim i
Dim numpts
wscript.echo "init phony"
numpts = pna.ActiveChannel.NumberOfPoints
ReDim v(numpts - 1, 1)
For i = 0 To numpts - 1
    v(i, 0) = i
    v(i, 1) = 0
Next
End Sub

' This sub creates a Cal Set, then writes the phony data to it
Sub PutPhonyData()
Dim cmgr
Dim cset
wscript.echo "putphony"
Set cmgr = pna.GetCalManager
Set cset = cmgr.CreateCalSet(1)
cset.OpenCalSet naCalType_OnePort, 1
const directivity = 0
const sourcematch=1
const reflectiontracking =2
cset.putErrorTerm directivity,1, 1, v
cset.putErrorTerm sourcematch,1, 1, v
cset.putErrorTerm reflectiontracking ,1, 1, v
cset.CloseCalSet
cset.Description = "Phony One Port"
cset.save
```

End Sub

## Upload a Source Power Cal using COM

---

This program can be run in either Visual Basic 6 or as a VBScript program. The PNA can run \*.vbs programs as [macros](#).

This program demonstrates:

- Uploading a source power calibration of Port 2 for Channel 1.
- Reading the calibration data.

Learn more about [Power Calibrations](#)

### [See Other COM Example Programs](#)

To run this program you need:

- Your PC and PNA both connected to a LAN (for communicating with each other).

To make this program work in VBS, save the following code in a text editor file such as Notepad and save as \*.vbs.

To make this program work in Visual Basic 6:

1. Create a new project
2. Click **Project, Add New Module**, click **Open**.
3. Paste the following code into the code window.
4. Delete the first two lines (comment and Main)
5. Click **Project, Properties**. Under **Startup Object**, select **Sub Main**
6. Click **Project, References**, and select the Agilent PNA Series Type Library.

```
' Run the Main subroutine
Main
Public Sub Main()
Dim PNA, chan          ' PNA COM objects
Const naCorrectionValues = 0 ' enum NASourcePowerCalBuffer
Const port = 2          ' PNA port #2 as source port
Dim stimulus, calvalues
Dim power, calpower, strResult
' Instantiate our PNA COM objects
Set PNA = CreateObject("AgilentPNA835x.Application")
Set chan = PNA.Channels(1)
```



```

' Set the number of sweep points to 2 on Channel 1.
chan.NumberOfPoints = 2

' Ensure there's currently no source power cal on for this channel and port.
chan.SourcePowerCorrection(port) = False

' Specify if the cal power level is offset (positive value for a gain, negative
' value for a loss) from the PNA port power setting on the channel when
' no source power cal is active. This is to account for components
' between the PNA test port and cal reference plane.
' In this example, let's set up our calibration
' at the output of an amplifier with 15 dB gain.
chan.SourcePowerCalPowerOffset(port) = 15

' Send our source power correction data to the PNA. For purpose of simplicity
' in this example, we'll set up for no correction (0) at our start stimulus and
' 0.5 dB at our stop stimulus (recall that our sweep currently has just 2 points).
calvalues = Array(0, 0.5)
chan.putSourcePowerCalDataEx naCorrectionValues, port, calvalues

' Set the number of sweep points to 21 on Channel 1.
chan.NumberOfPoints = 21

' Read the fixed power level for this port on Channel 1.
power = chan.TestPortPower(port)

' Turn the source power cal on.
chan.SourcePowerCorrection(port) = True

' Again read the fixed power level for this port on Channel 1
' (with our calibration turned on, this should now include the 15 dB offset
' we indicated our power amplifier provides).
calpower = chan.TestPortPower(port)

' Read the stimulus values from Channel 1.
stimulus = chan.GetXAxisValues

' Read back the source power correction data, now interpolated for 21 points
calvalues = chan.getSourcePowerCalDataEx(naCorrectionValues, port)

' Print the data using a message box (here, Chr returns the ASCII characters
' for Tab (9) and Linefeed (10)).
strResult = "PNA port power = " & power & Chr(10)
strResult = strResult & "Power at reference plane = " & calpower & Chr(10) & hr(10)
strResult = strResult & "Stimulus" & Chr(9) & Chr(9) & "Cal Value" & Chr(10)
For i = 0 To UBound(stimulus)
strResult = strResult & stimulus(i) & Chr(9) & calvalues(i) & Chr(10)
Next
MsgBox strResult
End Sub

```

## Upload and Download Segment Table

---

These example programs use the [SetAllSegments Method](#) and [GetAllSegments Method](#) to do the following:

- Creates a 2-dimensional array (7 x 10) 7 data elements that define each segment x 10 segments
- Uploads the data to the PNA
- [Downloads a segment table](#) from the PNA

This program does not make sweep type = segment or show the segment table.

The comments indicate the order in which the segment elements are specified: Index 0 - segment state, Index 4 is IFBW, and so forth.

### [See Other COM Example Programs](#)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as \*.vbs. [Learn how to setup and run the macro.](#)

```
' Create the application instance, and preset the application
Set app = CreateObject("AgilentPNA835x.Application")
app.Preset

Dim chan
Set chan = app.ActiveChannel
chan.sweepType = 4

Dim segs
Set segs = chan.Segments

Dim win
Set win = app.NAWindows(1)
win.ShowTable 2

' Multipliers
kHz = 1000
MHz = kHz*1000
GHz = MHz*1000
' Create segments from 10MHz to 3GHz
StartFreq = 10 * MHz
StopFreq = 3 * GHz
'*
'* Create 10 segments between StartFreq and StopFreq
'*
' Create a 2-D array of segments.
```

```

' 1st dimension is size 7 (6 is max index)
' to hold all the data per segment.
' 2nd dimension is size 10 (9 is max index)
' to hold 10 total segments.
Dim segdata(6, 9)
' Width of frequency segment, used below
SegmentWidth = (StopFreq-StartFreq)/10
' Fill up all 10 segments (indicies 0 to 9) with data
For i = 0 To 9
' element 0=segment state (on or off)
segdata(0, i) = True

' element 1=Num Points in this segment
segdata(1, i) = 500

' element 2=Start Freq
segdata(2, i) = StartFreq + i * SegmentWidth

' element 3=Stop Freq
segdata(3, i) = segdata(2, i) + SegmentWidth

' element 4=IFBW
segdata(4, i) = 35000

' element 5=Dwell Time
segdata(5, i) = 0

' element 6=Power
segdata(6, i) = 0

Next

' Configure Independent segment settings
segs.IFBandwidthOption = 1
segs.SourcePowerOption = 1

' Push the segment data into the PNA's Active Channel
segs.SetAllSegments segdata

```

### Download Segment Table

```

Option Explicit
Dim app
Set app = CreateObject("AgilentPNA835x.Application")
Dim chan
Set chan = app.ActiveChannel
chan.sweepstype = 4
Dim segs

```

```

Set segs = chan.Segments
Dim win
Set win = app.NAWindows(1)
win.ShowTable 2
Dim segData
segData = segs.GetAllSegments
' Get lower bound and upper bound on the data values per each segment
Dim segDataLB, segDataUB
segDataLB = LBound(segData,1)
segDataUB = UBound(segData,1)
' Get lower bound and upper bound corresponding to how many segments
Dim segArrayLB, segArrayUB
segArrayLB = LBound(segData,2)
segArrayUB = UBound(segData,2)
' If the VB LBound and UBound functions didn't generate an error
' before reaching this point, that implies a valid two-dimensional
' array was returned into 'segData'.
WScript.Echo "Number of segments = " & segArrayUB - segArrayLB + 1
WScript.Echo "Number of data values per segment = " & segDataUB - segDataLB + 1
Dim index
index = segDataLB
Dim segInfStr
segInfStr = "Segment 1: state = " & segData(index, segArrayLB)
index = index + 1
segInfStr = segInfStr & ", num points = " & segData(index, segArrayLB)
index = index + 1
segInfStr = segInfStr & ", start freq = " & segData(index, segArrayLB)
index = index + 1
segInfStr = segInfStr & ", stop freq = " & segData(index, segArrayLB)
index = index + 1
segInfStr = segInfStr & ", IFBW = " & segData(index, segArrayLB)
index = index + 1
segInfStr = segInfStr & ", dwell time = " & segData(index, segArrayLB)
' In case of a measurement receiver PNA like N5264A
' which has no source ports, chan.SourcePortNames will
' return an empty variant (no array)

```

```

Dim srcPortNames
srcPortNames = chan.SourcePortNames
Dim srcPortNamesLB, srcPortNamesUB
srcPortNamesUB = -1
On Error Resume Next
srcPortNamesLB = LBound(srcPortNames)
srcPortNamesUB = UBound(srcPortNames)
On Error GoTo 0
If (srcPortNamesUB >= 0) And ((srcPortNamesUB - srcPortNamesLB + 1) <> (segDataUB - index)) Then
    WScript.Echo "Mismatch in number of source port names!"
End If
Dim j
For j = index + 1 To segDataUB
    segInfStr = segInfStr & ", " & srcPortNames(j - (index + 1) + srcPortNamesLB)
    & " power = " & segData(j, segArrayLB)
Next
WScript.Echo segInfStr

```

---

Last Modified:

28-Apr-2009    Added Download example

## Create and Cal an SMC Measurement

---

This example creates and calibrates an SMC measurement. To run this example without modification you need the following:

- A Mixer setup file saved on the PNA: C:/Program Files/Agilent/Network Analyzer/Documents/Mixer/MyMixer.mxr.
- If the mixer file uses an external LO source, it must be connected and configured.
- An ECal module that covers the frequency range of the measurement.
- A power meter must be connected to the PNA using GPIB.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as SMC.vbs. Learn how to setup and run the macro.

### See Also

[Create an SMC Fixed Output Meas](#)

[Use Existing Power Cal for SMC](#)

### [See Other COM Example Programs](#)

```
Dim App
Set App = CreateObject("AgilentPNA835x.Application")
Dim Meas
Set Meas = App.ActiveMeasurement
Meas.Delete

App.CreateCustomMeasurementEx 1, "Scalar Mixer/Converter", "SC21"
'Other valid strings that can be specified to create a measurement with a parameter
'other than "SC21" are: "S11", "S22", "IPwr", and "OPwr"

Set Meas = App.ActiveMeasurement
'You can perform mixer setup here or
'recall a previous mixer setup from the PNA Hard drive
Dim mix
Set mix = Meas

'Load your own SMC measurement from the PNA Hard drive.
Meas.LoadFile "C:/Program Files/Agilent/Network
Analyzer/Documents/Mixer/MyMixer.mxr"

Dim CalMgr
Set CalMgr = App.GetCalManager
Dim SMC
```

```

Set SMC = CalMgr.CreateCustomCal("SMC")
SMC.Initialize 1, 1
SMC.Do2PortEcal = 1 'specify 0 for mechanical cal, 1 for ecal
'use Factory Characterization
SMC.ECALCharacterization(1) = 0
'only specify the ThruCalMethod if you have a non-insertable DUT
'SMC.ThruCalMethod = "Adapter Removal"
'If you specify Adapter Removal or Unknown Thru calibration
'then you need to specify the connector types of your DUT.
'Specify a connector that is the same type as your selected
'ECAL characterization. The characterization selected in
'this case is APC 2.4 female APC 2.4 male so the connectors
'specified for my DUT have to be APC 2.4 but can be any sex.
'SMC.ConnectorType(1) = "APC 2.4 female"
'SMC.ConnectorType(2) = "APC 2.4 female"
SMC.OmitIsolation = 1
SMC.AutoOrient = 1
' 1- forward, 2-reverse, or Both
SMC.CalibrationPort = "1"

Dim steps
steps = SMC.GenerateSteps
For i = 1 To steps
MsgBox SMC.GetStepDescription(i)
SMC.AcquireStep i
Next

Dim calset
calset = SMC.GenerateErrorTerms
Msgbox("SMC Cal Complete!")

```

Last Updated: June 13, 2006

## Create and Cal a VMC Measurement

---

The following VMC calibration options are presented in **VB Script** examples:

- [Full 2-port ECAL](#)
- [For Mixer Characterization ONLY](#)
- [Mechanical Calibration](#)

**Note:** Each of the following programs load a mixer setup file.  
Substitute "Mixer009.mxr" with your own .mxr file.

### Full 2-port ECAL

```
Dim App
Set App = CreateObject("AgilentPNA835x.Application")
Dim Meas
Set Meas = App.ActiveMeasurement
Meas.Delete

App.CreateCustomMeasurementEx ( 2, "Vector Mixer/Converter", "VC21" )
'Other valid strings that can be specified to create a measurement with a parameter
other than "VC21" are: "S11", and "S22"
Set Meas = App.ActiveMeasurement

Meas.LoadFile "C://Program Files//Agilent//Network
Analyzer//Documents//mixer//Mixer009.mxr"

Dim CalMgr
Set CalMgr = App.GetCalManager
Dim VMC
Set VMC = CalMgr.CreateCustomCal("VMC")
VMC.Initialize 1, 1
VMC.Do2PortEcal = 1
VMC.Do1PortEcal = 1
VMC.ECALCharacterization(1, 1) = 0 'APC 2.4 male APC 2.4 female
VMC.ECALCharacterization(1, 0) = 0
' could be Default, Flush Thru, Unknown Thru, or Adapter Removal
VMC.ThruCalMethod = "Adapter Removal"
VMC.OmitIsolation = 1
VMC.AutoOrient = 1
VMC.EcalOrientation1Port(1) = "A1"
VMC.CharacterizeMixerOnly = 0
VMC.LoadCharFromFile = 0
' to load mixer characterization from file, specify LoadCharFromFile = 1,
' then VMC.CharFileName = "C://Program Files//Agilent//Network
' Analyzer//Documents//YourFile.s2p" (specify your own .s2P filename)

VMC.ConnectorType(1) = "APC 2.4 female"
```



```
VMC.ConnectorType(2) = "APC 2.4 female"
```

```
Dim steps  
steps = VMC.GenerateSteps  
For i = 1 To steps  
MsgBox VMC.GetStepDescription(i)  
VMC.AcquireStep i  
Next
```

```
Dim calset  
calset = VMC.GenerateErrorTerms
```

## Mixer Characterization ONLY

```
Dim App  
Set App = CreateObject("AgilentPNA835x.Application")  
Dim Meas  
Set Meas = App.ActiveMeasurement  
Meas.Delete  
app.CreateCustomMeasurementEx ( 2, "Vector Mixer/Converter", "VC21" )  
'Other valid strings that can be specified to create a measurement with a parameter  
other than "VC21" are: "S11", and "S22"
```

```
Set Meas = App.ActiveMeasurement  
Meas.LoadFile "C://Program Files//Agilent//Network  
Analyzer//Documents//mixer//Mixer009.mxr"
```

```
Dim CalMgr  
Set CalMgr = App.GetCalManager  
Dim VMC  
Set VMC = CalMgr.CreateCustomCal("VectorMixerCal.VMCType")  
VMC.Initialize 1, 1  
VMC.Do2PortEcal = 1  
VMC.Do1PortEcal = 1  
VMC.ECALCharacterization(1, 1) = 0  
VMC.ECALCharacterization(1, 0) = 0  
VMC.ThruCalMethod = "Default"  
VMC.OmitIsolation = 1  
VMC.AutoOrient = 1  
VMC.EcalOrientation1Port(1) = "A1"  
VMC.CharacterizeMixerOnly = 1  
VMC.LoadCharFromFile = 0  
VMC.CharFileName = "C://Program Files//Agilent//Network  
Analyzer//Documents//MyMixerChar.s2p"
```

```
Dim steps  
steps = VMC.GenerateSteps  
For i = 1 To steps  
MsgBox VMC.GetStepDescription(i)  
VMC.AcquireStep i  
Next
```

```

Dim calset
calset = VMC.GenerateErrorTerms
MsgBox VMC.CharFileName

```

## Mechanical Calibration

```

Dim App
Set App = CreateObject("AgilentPNA835x.Application")
Dim Meas
Set Meas = App.ActiveMeasurement
Meas.Delete
app.CreateCustomMeasurementEx ( 2, "Vector Mixer/Converter", "VC21" )
'Other valid strings that can be specified to create a measurement with a parameter
other than "VC21" are: "S11", and "S22"

Set Meas = App.ActiveMeasurement
Meas.LoadFile "C://Program Files//Agilent//Network
Analyzer//Documents//mixer//Mixer009.mxr"
Dim CalMgr
Set CalMgr = App.GetCalManager
Dim VMC
Set VMC = CalMgr.CreateCustomCal("VectorMixerCal.VMCType")
VMC.Initialize 1, 1
VMC.Do2PortEcal = 1
VMC.Do1PortEcal = 0
VMC.ECALCharacterization(1, 1) = 0
VMC.ThruCalMethod = "Default"
VMC.OmitIsolation = 1
VMC.AutoOrient = 1
VMC.CharacterizeMixerOnly = 0
VMC.LoadCharFromFile = 0
VMC.ConnectorType(1) = "APC 3.5 female"
VMC.ConnectorType(2) = "APC 3.5 male"
VMC.CalKitType(1) = "85033D/E"
VMC.CalKitType(2) = "85033D/E"

Dim steps
steps = VMC.GenerateSteps
For i = 1 To steps
MsgBox VMC.GetStepDescription(i)
VMC.AcquireStep i
Next

Dim calset
calset = VMC.GenerateErrorTerms

```

## Create an SMC Fixed Output Measurement with COM

---

This VBScript example creates a calibrated SMC fixed output measurement using a controlled LO. Then a single sweep is taken and data is retrieved.

Fixed output measurements are only supported on SMC (not VMC) measurements. Fixed output measurements require that an external LO source be sweeping and synchronized with the PNA source. FCA can perform this synchronization using the external source configuration settings. The fastest, and recommended, method of controlling the LO source is [Hardware List \(BNC\) triggering mode](#). However, in this mode, FCA channels will not respond to manual triggers. Therefore, the following example uses the "Single" [IChannel::Single](#) mechanism to trigger a sweep.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as SMC.vbs. [Learn how to setup and run the macro](#).

```
option explicit
dim app
set app = createobject("agilentpna835x.application")
app.preset

' Put the channel in hold (highly recommended)
app.ActiveChannel.Hold 1

' Delete the standard measurement
app.ActiveMeasurement.Delete

' Create an SC21 measurement
app.CreateCustomMeasurementEx 1, "Scalar Mixer/Converter", "SC21"

' Set the number of points to 11
app.ActiveChannel.NumberOfPoints = 11

' Setup the mixer parameters for a swept LO, fixed output measurement

dim mixer
set mixer = app.ActiveMeasurement
mixer.InputStartFrequency = 200e6
mixer.InputStopFrequency = 700e6
mixer.LORangeMode(1) = 0 ' 0 = Swept mode
mixer.OutputFixedFrequency = 3.4e9
mixer.InputPower = -17
mixer.LOPower(1) = 10

' The CALC method calculates the LO frequency from the other parameters,
' It also applies ALL mixer parameters to the channel.
mixer.Calculate 3 ' Calculate the LO range

' Create an S11 in the same channel
app.CreateCustomMeasurementEx 1, "Scalar Mixer/Converter", "S11"
dim S11Meas
set S11Meas = app.ActiveMeasurement
```

```
' Create an IPwr in the same channel
app.CreateCustomMeasurementEx 1, "Scalar Mixer/Converter","S22"

' Create an OPwr in the same channel
app.CreateCustomMeasurementEx 1, "Scalar Mixer/Converter","OPwr"

' Perform a single sweep synchronously.
app.ActiveChannel.Single 1
function ToString(complexDataArray)
dim dataAsString
dim point
for point = 0 to UBound(data)
dataAsString = dataAsString & "(" & data(point,0) & "," & data(point,1) & ") "
next
ToString = dataAsString
end function

' Retrieve the SC21 data
dim data
'Get the calibrated real/imaginary values
data = mixer.GetData(1,3)
wscript.echo ToString(data)

' Retrieve the S11 data
'Get the calibrated real/imaginary values
data = S11Meas.GetData(1,3)
wscript.echo ToString(data)
```

---

Last Modified:

June 13, 2006    Last updated

## Create a Pulsed Measurement

**Note:** This example applies only to the E836x Opt H11. For PNA-X see [PNA-X Create a Narrowband Pulsed Measurement](#).

The following example demonstrates how to create a narrowband pulsed measurement using the Pulsed Application DLL. It first gets valid configuration settings and then uses those settings to configure the PNA and external pulsed generators.

To run this program, you need:

- [Pulsed Application](#) (Option H08)
- External Pulse Generators
- External Pulse Modulator / Pulse Bias

### See Also

- Learn how to [install and register the pulsed .dll](#) on your PC
- See the [ConfigureNarrowBand3](#) Method for sending and returning parameters to the .dll.
- See the documentation for the following [COM IF Configuration](#) commands.
- See the [SCPI IF Configuration](#) commands.
- Learn about the [Pulsed Application](#).

#### 'Interfaces

```
Dim OApp As AgilentPNA835x.Application
Dim OIF As AgilentPNA835x.IFConfiguration
```

#### 'Pulsed parameters

```
Dim DPRF As Double
Dim DOffset As Double
Dim DSampleRate As Double
Dim LNumTaps As Long
Dim LBW As Long
Dim IPrecision as Integer
Dim BPG81110 As Boolean
Dim BFixedPRF As Boolean
```

#### 'pulsed DLL interface

```
Dim OPulsed As New AgilentPNAPulsed.Application
DPRF=5123 'Hz
LBW=100 'Hz
BPG81110=True 'Using the Agilent 81110A Pulse Generator
BFixedPRF=True 'Do not change the PRF during filter alignment. Only adjust the IFBW.
```

'Calculate precision of pulse generators so that the config function returns the correct precision with the right filter. For example, DPRF=5000 Hz with a pulse

generator that will only take a total of four numeric values

```
' (5.123 kHz)
' ->log10(DPRF)=3.709
' ->int(3.709)=3
' ->3-3=0
```

'The algorithm will use a  $10^x$  value for decrementing the PRF for null computation. This means that the first numeric digit from the right should be the one that is decremented by the pulsed algorithm (i.e. 5.122 kHz) to compute the filter nulls. This ensures that the pulse generators receive a PRF that is within their precision with the associated nulling IFBW.

```
IPrecision = Int(Flog10(CSng(DPRF))) - 3
```

'Send desired pulsed parameters to the pulsed configuration DLL. The DLL will return a new set of pulse parameters that provide the proper filter nulling.

```
OPulsed.ConfigNarrowBand3 DPRF, LNumTaps, LBW, DOffset, DSampleRate, IPrecision,
BFixedPRF, BPG81110
```

'Send configuration to PNA

'Connect to the PNA application

```
Set OApp = CreateObject("AgilentPNA835x.Application")
Set OIF = OApp.ActiveChannel.IFConfiguration
```

'Set PNA IFBW close to that returned by pulsed algorithm. This ensures that the proper settling time is set on the PNA.

```
OApp.ActiveChannel.IFBandwidth = LBW
OIF.IFFilterSamplePeriodMode = naMANUAL
OIF.IFFilterSamplePeriod = DSampleRate
OIF.IFFilterSource = naIFFilterSourceManual
OIF.IFFilterSampleCount = LNumTaps
OIF.IFGateEnable = True
OApp.ActiveChannel.FrequencyOffsetState = naON
OApp.ActiveChannel.FrequencyOffsetFrequency = DOffset
```

'Set receivers to medium gain setting

```
OIF.IFGainMode("ALL") = naMANUAL
OIF.IFGainLevel("A") = 1
OIF.IFGainLevel("B") = 1
OIF.IFGainLevel("R1") = 1
OIF.IFGainLevel("R2") = 1
```

```
Public Function Flog10(SGNum As Single) As Single
Flog10 = Log(SGNum) / Log(10)
End Function
```

'Enter Code here to send configuration to external pulse generators

## Create a Balanced Measurement using COM

---

The following program creates several Balanced measurements in separate windows, generates markers, calculates statistics, and sets limit lines and queries results.

**Note:** By their nature, balanced measurements are extremely sensitive to phase differences between the two RF paths that make up the balanced port, especially at higher frequencies. A good calibration (not performed in this example) is critical to achieving good balanced measurement results.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as BalancedCOM.vbs. [Learn how to setup and run the macro.](#)

```
' PNA application object
Dim app

' Channel 1 object
Dim chan1

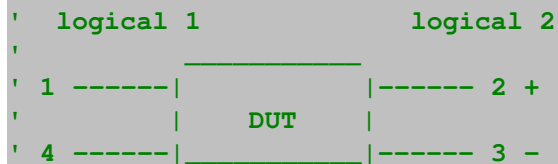
' start of marker/limit testing range
Dim minTestStimulus

' end of marker/limit testing range
Dim maxTestStimulus

' Set to true if you want additional balanced measurements.
Dim AdditionalMeasurements
AdditionalMeasurements = 1

' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
' Preset the instrument
app.Preset
' Get the Channel 1 object
Set chan1 = app.Channels(1)
' Stop data taking for now.
chan1.Hold true
' Set up the start / stop frequency for Channel 1 sweep.
MHZ = 1000000
GHZ = 1000*MHZ
chan1.StartFrequency = 10 *MHZ
chan1.StopFrequency = 1 *GHZ
chan1.NumberOfPoints = 801
' Define our test frequency range
minTestStimulus = 100*MHZ
maxTestStimulus = 900*MHZ
' This example uses DUT topology Bal-Bal -
' a DUT with a balanced input and balanced output.
'
' Port mapping for our DUT:
' logical port 1 = physical ports 1 and 4
```

```
' logical port 2 = physical ports 2 and 3
' The default is:
' logical port 1 = physical ports 1 and 2
' logical port 2 = physical ports 3 and 4
'
```



```
chan1.BalancedTopology.SetBBPorts 1, 4, 2, 3
```

```
' Now we create some Bal-Bal measurements.
' By creating Bal-Bal measurements ("BBAL:..."),
' the channel is set to Bal-Bal topology,
' so it is not necessary to do this explicitly
' with the BalancedTopology.DUTTopology command.
' We do it here just for clarity:
```

```
chan1.BalancedTopology.DUTTopology = 2
' 0 == SE-Bal, 1 == SE-SE-Bal, 2 == Bal-Bal
```

```
' Create four windows, each showing one category of balanced measurement:
' Create Forward Transmission Measurements in Bal-Bal topology on Channel 1, window
1
```

```
' differential mode transmission
```

```
app.CreateMeasurement 1, "BBAL:SDD21",1,1
Set sdd21_1 = app.ActiveMeasurement
```

```
' differential to common mode conversion
```

```
app.CreateMeasurement 1, "BBAL:SCD21",1,1
Set scd21_1 = app.ActiveMeasurement
```

```
' common to differential mode conversion
```

```
app.CreateMeasurement 1, "BBAL:SDC21",1,1
Set sdc21_1 = app.ActiveMeasurement
```

```
' common mode transmission
```

```
app.CreateMeasurement 1, "BBAL:SCC21",1,1
Set scc21_1 = app.ActiveMeasurement
```

```
' Optionally create some additional measurements
```

```
If AdditionalMeasurements Then
```

```
' Create (logical) Port 1 reflection measurements, channel 1, window 2
```

```
app.CreateMeasurement 1, "BBAL:SDD11",1,2 ' differential mode reflection
app.CreateMeasurement 1, "BBAL:SDC11",1,2 ' C to D mode conversion reflection
app.CreateMeasurement 1, "BBAL:SCD11",1,2 ' D to C mode conversion reflection
app.CreateMeasurement 1, "BBAL:SCC11",1,2 ' common mode reflection
```



```

' Create Reverse Transmission Measurements, channel 1, window 3
app.CreateMeasurement 1, "BBAL:SDD12",1,3 ' differential mode transmission
app.CreateMeasurement 1, "BBAL:SCD12",1,3 ' differential to common mode conversion
app.CreateMeasurement 1, "BBAL:SDC12",1,3 ' common to differential mode conversion
app.CreateMeasurement 1, "BBAL:SCC12",1,3 ' common mode transmission

' Create (logical) Port 2 reflection measurements in window 4
app.CreateMeasurement 1, "BBAL:SDD22",1,4 ' differential mode reflection
app.CreateMeasurement 1, "BBAL:SDC22",1,4 ' C to D mode conversion reflection
app.CreateMeasurement 1, "BBAL:SCD22",1,4 ' D to C mode conversion reflection
app.CreateMeasurement 1, "BBAL:SCC22",1,4 ' common mode reflection
End If

' Set up some limit lines to verify a minimum differential insertion loss
sdd21_1.LimitTest(1).BeginStimulus = minTestStimulus
sdd21_1.LimitTest(1).EndStimulus   = maxTestStimulus
sdd21_1.LimitTest(1).BeginResponse = -2
sdd21_1.LimitTest(1).EndResponse   = -2
sdd21_1.LimitTest(1).Type = 2 ' minimum limit
sdd21_1.LimitTest.State = 1

' Limit lines for maximum common mode to differential conversion
sdc21_1.LimitTest(1).BeginStimulus = minTestStimulus
sdc21_1.LimitTest(1).EndStimulus   = maxTestStimulus
sdc21_1.LimitTest(1).BeginResponse = -20
sdc21_1.LimitTest(1).EndResponse   = -20
sdc21_1.LimitTest(1).Type = 1 ' maximum limit
sdc21_1.LimitTest.State = 1

' Take a (synchronous) single sweep on channel 1
chan1.Single true

' Show differential forward transmission statistics.
sdd21_1.ShowStatistics = true

' Set up user range 1 to limit marker's search range.
chan1.UserRangeMin(0,1) = minTestStimulus
chan1.UserRangeMax(0,1) = maxTestStimulus

' Find/Show max common mode to differential conversion, and read back the frequency.
sdc21_1.MarkerState(1) = true
' Set marker 1 to use user range 1
sdc21_1.Marker(1).UserRange = 1
sdc21_1.Marker(1).SearchMax

' Find/Show max differential mode insertion loss, and read back the frequency.
sdd21_1.MarkerState(1) = true
' Set marker 1 to use user range 1
sdd21_1.Marker(1).UserRange = 1
sdd21_1.Marker(1).SearchMin
If sdd21_1.LimitTestFailed Then
Wscript.Echo "Differential insertion loss failed: " & sdd21_1.Marker(1).Stimulus
/MHz & "MHz, " & _sdd21_1.Marker(1).Value(1) & " dB"
End If

```

```
If sdc21_1.LimitTestFailed Then
Wscript.Echo "Common to differential conversion failed: " &
sdc21_1.Marker(1).Stimulus/MHZ & "MHz, " & _sdc21_1.Marker(1).Value(1) & " dB"
End If
```

## Create a PMAR Device and Measurement

---

The following program creates a new External Device: a Power Meter as Receiver, makes several power meter settings, and then creates a PMAR measurement.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as PMAR.vbs. [Learn how to setup and run the macro.](#)

### See Also

[ExternalDevices Collection](#)

[ExternalDevice Object](#)

[PowerSensorAsReceiver Object](#)

[PowerSensorCalFactorSegmentPMAR Object](#)

[PowerLossSegmentsPMAR\\_ Collection](#)

[PowerLossSegmentPMAR Object](#)

```
dim app
Set app = CreateObject("AgilentPNA835x.Application")
dim externalDevices
Set externalDevices = app.ExternalDevices
dim devicecount
devicecount = externalDevices.count
externalDevices.Add "NewPMAR"
dim newExternalDevice
Set newExternalDevice = externalDevices.Item("NewPMAR")
newExternalDevice.DeviceType = "Power Meter"
newExternalDevice.IOConfiguration= "GPIB0::14::INSTR"
'newExternalDevice.IOConfiguration = "USB0::2391::4865::GB45100278::0::INSTR"
newExternalDevice.IOEnable = true
dim PMAR
Set PMAR = newExternalDevice.ExtendedProperties
PMAR.SensorIndex = 1
PMAR.ReadingsPerPoint = 10
dim avr
avr = PMAR.ReadingsPerPoint
PMAR.ReadingsTolerance = 0.1
dim tole
```

```
tole = PMAR.ReadingsTolerance
PMAR.MinimumFrequency = 100000000
PMAR.MaximumFrequency = 10000000000
PMAR.LimitFrequency = false
PMAR.referenceCalFactor = 99
Set powerCalFactorSegments = PMAR.CalFactorSegments
powerCalFactorSegments.Add 1,10
Set calpair = powerCalFactorSegments(1)
calpair.Frequency = 1e9
calpair.CalFactor = 99
Set calpair = powerCalFactorSegments(2)
calpair.Frequency = 2e9
calpair.CalFactor = 98
powerCalFactorSegments.Remove 3,8
PMAR.UsePowerLossSegments = true
Set pls = PMAR.PowerLossSegments
pls.Add 1,5
Set pl = pls(1)
pl.Loss = -1
pl.Frequency = 1e9
Set pl = pls(2)
pl.Loss = -2
pl.Frequency = 2e9
pls.Remove 3,3
newExternalDevice.active = true
'Create a PMAR trace with power meter connected to port 3
app.CreateMeasurement 1,"NewPMAR",3,1
```

Last Modified:

3-Sep-2009    MX New topic

## Create a Wideband Pulsed Measurement using the PNA-X

This Visual Basic COM example shows you how to configure the PNA-X internal pulse generators and modulators to make wideband pulsed measurements in **pulse profile** mode using the PNA-X.

[Visit the PNA website](#) where you can download a free Wideband Pulsed Application that performs this measurement on the PNA-X.

[See all COM Pulsed examples](#)

```
'Create an PNA Application instance
Dim pnaApp As New AgilentPNA835x.Application
'Create a PathConfiguration instance
Dim pathConf As AgilentPNA835x.PathConfiguration
'Create an PulseGenerator instance
Dim pulseGen As AgilentPNA835x.PulseGenerator
'Create a Channel instance
Dim myChan As AgilentPNA835x.Channel

'Preset PNA-X
pnaApp.Preset
'Assign current active channel to myChan object
Set myChan = pnaApp.ActiveChannel
'Let PNA-X work in CW mode because of doing pulse profile measurement
myChan.SweepType = naCWTimeSweep
'Set CW Freq to 4 GHz
myChan.CWFrequency = 4000000000#
'Set IF Bandwidth to 5 MHz to get the best time resolution
myChan.IFBandwidth = 5000000#
'Assign current active channel path configuration to pathConf object
Set pathConf = myChan.PathConfiguration
'Let PNA-X source work in ALC Open Loop mode
myChan.ALCLevelingMode(1) = naALCOpenLoop
'Make the Pulse1 as modulation pulse to generate Pulsed-RF signal
pathConf.Element("PulseModDrive").Value = "Pulse1"
'Enable pulse modulation at Source1Out1 path
pathConf.Element("Src1Out1PulseModEnable").Value = "Enable"
'Assign current active channel pulse generator to pulseGen object
Set pulseGen = myChan.PulseGenerator
```

```
'Internal pulse generator has five channels,  
'default the channel 0 use as internal ADC trigger signal  
'Enable channel 0 of internal pulse generator as trigger signal  
pulseGen.State(0) = True  
'Enable channel 1 of internal pulse generator as modulation signal  
pulseGen.State(1) = True  
'Set pulse period to 10 us  
pulseGen.Period = 0.00001 '10 us  
'Set pulse width of channel 0 to 1 us  
pulseGen.Width(0) = 0.000001 ' 1 us  
'Set pulse width of channel 1 to 5 us  
pulseGen.Width(1) = 0.000005 '5 us
```

**End Sub**

---

Last Modified:

4-Jan-2008    Added point trigger note

2-Oct-2007    MX New topic

## Create an IM Spectrum Measurement

---

This VBScript example creates IM Spectrum measurement based on passed parameters.

This subprogram is extracted from the macro on the PNA that produces an IM spectrum channel from the Marker function. You can see the entire program at C:/Program Files/Agilent/Network Analyzer/Applications/IMD/IMD.VBS".

This VBScript (\*.vbs) program must be used as part of a program that supplies the required parameters. When complete, it can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as IMD.vbs.

[Learn how to setup and run the macro.](#)

[See SweptIMD Object.](#)

```
' ' SetupIMSpectrum
' ' Setup an IM Spectrum (non-converter) channel based upon supplied parameters
sub SetupIMSpectrum(app, MkrPos, ToneSpacing, TonePower)
    dim objIMXChan, objIMDChan
    dim Fstart, Fstop, NumPoints, ToneFc

    set objIMXChan = objIMChan.CustomChannelConfiguration
    set objIMDChan = objSIMDChan.CustomChannelConfiguration

    NumPoints = objSIMDChan.NumberOfPoints
    select case objIMDChan.SweepType

    case naIMDToneCWSweep
        ToneFc = objIMDChan.FrequencyCenter

    case naIMDTonePowerSweep
        Fstart = objIMDChan.TonePowerStart(0)
        Fstop = objIMDChan.TonePowerStop(0)
        TonePower = CalcMkrValue(Fstart, Fstop, MkrPos, NumPoints)
        ToneFc = objIMDChan.FrequencyCenter

    case naIMDToneCenterFreqSweep
        Fstart = objIMDChan.FrequencyCenterStart
```

```
Fstop = objIMDChan.FrequencyCenterStop
ToneFc = CalcMkrValue(Fstart, Fstop, MkrPos, NumPoints)

case naIMDDeltaFrequencySweep
    ToneFc = objIMDChan.FrequencyCenter
    Fstart = objIMDChan.DeltaFrequencyStart
    Fstop = objIMDChan.DeltaFrequencyStop
    ToneSpacing = CalcMkrValue(Fstart, Fstop, MkrPos, NumPoints)

case naIMDToneSegmentSweep
    ToneFc = MarkerXValue

end Select
objIMXChan.FrequencyCenter = ToneFc
objIMXChan.DeltaFrequency = ToneSpacing
objIMXChan.TonePower(0) = TonePower
objIMXChan.TonePower(1) = TonePower
app.ActiveMeasurement.Trace.ReferenceValue = TonePower + 10
objIMSChan.continuous
end Sub
```

---

Last Modified:

31-Mar-2009    MX New topic



## Create an iTMSA Measurement

---

The following VB Script example shows how to create an iTMSA measurement with Power Sweep. Click each link to see a detailed description of each command.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as Testset.vbs. Learn how to setup and run the macro.

```
dim app      'PNA App
dim meas     'Measurement
dim balancemeas
dim balstimulus
dim chan

set app = createobject("agilentpna835x.application")
set chan = app.activechannel
chan.SweepType = 2 ' Set the sweep type to power sweep

set meas = app.ActiveMeasurement
set balancemeas = meas.BalancedMeasurement
balancemeas.BalancedTopology.DUTTopology = 2 ' Bal-Bal topology
balancemeas.BalancedStimulus.Mode = 1 ' Turn on true mode
'The PNA-X balanced port numbers are always (0)=Bal 1; (-1)=Bal2
chan.StartPowerEx(0) = -5 ' Set the balanced port 1 start power to -5 dbm
chan.StopPowerEx(0) = 5 ' Set the balanced port 1 stop power to 5 dbm
chan.StartPowerEx(-1) = -10 ' Set the balanced port 2 start power to -5 dbm
chan.StopPowerEx(-1) = 0 ' Set the balanced port 2 stop power to 5 dbm
```

---

Last Modified:

2-Jun-2008    MX New topic

## Create and Cal a Gain Compression Measurement

---

This VBScript example creates and calibrates a Gain Compression measurement and performs Compression analysis.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as GCA.vbs. Learn how to setup and run the macro .

### See Gain Compression Object.

```
option explicit
```

```
dim CompLevel , Tolerance , StartFreq , StopFreq , NumFreqs , Scale , LinearPower
dim AcqMode , BackOff , StartPower , StopPower , NumPowers , EnableInterp , CompAlg
dim DwellTime , IFBandwidth , ShowIterations , host , app

' GCA Settings/Values
''
' Acquisition Mode:
' naSmartSweep = 0
' naSweepPowerAtEachFreq2D = 1
' naSweepFreqAtEachPower2D = 2
''
' Compression Algorithm
' naCompressionFromLinearGain = 0
' naCompressionFromMaximumGain = 1
' naBackoffCompression = 2
' naXYCompression = 3
''
' EndOfSweepOperation
' naDefaultPowerSet = 0
' naSetToStartPower = 1
' naSetToStopPower = 2
' naSetRFOff = 3
''

CompLevel      = 1      ' 1 dB compression level
Tolerance      = 0.05   ' SMART Sweep tolerance
```

```

StartFreq      = 1E9
StopFreq       = 9E9
NumFreqs       = 201
Scale          = 0.1
LinearPower     = -20
BackOff        = 10      ' Not used for Deviation from linear gain
StartPower     = -20
StopPower      = 8
NumPowers      = 60      ' Not used for SMART Sweep
DwellTime      = 0.0005 ' Allow some time for DUT bias/thermal effects
IFBandwidth    = 1000   ' Reasonable trace noise at -20 dBm
EnableInterp   = False   ' Disable interpolation
AcqMode        = 0       ' Smart Sweep
CompAlg        = 0       ' Deviation from linear gain
ShowIterations = False   ' Configure SMART to not show iteration results

```

```

dim objargs
set objargs = wscript .Arguments
if (objArgs .Count = 1) then host = objargs (0)

```

```

' .....
' Create and Configuration GCA Channel:
' .....

```

```

set app = CreateObject ("Agilentpna835x.application" )
call SetupGCA ( app ,_
    StartFreq ,_
    StopFreq ,_
    NumFreqs ,_
    EnableInterp ,_
    Scale ,_
    CompLevel ,_
    LinearPower ,_
    AcqMode ,_
    BackOff ,_
    StartPower ,_
    StopPower ,_

```

```

        NumPowers , _
        CompAlg , _
        DwellTime , _
        IFBAndwidth , _
        ShowIterations )

call CalGCA ( app )
call Analysis ( app )

.....

'' GCA Setup
.....

sub SetupGCA ( app , StartFreq , StopFreq , NumFreqs , EnableInterp , Scale ,
CompLevel , LinearPower , _
        AcqMode , BackOff , StartPower , StopPower , NumPowers , CompAlg ,
DwellTime , IFBAndwidth , _
        ShowIterations )

dim chan , gca
app .reset
app .CreateCustomMeasurementEx 1, "Gain Compression" , "S21" , 1
set chan = app .channels (1)
chan .hold 1
app .CreateCustomMeasurementEx 1, "Gain Compression" , "CompIn21" , 1
app .CreateCustomMeasurementEx 1, "Gain Compression" , "DeltaGain21" , 1
app .nawindows (1).traces (3).YScale = Scale
app .nawindows (1).traces (3).ReferenceValue = -CompLevel
set gca = chan .CustomChannelConfiguration
gca .InputLinearPowerLevel = LinearPower
gca .AcquisitionMode = AcqMode
gca .CompressionLevel = CompLevel
gca .CompressionBackoff = BackOff
gca .CompressionDeltaX = BackOff
gca .CompressionDeltaY = BackOff - CompLevel
gca .CompressionAlgorithm = CompAlg
gca .NumberOfPowerPoints = NumPowers
gca .CompressionInterpolation = EnableInterp
gca .SmartSweepSettlingTime = DwellTime
gca .SmartSweepShowIterations = ShowIterations
chan .IFBAndwidth = IFBAndwidth

```

```

chan .DwellTime = DwellTime
chan .StartPower = StartPower
chan .StopPower = StopPower
chan .StartFrequency = StartFreq
chan .StopFrequency = StopFreq
chan .NumberOfPoints = NumFreqs
chan .single 1
end sub

.....

' GCA Calibration
.....

sub CalGCA ( app )

    dim chan , CalMgr , GCACal , GCACustomCal , CalSteps , I , CalSet
    set chan = app .ActiveChannel
    set CalMgr = app .GetCalManager
    set GCACal = CalMgr .CreateCustomCalEx (1)
    ' Configure GCA Guided Cal for the connector types and ECal module that will be
    used:
    GCACal .Initialize 1, 0
    GCACal .ConnectorType (1) = "APC 3.5 female"
    GCACal .ConnectorType (2) = "APC 3.5 male"
    GCACal .CalKitType (1) = "N4691-60004 ECal"
    GCACal .CalKitType (2) = "N4691-60004 ECal"
    set GCACustomCal = GCACal .CustomCalConfiguration
    GCACustomCal .PowerLevel = 0      ' Set power level for source cal to 0 dBm
    CalSteps = GCACal .GenerateSteps
    for I = 1 to CalSteps
        msgBox GCACal .GetStepDescription (I )
        GCACal .AcquireStep (I )
    next
    CalSet = GCACal .GenerateErrorTerms ' Calculate error terms and apply CalSet to GCA
    Channel
    chan .CalSet .Save ("GCA 2P" ) ' Save CalSet
    msgBox "Done"
end sub

```

```

.....
'' GCA Analysis (PNA Rev A.09.00 or later)
.....

sub Analysis (app)
Dim meass
Dim ana
Set meass = app.Measurements ' get the measurements
Set ana = meass(1).CustomMeasurementConfiguration 'get the measurement
ana.AnalysisEnable = true ' enable the analysis mode
ana.AnalysisCWFreq = 3e9 ' set the analysis cw frequency to 3GHz
Set ana = meass(2).CustomMeasurementConfiguration
ana.AnalysisEnable = true
ana.AnalysisCWFreq = 4e9
ana.AnalysisXAxis = naPsourceAsXAxis ' set the XAxis as the source power setting
Set ana = meass(3).CustomMeasurementConfiguration
ana.AnalysisEnable = true
ana.AnalysisIsDiscreteFreq = false ' turn off the discrete frequency option
ana.AnalysisCWFreq = 4.5e9
end sub

```

---

**Last Modified:**

3-Sep-2009    Added GCA analysis

17-Dec-2007    MX New topic

## Create and Cal a Noise Figure Measurement

---

This VBScript example creates a Noise figure measurement, then calibrates the measurement.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Noise.vbs. [Learn how to setup and run the macro.](#)

### Bonus: Accessing the NoiseFigure object and NoiseCal object using C#

Replace **<hostname>** with the full computer name of your PNA

```
Type pna = Type.GetTypeFromProgID("AgilentPNA835x.Application", "<hostname>");
    AgilentPNA835x.Application app =
(AgilentPNA835x.Application)Activator.CreateInstance(pna);

    app.Reset();

    app.CreateCustomMeasurementEx(1, "NoiseFigure", "NF", 1);

    AgilentPNA835x.ICalManager5 calManager =
(AgilentPNA835x.ICalManager5)app.GetCalManager();

    AgilentPNA835x.IGuidedCalibration4 guidedCal4 =
(AgilentPNA835x.IGuidedCalibration4)
        calManager.CreateCustomCalEx(1);

    AgilentPNA835x.INoiseCal noiseCal =
(AgilentPNA835x.INoiseCal)guidedCal4.CustomCalConfiguration;
```

[See NoiseFigure Object.](#)

[See other COM Examples](#)

```
windowNum = 1
channelNum = 1
set pna=CreateObject("AgilentPNA835x.Application", hostname)
set scpi = pna.ScpiStringParser
pna.reset
' Create noise figure measurement
set noise = pna.createcustommeasurementex(channelNum, "NoiseFigure", "NF",
windowNum)
set noisechan = pna.activechannel
' Create object to access noise-specific channel attributes
set noiseConfig = pna.activechannel.CustomChannelConfiguration
' Create guided noise calibration object on our channel
set noisecal = pna.GetCalmanager.CreateCustomCalEx(channelNum)
```

```

set noiseCalExtension = noisecal.CustomCalConfiguration
noiseCalExtension.NoiseSourceCold = 300
' Substitute appropriate Ecal identification strings here
tunerEcal = "N4691-60004 ECal 02821"
pullEcal = "N4691-60004 ECal 02297"
' configuration
ConfigureChannel
ConfigureNoiseSettings
' perform calibration
SetupCalAttributes_Insertable
SetupNoiseSource
FinishCalibration
' ----- Support subroutines -----
' Configure noise channel
sub ConfigureChannel
    noisechan.startfrequency = 500e6
    noisechan.stopfrequency = 5.0e9
    noisechan.numberofpoints = 201
    noisechan.IFBandwidth = 1.0E3
end sub
' Configure noise-specific channel settings
sub ConfigureNoiseSettings
    noiseConfig.noiseaveragestate = true
    noiseConfig.NoiseAverageFactor = 40
    noiseConfig.NoiseTuner = tunerEcal
    noiseConfig.NoiseTunerIn = "B"
    noiseConfig.NoiseTunerOut = "A"
    noiseConfig.NoiseBandwidth = 8e6
end sub
sub SetupCalAttributes_Insertable
    noisecal.Initialize channelNum, true
    noisecal.ConnectorType( 1 ) = "APC 3.5 female"
    noisecal.ConnectorType( 2 ) = "APC 3.5 male"

    noisecal.CalKitType (1) = pullEcal
    noisecal.CalKitType (2) = pullEcal

```



```

noiseCalExtension.NoiseSourceConnectorType = "APC 3.5 male"
noiseCalExtension.NoiseSourceCalKitType = pullEcal
noiseCalExtension.CalMethod = "Vector"
end sub
sub SetupNoiseSource
' specify the ENR file for the noise source
noiseCalExtension.ENRFile = "C:/Program Files/Agilent/Network
Analyzer/Documents/346C_MY44420454.enr"
noiseCalExtension.NoiseSourceCold = 301.1
end sub
' Build the connection list and acquire the calibration
sub FinishCalibration
steps = noisecal.GenerateSteps
for i = 1 to steps
str = noisecal.GetStepDescription( i )
msgbox str
noisecal.AcquireStep i
next
guid = noisecal.generateerrorterms
wscript.echo "Calibration created calset guid: ",guid
end sub

```

---

Last Modified:

14-May-2010 Fixed hostname

9-Nov-2007 MX New topic

## Create and Cal a Swept IMD Measurement

---

This VBScript example creates IMD power and IM3 measurements, sets sweep mode to Center Frequency Sweep, and performs an IMD cal.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as IMD.vbs.

You can see the VB Script program on the PNA that produces an IM spectrum channel from the Marker function at : C:/Program Files/Agilent/Network Analyzer/Applications/IMD/IMD.VBS".

[Learn how to setup and run the macro.](#)

[See SweptIMD Object.](#)

```
option explicit
'declare variables
dim SweepMode, StartDeltaFreq, StopDeltaFreq, NumFreqs, TonePower, CWFreq
dim app, hostname
' Sweep type:
' naIMDToneCWSweep          = 0
' naIMDTonePowerSweep       = 1
' naIMDToneCenterFreqSweep = 2
' naIMDDeltaFrequencySweep = 3
' naIMDToneSegmentSweep    = 4
'init variables
SweepMode    = 3          ' Sweep DeltaF
StartDeltaFreq = 100e3
StopDeltaFreq  = 1e9
NumFreqs      = 201
TonePower     = -7
CWFreq        = 5e9
' get host name from commandline
dim objargs
set objargs = wscript.arguments
if(objargs.Count = 1) then hostname = objargs(0)
set app = CreateObject("Agilentpna835x.application", hostname)
call SetupIMD
call CalIMD
```

```

.....

'' Create and Configure IMD channel
.....

sub SetupIMD
    dim chan, imd
    app.reset
    ' Create IMD measurements
    app.CreateCustomMeasurementEx 1, "Swept IMD", "PwrMain", 1
    app.CreateCustomMeasurementEx 1, "Swept IMD", "IM3", 1
    set chan = app.channels(1)
    chan.hold 1

    set imd = chan.CustomChannelConfiguration
    imd.SweepType = SweepMode
    imd.FrequencyCenter = CWFreq
    imd.DeltaFrequencyStart = StartDeltaFreq
    imd.DeltaFrequencyStop = StopDeltaFreq
    imd.TonePower(0) = TonePower 'F1 power
    imd.TonePower(1) = TonePower 'F2 power

    chan.NumberOfPoints = NumFreqs
    chan.single 1
end sub

sub CalIMD
    dim chan, CalMgr, IMDCal, IMDCustomCal, CalSteps, I, CalSet
    set chan = app.ActiveChannel
    set CalMgr = app.GetCalManager
    set IMDCal = CalMgr.CreateCustomCalEx(1)

    'Configure IMD GuidedCal for the connector types and ECal module that will be
used
    ' Substitute appropriate connector type and ECal identification strings here
    IMDCal.Initialize 1, true 'channel number is 1
    IMDCal.ConnectorType(1) = "APC 3.5 female"
    IMDCal.ConnectorType(2) = "APC 3.5 male"
    IMDCal.CalKitType(1) = "N4693-60001 User 2 ECal 00012"

```

```

IMDCal.CalKitType(2) = "N4693-60001 User 2 ECal 00012"

' IMD Custom settings
set IMDCustomCal = IMDCal.CustomCalConfiguration

' Set the Power Level at the power sensor to be used in calibration
IMDCustomCal.PowerLevel = 0

' Specify the connector type of the power sensor. If there is an adapter
between
' the input port and the power sensor, specify the connector type here, and
set
' the appropriate cal kit type for the connector so that extra calibration
can be
' performed. To skip the calibration for the adapter, set
PowerSensorConnectorType to "Ignored"
' i.e.: IMDCustomCal.PowerSensorConnectorType = "Ignored"
IMDCustomCal.PowerSensorConnectorType = "APC 3.5 female"
IMDCustomCal.PowerSensorCalKitType = "N4693-60001 User 2 ECal 00012"

' Set the Max product to calibrate, valid values are 3, 5, 7, and 9
IMDCustomCal.MaxProduct = 3

' Set the calibration Frequencies, can choose between calibrate only at
center Frequencies (0)
' or calibrate at all frequencies (1).
IMDCustomCal.CalibrationFrequencies = 1

'Include 2nd order product in calibration
IMDCustomCal.Include2ndOrderProduct = true

CalSteps = IMDCal.GenerateSteps
for I = 1 to CalSteps
    msgBox IMDCal.GetStepDescription(I)
    IMDCal.AcquireStep(I)
next
CalSet = IMDCal.GenerateErrorTerms
msgBox "IMD Cal Done"

```

end sub

---

Last Modified:

31-Mar-2009    MX New topic

## ECAL Confidence Check

---

This Visual Basic program:

- Initializes the PNA objects.
- Performs a complete ECAL confidence check

Before using this code:

- The active channel must contain an S11 measurement with a 1-port or N-port calibration
- Prepare a form with two buttons named **cmdRun** and **cmdQuit**

```
Private oPNA As AgilentPNA835x.Application
Private oChan As Channel
Private oCal As Calibrator
Private oMeas As Measurement

Private Sub cmdRun_Click()
Dim iMeasIndex As Integer

Set oPNA = CreateObject("AgilentPNA835x.Application", "MachineName")
Set oChan = oPNA.ActiveChannel
Set oCal = oChan.Calibrator

iMeasIndex = 1

' Loop through measurements until an S11 on the active channel
' is found, or the end of the measurement collection is reached.
Do
    Set oMeas = oPNA.Measurements(iMeasIndex)
    If oMeas.Parameter = "S11" And _
        oMeas.channelNumber = oChan.channelNumber Then Exit Do
    iMeasIndex = iMeasIndex + 1
    If iMeasIndex > oPNA.Measurements.Count Then
        MsgBox "No S11 measurement found on the active channel." _
            & " Create an S11 measurement, then try again."
        Exit Sub
    End If
```

Loop

```
' Set up trace view so we are viewing only the data trace.
oMeas.View = naData
' Acquire the S11 confidence check data from ECal Module A
' into the memory buffer.
oCal.AcquireCalConfidenceCheckECALEx "S11", 1
' Turn on trace math so the trace shows data divided by memory.
' You can be confident the S11 calibration is reasonably good if
' the displayed trace varies no more than a few tenths of a dB
' from 0 dB across the entire span.
oMeas.TraceMath = naDataDivMemory
End Sub
```

```
Sub cmdQuit_Click()
' Turn off trace math
' in case someone clicks Quit without having clicked Run
If oMeas <> Nothing Then oMeas.TraceMath = naDataNormal
' Conclude the confidence check to set the ECal module
' back to it's idle state.
If oCal <> Nothing Then oCal.DoneCalConfidenceCheckECAL
' End the program
End
End Sub
```

## ENR File Management Example

---

This VB Script program illustrates ENR file management using COM commands.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as Calibrate.vbs. Learn how to setup and run the macro.

[See Other COM Example Programs](#)

```
' Sample VBS program illustrating COM commands for ENR file management.

option explicit
dim pna      ' application
dim enr      ' ENRFile object
dim scpi, hostname

set pna=CreateObject("agilentpna835x.application")
set scpi = pna.ScpiStringParser
set enr = pna.ENRFile

' Generate data to put in ENR file
Dim vdata(3)
vdata(0) = 100E6      ' first frequency point
vdata(1) = 14.532     ' first ENR value
vdata(2) = 20E9       ' second frequency point
vdata(3) = 15.731     ' second ENR value

' send data to ENRFile object
enr.PutENRData(vdata)

' Set noise source serial number
enr.ENRSN = "ABCD1234"

' Write ENR file to disk
enr.SaveENRFile("C:/Program Files/Agilent/Network Analyzer/Documents/sample.enr")
```



The contents of the file created by this program are shown below.

[Filetype ENR]

[Version 1.0]

[Serialnumber ABCD1234]

!    Frequency    ENR

!        Hz        dB

100000000 14.53200

2e+010 15.73100

---

Last Modified:

14-May-2010    Fixed hostname

2-Aug-2007    MX New topic

## Events with C++

---

The following code, along with the Header file, shows how to use the PNA Events.

[Download the Header file 'preventcatcher.h'](#)

```
#include <atlbase.h>
#include <atlcom.h>
#include <iostream>
#import "835x.tlb" no_namespace,raw_interfaces_only,named_guids
#include "pnaeventcatcher.h"
```

```
inline void HR(HRESULT hr)
{
    if (FAILED(hr))
        throw hr;
}
```

```
class MyEventCatcher : public CPNAEventCatcher
{
public:
    MyEventCatcher()
    {
        CoInitialize(NULL);
        CComPtr<IApplication> app;
        HR(app.CoCreateInstance(CLSID_Application));
        CPNAEventCatcher::SubscribeCatcher(app);
        HR(app->AllowAllEvents());
    }
    ~MyEventCatcher()
    {
        CPNAEventCatcher::Release();
        CoUninitialize();
    }
}
```

```

}
virtual void OnMeasurementEvent(long eventID,long measurementNumber) {}
virtual void OnChannelEvent(long eventID,long ch)
{
    if (eventID == 0x68070709L) //MSG_ALL_SWEEPS_COMPLETED_AND_PROCESSED
    {
        static int i =0;
        ++i;
        std::cout << "Sweep:" << i << std::endl;
    }
}
};

```

-----

In a .cpp file, (just like most ATL projects) you must have a declared an instance of CComModule. This will work:

```
CComModule _Module;
```

-----

Remember that you are now the "Server" and the PNA is the Client. That makes DCOM a bit complicated.

This code was tested in VS2005 using a wizard generated MFC MDI project.

---

Last Modified:

13-Nov-2007    MX New topic

## FOM Examples

---

All three VBScript examples in this topic create a FOM measurement with the following attributes:

- Sweep the Source (input) from 1 GHz to 2 GHz
- Sweep the Receivers (output) from 2 GHz to 3 GHz
- You provide an LO at 1 GHz

### [Learn more about Frequency Offset Mode](#)

These programs can be run as a macro in the PNA. To do this, copy the code into a text editor file such as Notepad and save on the PNA hard drive as FOM.vbs. [Learn how to setup and run the macro.](#)

The following example will run on any PNA model with FOM (opt 080). however, These commands have no provisions for internal second source. It uses commands introduced before 'enhanced FOM' was released for the A.07.10 release.

```
set app = CreateObject("Agilentpna835x.application")
set chan = app.ActiveChannel
chan.startFrequency = 1e9
chan.StopFrequency = 2e9
' set the receiver frequencies to be 2e9->3e9
chan.FrequencyOffsetFrequency = 1e9
chan.FrequencyOffsetState = 1
```

The following example can be run ONLY on a PNA with revision A.07.10 or later and has FOM (opt 080). It uses new FOM commands. [See FOMRange object.](#)

```
set app = CreateObject("Agilentpna835x.application")
set chan = app.ActiveChannel
chan.startFrequency = 1e9
chan.StopFrequency = 2e9
' set the receiver frequencies to be 2e9->3e9
chan.fom("Receivers").Offset = 1e9
chan.fom.State = 1
```

The following example can be run ONLY on a PNA with a second internal source, has revision A.07.10 or later, and has FOM (opt 080). It uses the internal 2nd source for the fixed LO frequency.

```
set app = CreateObject("Agilentpna835x.application")
set chan = app.ActiveChannel
chan.startFrequency = 1e9
chan.StopFrequency = 2e9
' set the receiver frequencies to be 2e9->3e9
chan.fom("Receivers").Offset = 1e9
chan.fom("Source2").Coupled = 0
chan.fom("Source2").StartFrequency = 1e9
chan.fom("Source2").StopFrequency = 1e9
' turn off port coupling
chan.coupleports = 0
' set LO to 10 dBm
chan.TestPortPower(3) = 10
'Turn ON port 3, our LO signal on our 2 source PNA
chan.SourcePortMode(3) = 1
chan.fom.State = 1
```

---

Last Modified:

8-Oct-2007    MX New topic

## Limit Line Testing with COM

---

This Visual Basic program:

- Turns off existing Limit Lines
- Establishes Limit Lines with the following settings:
  - Frequency range - 4 GHz to 8 GHz
  - Maximum value - (10dB)
  - Minimum value - (-30dB)
- Turns on Lines, Testing, and Sound

If using [Global Pass/Fail](#) to report limit results, trigger the PNA after configuring and enabling Limit lines.

```
Public limts As LimitTest
Set limts = meas.LimitTest
'All Off
For i = 1 To 20
    limts(i).Type = naLimitSegmentType_OFF
Next i

'Set up Limit Lines
limts(1).Type = naLimitSegmentType_Maximum
limts(1).BeginResponse = 10
limts(1).EndResponse = 10
limts(1).BeginStimulus = 4000000000#
limts(1).EndStimulus = 8000000000#
limts(2).Type = naLimitSegmentType_Minimum
limts(2).BeginResponse = -30
limts(2).EndResponse = -30
limts(2).BeginStimulus = 4000000000#
limts(2).EndStimulus = 8000000000#

'Turn on Lines, Testing, and Sound
limts.LineDisplay = 1
limts.State = 1
limts.SoundOnFail = 1
```

---

## Modify Display Colors

---

This VBScript example modifies display colors, modifies trace1 colors, then saves and recalls the theme.

These programs can be run as a macro in the PNA. To do this, copy the code into a text editor file such as Notepad and save on the PNA hard drive as FOM.vbs. [Learn how to setup and run the macro.](#)

```
function RGB(R, G, B)
    RGB = R + G*(2^8) + B*(2^16)
end Function

shell.AppActivate "PNA Series Network Analyzer"
Set app = CreateObject("AgilentPNA835X.Application")
app.preset
Set colors = app.Preferences.DisplayColors
' Uncomment the following line to modify Print colors
' Set colors = app.Preferences.PrintColors
colors.ResetTheme( )
colors.background      = RGB(64,0,64)      ' purple
displaycolors.grid     = RGB(0,255,128)    ' greenish
colors.activeLabels    = RGB(0,0,255)      ' blue
colors.inactiveLabels  = RGB(255,0,0)      ' red
colors.failedTraces    = RGB(255,128,64)   ' orange
dim Trace1
Set Trace1 = colors.Trace(1)
Trace1.DataAndLimits   = RGB(1,251,1)      ' green
Trace1.Memory          = RGB(251,1,1)      ' red
Trace1.Markers         = RGB(251,251,251)   ' white
Trace1.MemoryMarkers   = RGB(1,251,251)    ' green + blue
colors.StoreTheme("c:/Program Files/Agilent/Network
Analyzer/Colors/Theme1.colors")
colors.LoadTheme("c:/Program Files/Agilent/Network
Analyzer/Colors/Theme1.colors")
```

---

Last Modified:

3-Jun-2010    MX New topic

## E5091Testset Control

---

The following VB Script example exercises the COM commands used to control the E5091A testset.

For a description of each command, see [E5091Testsets collection](#).

```
Sub Main()  
Set pna = CreateObject("AgilentPNA835x.Application")  
Dim testsets As E5091Testsets  
Set testsets = pna.E5091Testsets  
Dim tset1 As E5091Testset  
Set tset1 = testsets(1)  
tset1.OutputPort(1, 3) = naE5091PortR2  
tset1.ControlLines(1) = 5  
tset1.ShowProperties = True  
tset1.Enabled = True  
MsgBox tset1.ID  
MsgBox tset1.Enabled  
MsgBox tset1.ShowProperties  
' NumberOfPorts property returns 0 when testset not connected  
MsgBox tset1.NumberOfPorts  
MsgBox tset1.OutputPort(1, 3)  
MsgBox tset1.ControlLines(1)  
  
Dim tset2 As E5091Testset  
Set tset2 = testsets(2)  
tset2.Enabled = True  
tset2.ShowProperties = True  
MsgBox tset2.Enabled  
MsgBox tset2.ShowProperties  
End Sub
```



## Errors and the SCPIStringParser Object

---

This C++ program uses the [SCPIStringParser.Parse](#) command to detect the failed HRESULT and interrogate the `errorInfo` object for more details.

```
// scpierrors.cpp : Defines the entry point for the console application.
//

#include <iostream>
#include "afx.h"
#include "atlbase.h"

#import "C:/program files/common files/agilent/pna/835x.tlb" raw_interfaces_only,
no_namespace, named_guids

using namespace std;

HRESULT SendScpiCommand( IScpiStringParser* parser, CComBSTR& cmd, CComBSTR&
response)
{
    CComBSTR bstr;
    HRESULT hr = parser->Parse(CComBSTR(cmd), &response);
    if (FAILED(hr))
    {
        // see if this interface supports ErrInfo
        CComPtr<ISupportErrorInfo> spSupportsErrInfo;
        if (SUCCEEDED(parser->QueryInterface(&spSupportsErrInfo)))
        {
            // it does, so let's get the errorinfo object
            CComPtr<IErrorInfo> spErrorInfo;
            if (SUCCEEDED(GetErrorInfo(0, &spErrorInfo)))
            {
                CComBSTR errStr;
                spErrorInfo->GetDescription(&errStr);
                std::cout << "ERROR: " << CString(errStr) << std::endl;
            }
        }
    }
    return hr;
}
```

```

int main()
{
    CoInitialize(NULL);
    {
        CComBSTR response;
        CComPtr<IApplication> spPNA;
        CComPtr<IScpiStringParser> spSCPI;
        if (SUCCEEDED(spPNA.CoCreateInstance(CLSID_Application)))
        {
            spPNA->get_ScpiStringParser(&spSCPI);
            SendScpiCommand(spSCPI, CComBSTR("SYSTEM:PRESET"), response);
            SendScpiCommand(spSCPI, CComBSTR("CALC:PAR:CAT?"), response);
            std::cout << CString(response) << std::endl;
            SendScpiCommand(spSCPI, CComBSTR("THIS:IS:A:SYNTAX:ERROR"),
                response);
        }
    }
    CoUninitialize();
    return 0;
}

```

## External Testset Control

---

The following VB Script example exercises the COM commands used to control the Z5623AK64 testset. For a description of each command, see [TestsetControl Object](#)

### [See Other COM Example Programs](#)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as Testset.vbs. [Learn how to setup and run the macro.](#)

```
' Demonstrate some COM commands for external testsets.

Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")
Sub DemoTestset (na)
Dim testsets, tset1
Dim portNum
Dim chNum, address
Set testsets = na.ExternalTestsets
chNum = 1

' Load a configuration file.
' NOTE: the K64 testset is only compatible with 4-port analyzers.
address = 0
testsets.Add "Z5623AK64", address

' Get the testset object
' in the testsets collection.
Set tset1 = testsets(1)

' Show the selections available for each port.
For portNum = 1 To 4
MsgBox("Port " & CStr(portNum) & " catalog: " & tset1.PortCatalog(portNum))
Next

' Set port mappings on channel 1.
tset1.OutputPorts(chNum) = "5 ext R,2 int R,3 int R,6 int R"
```

```
' Set control lines.
tset1.ControlLines(chNum) = 85

' Set label.
tset1.Label(chNum) = "Some label"

' Enable external testset control. This automatically enables status bar display
as well.
tset1.Enabled = True
End Sub

' The testset used in this demo is only usable on 4-port analyzers
If (pna.NumberOfPorts <> 4) Then
MsgBox("This program only runs on 4-port analyzers.")
Else
DemoTestset(pna)
End If
```

## PathConfiguration Example

---

**Note:** These commands are accessible only for PNA-X models.

These Visual Basic and [C# examples](#) exercise various commands on the:

- [PathConfigurationManager Object](#)
- [PathConfiguration Object](#)
- [PathElement Object](#)

### VB Example

```
' Create / Get the PNA application
Dim app
Set app = CreateObject("AgilentPNA835x.Application")
' Preset the instrument
app.Preset
' Get a channel interface on which to operate
Dim chan
Set chan = app.ActiveChannel
' Modify the Default configuration, and save it as "My Config"
chan.PathConfiguration = "Default"
' Set the "Combiner" element to value "Reversed"
chan.PathConfiguration.Element("Combiner").Value = "Reversed"
' Set the "Src1" element to value "High Power"
chan.PathConfiguration.Element("Src1").Value = "High Power"
' Change the description text
chan.PathConfiguration.DescriptionText = "Connect J8 to J9."
' Store the modified configuration
chan.PathConfiguration.Store ("My Config")
' Set the instrument's path config back to the default (req. 8)
chan.PathConfiguration = "Default"
' Load a previously saved configuration onto channel 1
app.PathConfigurationManager.Load 1, "My Config"
```

### C# Example

```
Type pnaType = Type.GetTypeFromProgID("AgilentPNA835x.Application", "PNA-NAME-  
HERE");  
  
AgilentPNA835x.Application pna =  
(AgilentPNA835x.Application)Activator.CreateInstance(pnaType);  
  
AgilentPNA835x.Channel chan = (AgilentPNA835x.Channel)pna.ActiveChannel;  
  
// Preset the Instrument  
pna.Preset();  
  
// Modify the Default configuration, and save it as "My Config"  
chan.set_PathConfiguration("Default");  
  
// Set the "Combiner" element to value "Reversed"  
chan.get_PathConfiguration().get_Element("Combiner").Value = "Reversed";  
  
// Change the description text  
chan.get_PathConfiguration().DescriptionText = "Connect J8 to J9.";  
  
// Store the modified configuration  
chan.get_PathConfiguration().Store("My Config");  
  
// Set the instrument's path config back to the default (req. 8)  
chan.set_PathConfiguration("Default");  
  
// Load a previously saved configuration onto channel 2  
pna.PathConfigurationManager.LoadConfiguration(1, "My Config");
```

---

Last Modified:

6-Oct-2009    Fixed last two lines (ch1) and Add C#

## Create a Narrowband Pulsed Measurement using the PNA-X and COM

---

The following COM example demonstrates how to create a narrowband pulsed measurement using the Pulsed Application DLL on the [PNA-X](#).

See the [example program](#) for the E836x PNA models.

See the example program for wideband pulsed measurements on the PNA-X.

It first gets valid configuration settings and then uses those settings to configure the PNA and internal pulsed generators.

To run this program, you need:

- PNA-X
- [Pulsed Application](#) (Option H08)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as BalancedCOM.vbs. [Learn how to setup and run the macro.](#)

### See Also

- Learn how to [install and register the pulsed .dll](#) on your PC
- See the [ConfigEnhancedNB2](#) method for sending and returning parameters to the .dll.
- See the [ConfigEnhancedNBIFAtten](#) method for setting the receiver IF gain.
- See the [COM IF Configuration](#) commands used in the program.
- See the equivalent [SCPI IF Configuration](#) commands.

```
'Interfaces
Dim OApp As AgilentPNA835x.application
Dim OIntPG As AgilentPNA835x.PulseGenerator
Dim OPathConf As AgilentPNA835x.PathConfiguration
Dim OFilter As AgilentPNA835x.SignalProcessingModuleFour
Dim OIF As AgilentPNA835x.IFConfiguration

'Pulsed parameters
Dim DPRF As Double
Dim DBW As Double
Dim DPhysicalIF As Double
Dim DNCO As Double
Dim DCF As Double
```

```

Dim DGD As Double
Dim DGW As Double
Dim DSWGD As Double
Dim DSWGW As Double
Dim DSWGR As Long
Dim LStage1TapArray() As Long
Dim LStage2TapArray() As Long
Dim LStage3TapArray() As Long
Dim BFixedPRF As Boolean
Dim IIFatten As Integer
'pulsed DLL interface
Dim OPulsed As New AgilentPNAPulsed.application
'Pulsed settings
DPRF = 5000 'Hz
DBW = 500 'Hz
BFixedPRF = True
DNCO = 0#
DCF = 0#
DGD = 0#
DGW = 0.000001
DSWGR = 0#
'Send desired pulsed parameters to the pulsed configuration DLL. The DLL will return
a new set of pulse parameters to send to the PNA.
OPulsed.ConfigEnhancedNB2 DPRF, DBW, DPhysicalIF, DNCO, DCF, LStage1TapArray,
LStage2TapArray, LStage3TapArray, BFixedPRF, DGD, DGW, DSWGD, DSWGW, DSWGR
'
'Send configuration to PNA
'
'Connect to the PNA application
Set OApp = CreateObject("AgilentPNA835x.Application")
'Create instance of pulse generators on active channel
Set OIntPG = OApp.ActiveChannel.PulseGenerator
'Create instance of path configuration on active channel
Set OPathConf = OApp.ActiveChannel.PathConfiguration
'Create instance of digital filter on active channel
Set OIF = OApp.ActiveChannel.IFConfiguration
'Create instance of Hana digital filter on active channel

```



```

Set OFilter = OApp.ActiveChannel.SignalProcessingModuleFour
'Set up master pulse period for internal pulse generators
OIntPG.Period = 1 / DPRF
'Set up internal pulse generator output #1 to drive internal source modulation
OIntPG.Width(1) = 0.0001 '100us
OIntPG.Delay(1) = 0.00001 '10us
OIntPG.State(1) = True
OPathConf.Element("PulseModDrive").Value = "Pulse1"
'Set up internal pulse generator output #2 to drive internal receiver gates for a 2
port PNA-X
OIntPG.Width(2) = 0.000001 '1us
OIntPG.Delay(2) = 0.00005 '50us
OIntPG.State(2) = True
OPathConf.Element("IFGateA").Value = "Pulse2"
OPathConf.Element("IFGateB").Value = "Pulse2"
OPathConf.Element("IFGateR1").Value = "Pulse2"
OPathConf.Element("IFGateR2").Value = "Pulse2"
'
'Configure PNA in pulsed mode operation
'
'Turn off ALC and turn on modulator control
OApp.ActiveChannel.ALCLevelingMode(1) = naALCOpenLoop 'Source 1 output #1 ALC off
OPathConf.Element("Src1Out1PulseModEnable").Value = "Enable" 'Enable Source 1 pulse
modulator
'Set path and enable IF gates
OApp.ActiveChannel.IFBandwidth = DBW
OPathConf.Element("IFSigPathAll").Value = "NBF"
'Set filter stages based on pulse parameters
OIF.IFFrequency = DPhysicalIF
OIF.IFFrequencyMode = naMANUAL
OFilter.Stage1Frequency = DNCO
OFilter.Stage1Coefficients = LStage1TapArray
OFilter.Stage2Coefficients = LStage2TapArray
OFilter.Stage3FilterType = "RECT"
OFilter.Stage3Parameter("C") = LStage3TapArray(0)
OFilter.FilterMode = naMANUAL
'Set receivers to auto gain setting

```

```
OPulsed.ConfigEnhancedNBIFatten DPRF, DGW, IIFatten '1us pulse width
OPathConf.Element("NBFATNA").Value = IIFatten
OPathConf.Element("NBFATNB").Value = IIFatten
OPathConf.Element("NBFATNR1").Value = IIFatten
OPathConf.Element("NBFATNR2").Value = IIFatten
MsgBox "Done"
```

---

Last Modified:

11-Jun-2007    Rev 2- some edits

16-Feb-2007    MX New topic

## Setup Basic Measurements

---

This VBScript program sets up four basic s-parameter measurements in four windows, all in a single channel. Handles are created to the measurement, channel, and window objects so that subsequent settings can be made for each.

**Note:** This is only an example. This is not necessarily the most efficient way to make basic S-parameter measurements.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Basic.vbs. [Learn how to setup and run the macro.](#)

[See PNA Object Model](#)

See [CreateSParameterEx](#)

[See Other COM Example Programs](#)

```
Set pna = CreateObject("AgilentPNA835x.Application")
pna.Preset
' Get a handle to the preset channel 1, S11 meas, and window(1)
set meas1=pna.ActiveMeasurement
set chan1=pna.ActiveChannel
set win1=pna.ActiveNAWindow
' Creates a new S21 measurement in New window(2)
pna.CreateSParameterEx 1,2,1,2,2
set meas2=pna.ActiveMeasurement
set win2=pna.ActiveNAWindow
' Creates a new S12 measurement in New window(3)
pna.CreateSParameterEx 1,1,2,1,3
set meas3=pna.ActiveMeasurement
set win3=pna.ActiveNAWindow
' Creates a new S22 measurement in New window(4)
pna.CreateSParameterEx 1,2,2,2,4
set meas4=pna.ActiveMeasurement
set win4=pna.ActiveNAWindow
'Make settings
'set Stop Frequency for channel
chan1.StopFrequency=1e9
'set Display formats
meas1.format=1 'Lin Mag
```

```
meas2.format=2 'Log Mag
meas3.format=3 'Phase
meas4.format=4 'Smith
'Show title in all windows
win1.title="Win #1"
win2.title="Win #2"
win3.title="Win #3"
win4.title="Win #4"
```

---

Last Modified:

19-Apr-2010    Modified example for same channel

## Setup Compression Marker

---

This example program does the following:

- Creates a compression marker
- Queries the Power Out and Power In values

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as CompMkr.vbs. [Learn how to setup and run the macro.](#)

[See the FIFO object.](#)

---

### [See Other COM Example Programs](#)

```
Set app = CreateObject("AgilentPNA835X.Application")
set meas = app.activemeasurement
'get the COM marker object
'and create marker1
set mark = meas.marker(1)
'set the compression level
mark.compressionlevel = 1.5
'make it a compression marker
'and find the compression point
mark.searchcompressionpoint
'return power out and power in
'power in
dim answer
answer = mark.compressionpin
wscript.echo("pin: " & answer)
'power out
answer = mark.compressionpout
wscript.echo("pout: " & answer)
```

---

Last Modified:

12-Feb-2009    MX New topic

## Setup FastCW and FIFO

---

This example program does the following:

- Setup an A/R and B/R measurement
- Turn ON point averaging
- Set external edge triggering (commented out)
- Set FIFO and Fast CW
- Write data into FIFO data buffer
- Read FIFO data buffer

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as FIFO.vbs. [Learn how to setup and run the macro.](#)  
[See the FIFO object.](#)

[See Other COM Example Programs](#)

```
dim app
set app = createobject("Agilentpna835x.application")
' Setup and measure A/R and B/R
app.Reset
app.CreateMeasurement 1,"A/R1",0
app.CreateMeasurement 1,"B/R1",0
' Set IFBW to 600khz (400thousand pts/second)
app.activeChannel.IFBandwidth = 600e3
' Point Averaging Count = 10
app.activeChannel.AverageMode = 0 ' point
app.activeChannel.averagingFactor = 10
app.ActiveChannel.averaging = 1 ' turn on
' Edge triggering - positive edge
'app.TriggerSetup.ExternalTriggerConnectionBehavior(1) = 2 ' BNC1 = trigger
positive edge
'app.TriggerSetup.Source = 2 ' external
'app.ActiveChannel.TriggerMode = 0 'point
' Setup FIFO and Fast CW count
```

```
app.ActiveChannel.Hold 1 ' hold - synchronous
app.FIFO.State = 1' turn on FIFO
app.FIFO.Clear
app.activechannel.sweepType = 3 ' CW sweep
app.activechannel.fastCWpointcount = 1000000' set the point count to 1million
app.activechannel.single 1 ' synchronous single
' the single will wait until the end of sweep.
'You do not have to wait until end of sweep to start emptying FIFO.
points = app.fifo.datacount
msgbox points
' points == 2000000 ' points = 2million. Took 5 seconds to acquire
For I = 0 to 1 ' 2 iterations (2 parameters * 2 sets of 1 million)
Dim data
Data = app.fifo.data(1000000)
Next
msgbox data(0)
```

---

Last Modified:

10-Oct-2008    MX New topic

## Setup Receiver Leveling

---

The following VB Script example exercises the COM commands used to setup Receiver Leveling.

### See Also

[About Receiver Leveling](#)

[RxLevelingConfiguration Object](#)

---

### See Other COM Example Programs

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save on the PNA hard drive as RxLevel.vbs. [Learn how to setup and run the macro.](#)

```
' Demonstrate some COM commands for Receiver Leveling.
Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")
Dim chan
Set chan = pna.ActiveChannel
Dim RxLevel
Set RxLevel = chan.GetRxLevelingConfiguration
Dim srcPort
srcPort = 1
pna.Preset
RxLevel.ReferenceReceiver(srcPort) = "R1"
RxLevel.Tolerance(srcPort)= 0.02
RxLevel.IterationNumber(srcPort)= 10
RxLevel.FastMode(srcPort)=True
RxLevel.LevelingIFBW(srcPort)= 100
RxLevel.PowerOffset(srcPort)= 0
RxLevel.PowerMax(srcPort)= 20
RxLevel.PowerMin(srcPort)= -50
RxLevel.SafeMode(srcPort)= True
RxLevel.State(srcPort)= True
```

---

Last Modified:

3-Jun-2010    MX New topic





## Show Custom Cal Windows during a Guided Calibration

---

This VBScript program shows how to send commands that allow you to view specific 'custom' windows, and sweep specific channels, during a UI (Cal Wizard) or remote calibration.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as CalWindow.vbs. [Learn how to setup and run the macro.](#)

These commands are used to show and sweep specific windows and channels:

- [AllowChannelToSweepDuringCalAcquisition Method](#)
- [DisplayNAWindowDuringCalAcquisition Method](#)
- [DisplayOnlyCalWindowDuringCalAcquisition Method](#)
- [SweepOnlyCalChannelDuringCalAcquisition Method](#)

The following command sweeps the Cal Windows before remote acquisition:

- [SetupMeasurementsForStep Method](#)

---

### [See Other COM Example Programs](#)

```
Set pna = CreateObject("AgilentPNA835x.Application")
pna.Preset
' get a handle to the preset channel 1 so that we can later cal it
set meas=pna.ActiveMeasurement
' Creates a new S21 measurement in channel 2 and New window(2)
' this will be the channel and window to show during cal
pna.CreateSParameterEx 2,2,1,2,2
Set calMgr = pna.GetCalManager
Set guidedCal = calMgr.GuidedCalibration
' show window 2 during cal
calMgr.DisplayNAWindowDuringCalAcquisition 2,True
' sweep channel 2 during calibration of chan 1
calMgr.AllowChannelToSweepDuringCalAcquisition 1,2,True
' make Channel1 the active channel
' activating the measurement also activates the channel
meas.Activate
```

```

guidedCal.Initialize 1, True
' Do 2-port cal
' Select the connectors
guidedCal.ConnectorType(1) = "APC 3.5 female"
guidedCal.ConnectorType(2) = "APC 3.5 male"
' Select the Cal Kit for each port being calibrated.
guidedCal.CalKitType(1) = "85052D"
guidedCal.CalKitType(2) = "85052D"
' Initiate the calibration and query the number of steps
numSteps = guidedCal.GenerateSteps
' Measure the standards, compute and apply the cal
value = MsgBox("Number of steps is " + CStr(numSteps))
' Measure the standards
For i = 1 to NumSteps
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = guidedCal.GetStepDescription(i)
' Sweep the Cal window prior to standard acquisition
guidedCal.SetupMeasurementsForStep i
' prompt to connect standard
value = MsgBox(strPrompt, vbOKOnly, step)
' measure standard
guidedCal.AcquireStep i
Next
' Conclude the calibration
guidedCal.GenerateErrorTerms
MsgBox ("Cal is complete!")
' clear the Cal window and channel flags
calMgr.DisplayOnlyCalWindowDuringCalAcquisition
calMgr.SweepOnlyCalChannelDuringCalAcquisition

```

Last Modified:

1-Nov-2007    New topic

## COM Events Example

---

This Visual Basic program shows how to monitor the end of sweep.

The program will set sweep time to various amounts and BEEPs when sweep is completed. This method allows other processes to continue while waiting for end-of-sweep. This program stops after 10 loops.

**Note:** To avoid **Permission Denied** problems, this should be run on the PNA and not a PC. To run it from a PC both units must be "trusted" and on the same [domain/workgroup](#).

```
Option Explicit
Dim na As AgilentPNA835x.Application
Dim WithEvents naEvt As AgilentPNA835x.Application
Dim ch As AgilentPNA835x.Channel
Dim sweepComplete As Boolean

Private Sub Form_Load()

Dim N As Integer
Set na = CreateObject("AgilentPNA835x.application")
na.preset
Set ch = na.ActiveChannel
na.DisallowAllEvents          ' Turn off all events
Set naEvt = na                ' Enable event interrupts
Do
N = N + 1                    ' Loop counter
ch.sweepTime = 1 + (Rnd * 9) ' Set random sweep-time from 1-10 sec
sweepComplete = False
ch.Single False              ' Trigger sweep
naEvt.AllowEventCategory naEventCategory_CHANNEL, True ' Enable Channel event
Do
DoEvents                    ' Allows other processes to continue
Loop Until sweepComplete = True
naEvt.AllowEventCategory naEventCategory_CHANNEL, False ' Disable event until
ready for next one
Beep                        ' Do end-of-sweep processing here;

Loop Until N > 10
End

End Sub

Private Sub naEvt_OnChannelEvent(ByVal eventID As Variant, ByVal chNumber As
Variant)
' In this example we don't care about the channel info
If eventID = naEventID_CHANNEL_TRIGGER_COMPLETE Then sweepComplete = True
End Sub
```

## Upload a Segment Table using C++

---

This example program shows the Variant method for uploading a segment sweep to the PNA using the [SetAllSegments](#) method

```
#include "stdafx.h"
#include <stdio.h>
#include "atlbase.h"
#include "objbase.h"

// import the PNA type library
//-----
#import "C:/Program Files/Common Files/Agilent/PNA/835x.tlb" no_namespace,
named_guids

int _tmain(int argc, _TCHAR* argv[])
{

    // interface pointers to retrieve COM interfaces
    HRESULT hr;
    IUnknown* pUnk = 0;
    IApplication* pNA = 0;
    IChannel* pChan = 0;
    IMeasurement* pMeas = 0;
    IMeasurement5* pMeas5 = 0;
    IArrayTransfer* pTrans = 0;
    ISegments* pSeg = 0;
    ISegments2* pSeg2 = 0;

    //Variables for X and Y data read portion
    SAFEARRAY* sArray;
    _variant_t vXVals;
    double HUGE* xVals;
    float* pScalarData;

    //Variables for Segment portion
    double Fstart, Fstop, SegWidth;
    long i[2];
```

```

int num_points = 11;
SAFEARRAY* pSA;
VARIANT vSeg;
VARIANT v;

int NUM_SEGS = 10;
int SEG_SIZE = 7;

//Create SafeArray to hold the segment data
SAFEARRAYBOUND aDim[2]; //This must be 2 the PNA expects to see a 2 dimensional
array
aDim[0].lLbound = 0;
aDim[0].cElements = SEG_SIZE; //This will be set to 7 unless port power is
uncoupled
aDim[1].lLbound = 0;
aDim[1].cElements = NUM_SEGS;
pSA= SafeArrayCreate(VT_VARIANT,2,aDim); //The cDim parameter must be set to 2 as
the PNA expects a 2D array

//Init Variant to set values in Safearray
VariantInit(&vSeg);

Fstart=10e6;
Fstop=3e9;
SegWidth=(Fstop-Fstart)/NUM_SEGS;

//Loop to write segment data
for(int j=0; j<NUM_SEGS; ++j)
{
    i[1]=j; //Set Segment #

    //Segment Definition
    i[0] = 0;
    vSeg.vt = VT_BOOL; //First parameter is Boolean
    vSeg.boolVal = VARIANT_TRUE; //Segment State
    SafeArrayPutElement(pSA, i, &vSeg);
    i[0] += 1;
}

```

```

vSeg.vt = VT_I4; //Second parameter is an integer
vSeg.intVal = num_points; //Number of Points
SafeArrayPutElement(pSA, i, &vSeg);
i[0] += 1;
vSeg.vt = VT_R8; //Remaining parameters are of type double
vSeg.dblVal = Fstart+j*SegWidth; //Start Frequency
SafeArrayPutElement(pSA, i, &vSeg);
i[0] += 1;
vSeg.dblVal=vSeg.dblVal+SegWidth; //Stop Frequency
SafeArrayPutElement(pSA, i, &vSeg);
i[0] += 1;
vSeg.dblVal = 1.0e3; //IF Bandwidth
SafeArrayPutElement(pSA, i, &vSeg);
i[0] += 1;
vSeg.dblVal = 0.0; //Dwell time
SafeArrayPutElement(pSA, i, &vSeg);
i[0] += 1;
vSeg.dblVal = -5.0; //Power
SafeArrayPutElement(pSA, i, &vSeg);
}

```

```

//vSeg no longer needed, clean up

```

```

VariantClear(&vSeg);

```

```

//Declare Variant to use with Segment data

```

```

VariantInit(&v);

```

```

v.vt = VT_ARRAY|VT_VARIANT;

```

```

v.parray = pSA; //write safearray to variant

```

```

// Initialize the COM subsystem

```

```

CoInitialize(NULL);

```

```

CoInitializeSecurity(NULL,          //security descriptor

```

```

    -1,          // authn svc entries

```

```

    NULL,        // authn svcs

```

```

    NULL,        // reserved

```

```

    RPC_C_AUTHN_LEVEL_NONE,

```

```

RPC_C_IMP_LEVEL_IMPERSONATE,
NULL,          // authn info
0,             // capabilities
NULL);         // reserved

// Create an instance of the network analyzer
// Request the NA's IUnknown interface
hr = CoCreateInstance(CLSID_Application, 0, CLSCTX_ALL, IID_IUnknown, (void**)
&pUnk);
if (!FAILED(hr))
{
    // QueryInterface for the INetworkAnalyzer interface of the NetworkAnalyzer
    object
    hr = pUnk->QueryInterface(IID_IApplication, (void**) &pNA);

    if (!FAILED(hr))
    {
        // Reset the analyzer to instrument preset
        pNA->Reset();

        // Create S11 measurement
        pNA->CreateSParameter(1, 1, 1, 1);

        // Set pChan variable to point to the active channel
        pNA->get_ActiveChannel(&pChan);

        //Show Segment table
        pNA->NAWindows->Item(1)->ShowTable((NATableType)2);

        //Get handle to ISegments Interface
        pChan->get_Segments(&pSeg);

        //Get handle to ISegments2 Interface
        hr = pSeg->QueryInterface(IID_ISegments2, (void**) &pSeg2);

        //Set Segment Sweep Options

```



```

pSeg2->IFBandwidthOption = VARIANT_TRUE;
pSeg2->SourcePowerOption = VARIANT_TRUE;

//Push segments to PNA
pSeg2->SetAllSegments(v);

//Set Sweep Type to Segment Sweep
pChan->SweepType = naSegmentSweep;

if (pChan)
{
    // Set pMeas variable to point to the active measurement
    pNA->get_ActiveMeasurement(&pMeas);

    if(pMeas)
    {
        // Setup the channel for a single trigger
        pChan->Hold(true);
        pNA->TriggerSignal = naTriggerManual;
        pChan->TriggerMode = naTriggerModeMeasurement;

        // Make the PNA application visible
        pNA->put_Visible(true);

        // Send a manual trigger to initiate a single sweep
        pChan->Single(true);

        // QueryInterface for the IArrayTransfer interface of the
NetworkAnalyzer object
        hr = pMeas->QueryInterface(IID_IArrayTransfer, (void**)&pTrans);

        // Get handle for IMeasurement5 interface
        hr = pMeas->QueryInterface(IID_IMeasurement5, (void**)&pMeas5);

        if (!FAILED(hr))
        {
            int val = num_points*NUM_SEGS;

```

```

// Store the data in the "result" variable
pScalarData = new float[val];
xVals = new double[val*2];

//Get X axis values
vXVals=pMeas5->GetXAxisValues();

//Convert _variant_t array to a SAFEARRAY
sArray = vXVals.parray;

//Convert data from SAFEARRAY to double array. Each
SAFEARRAY value
double
data point

//is 16 bytes so it takes up 2 floats so the xVals size is

//the number of points. This also means that every other

//in the resulting array can be discarded.
hr = SafeArrayAccessData(sArray, (void HUGEPP*)&xVals);

//Get Measurement Values
pTrans->getScalar(naRawData, naDataFormat_LogMag, (long
*)&val, pScalarData);

// Display the result
printf("S11(dB) - Visual C++ COM Example for PNA operating
in segment sweep mode/n/n");
for (int j = 0; j < val; j++)
{
    //Write value...the xVals array is offset by 1 in each
data point since

    //the return data is 16 bytes and each double is 8.
printf("%.3lf GHz, %.4f/n",xVals[2*j+1]/1e9,
pScalarData[j]);

}

}

}

```

```
    }  
    else  
    {  
        printf("Programmed failed to connect to the PNA.");  
    }  
}  
CoUninitialize();  
  
system("PAUSE");  
return 0;  
}
```

## Using C#

---

The following are common C# examples:

### Connecting to a specific PNA via DCOM:

```
public AgilentPNA835x.Application Connect(string hostname)
{
    AgilentPNA835x.Application pna = null;
    try
    {
        Type t = Type.GetTypeFromProgID("AgilentPNA835x.Application", hostname,
true);
        pna = (AgilentPNA835x.Application) Activator.CreateInstance(t);

    }
    catch (Exception e)
    {
        HandleExceptions(e);
    }
    return pna;
}
```

### Using the GetData Interface

```
AgilentPNA835x.IMeasurement meas = app.ActiveMeasurement;
    object[] dataArrayAsObj;
    dataArrayAsObj =
(object[])meas.getData(AgilentPNA835x.NADataStore.naMeasResult,
AgilentPNA835x.NADataFormat.naDataFormat_LogMag);
    float[] dataArray = new float[dataArrayAsObj.Length];
    for (int j = 0; j < dataArrayAsObj.Length; j++)
    {
        dataArray[j] = (float)dataArrayAsObj[j];
    }
```

### 2-dimensional GetData

```
AgilentPNA835x.IMeasurement meas = app.ActiveMeasurement;
    app.ActiveChannel.Single(true);
    object[,] dataArrayAsObj;
    dataArrayAsObj =
(object[,])meas.GetData(AgilentPNA835x.NADataStore.naRawData,
AgilentPNA835x.NADataFormat.naDataFormat_Smith);
    float[,] dataArray = new float[dataArrayAsObj.Length,2];
    for (int j = 0; j < dataArrayAsObj.Length; j++)
    {
        dataArray[j,0] = (float)dataArrayAsObj[j,0];
        dataArray[j,1] = (float)dataArrayAsObj[j,1];
    }
```

### Other C# / .NET Topics

[Perform a Guided Cal with CSharp](#)

[Getting a handle to the Noise Figure Cal object.](#)

[Using .NET](#)

---

Last Modified:

15-Sep-2009    Modified GetData example

## Configure for COM-DCOM Programming

---

Before developing or running a COM program, you should first establish communication between your PC and the analyzer. This process is referred to as gaining **Access** to the analyzer. You should then register the PNA type library on your PC.

**DCOM** (Distributed Component Object Model) refers to accessing the PNA from a remote PC.

**COM** refers to accessing the PNA application from the analyzer PC.

- [Access Concepts](#)
- [Access Procedures](#)
- [Register the PNA Type Library on Your PC](#)
- [Problems?](#)

**Note:** After performing a [Firmware Upgrade](#) you must copy the new type library to your development PC to get access to new COM commands. See [Register the analyzer on your PC](#).

### [Other Topics about COM Concepts](#)

For detailed information on this subject, see <http://na.tm.agilent.com/pna/DCOMSecurity.html>

## Access Concepts

PNAs are shipped from the factory such that **Everyone** has permission to launch and access the PNA application via COM/DCOM. The term **Everyone** refers to a different range of users depending on whether the PNA is a member of a **Domain** or **Workgroup** (it must be one or the other; not both). By default, the PNA is configured as members of a workgroup. Therefore, **Everyone** includes only those users who have been given logon accounts on the PNA.

### Workgroup

A workgroup is established by the **PNA administrator** declaring the workgroup name and declaring the PNA as a member of the workgroup. A workgroup does not require a network administrator to create it or control membership.

**Everyone** includes only those users who have been given logon accounts on the PNA.

By default, the PNA is configured as members of a workgroup named WORKGROUP.

**Note:** To setup a logon account for a new user, see [Additional Users](#).

The easiest method of gaining DCOM access, is to make the user's account name and password on the PNA to EXACTLY match their PC logon account name and password.

### Domain

A domain is typically a large organizational group of computers. Network administrators maintain the domain and control which machines have membership in it.

**Everyone** includes those people who have membership in the domain. In addition, those with logon accounts can also access the analyzer.

## Summary

- A **Workgroup** requires no maintenance, but allows DCOM access to only those users with a log-on account for the PNA.
- A **Domain** requires an administrator, but all members of the domain and those with logons to the analyzer are allowed DCOM access to the PNA.

The following section "Access Procedures" provides a tighter level of security allowing only **selected** (not **Everyone**) domain and workgroup users DCOM **Access** and **Launch** capability of the PNA.

## Access Procedures

Perform this procedure for the following reasons:

- To allow only selected users (not everyone) remote Access and remote Launch capability to the PNA. Launch capability is starting the PNA application if it is not already open.
- To verify that you have DCOM access to the analyzer.

**Note:** Before doing this procedure, you must first have a logon account on the PNA. See [PNA User Accounts](#)

**The following procedure grants specific users DCOM access and launch capability of the PNA application:**

To perform this procedure, you must first [minimize the PNA](#) application.

[How do I know which Operating System I have?](#)

Windows 2000	Windows XP
On the PNA, click the Windows <b>Start</b> button	On the PNA, click the Windows <b>Start</b> button
Click <b>Run</b>	Click <b>Run</b>
In the <b>Open:</b> box, type <b>dcomcnfg</b>	In the <b>Open:</b> box, type <b>dcomcnfg</b>
Click <b>OK</b>	Click <b>OK</b>
In the Distributed COM Configuration Properties window, Click on <b>Agilent PNA Series</b> in the Applications list. Then click <b>Properties...</b>	Open the following folder sequence: Component Services Window Component Services Computers My Computer DCOM Config Right click <b>Agilent PNA Series</b>

	Click <b>Properties</b>
Click the <b>Security</b> tab	Click the <b>Security</b> tab
Click <b>Use custom access permissions</b> then click <b>Edit</b>	Under <b>Access Permissions</b> , click <b>Customize</b> , then click <b>Edit</b>
In <b>Registry Value Permissions</b> , select <b>Everyone</b> , then click <b>Remove</b>	Select <b>Everyone</b> , then click <b>Remove</b>
Click <b>Add</b>	Click <b>Add</b>
You could either select one or more of these groups to have access to the PNA, or specific users. To give specific users access, click <b>Show users</b> or <b>Members</b> , then select the name from the list.	Type a group name or user account name
Click <b>Add</b> , then click <b>OK</b>	Click <b>OK</b>
<b>Launch Permission</b>	
Click <b>Use custom launch permissions</b> , then click <b>Edit</b>	Under <b>Launch Permissions</b> , click <b>Customize</b> , then click <b>Edit</b>
In Registry Value Permissions, select <b>Everyone</b> then click <b>Remove</b>	Select <b>Everyone</b> , then click <b>Remove</b>
Click <b>Add</b>	Click <b>Add</b>
<p>You could either select one or more of these groups to have launch permission of the PNA, or specific users.</p> <ul style="list-style-type: none"> <li>• To give groups launch permission, select the group from the list.</li> <li>• To give specific users launch permission, click <b>Show users</b> or <b>Members</b>, then select the name from the list.</li> </ul>	Type a group name or user account name
Click <b>Add</b> , then click <b>OK</b>	Click <b>OK</b>

## Register the PNA Type Library on Your PC

The type library contains the PNA object model. On your PC, there is a Registry file that keeps track of where object models are located. Therefore, you must register the type library on the PC that will be used to develop code and run the program. It is much more efficient to have the type library registered at design time (BEFORE running your COM program).

Do the following two items before proceeding:



1. Connect your PC and the PNA to LAN.
2. Either map a drive to the analyzer or copy the type library files on a floppy disk or other media. See [Drive Mapping](#).

**Note:** To register the type library on your PC, you must be logged on as an administrator of your PC. Learn about [User Accounts](#).

This procedure will do the following:

- Register the Network Analyzer application on your PC.
  - Copy and register the proxystub (835xps.DLL) onto the PC.
  - Copy and register the PNA type library (835x.tlb) onto the PC.
  - Copy and register the FCA type library (fca.tlb) onto the PC.
1. Using Windows Explorer on your PC, find the Analyzer's C: drive. The drive will not be named "C:" on your PC, but a letter you assigned when mapping the drive.
  2. Navigate to **Program Files / Agilent / Network Analyzer / Automation**
  3. Double-click **pnaproxy.exe** and follow the prompts to Install PNA Proxy. If the installation offers a choice of Modify, Repair, or Remove, then select **Remove**. Then double-click on **pnaproxy.exe** again.
  4. When prompted, type the Computer name of the PNA ([Learn how to find this](#)).
  5. After the install program runs, the PNA and FCA type library should be registered on your PC.
  6. Your programming environment may require you to set a reference to the PNA type library now located on your PC. In Visual Basic, click **Project, References**. Then browse to **C:/Program Files/Common Files/Agilent/PNA**. Select **835x.tlb**

### Problems?

- These procedures will fail if there are any programs using the PNA type library (for example: Visual basic, VEE, Visual Studio, or any other application program that may communicate with the PNA).
  - Perform the following procedure if the previous procedure did not return an error, but you cannot connect to the PNA.
  - If you received an error, check that both the account name and password used on both the PNA and PC match EXACTLY.
  - If you still get errors, see <http://na.tm.agilent.com/pna/DCOMSecurity.html>.
1. [Map a drive](#) from your remote PC to the PNA. Note the drive letter your PC assigns to the PNA. Substitute this drive letter for **PNA** in the following procedure.
  2. On your PC, go to a DOS prompt c:>

3. Type **PNA:** (for example o:)
4. Type **cd program files/agilent/network analyzer/automation**
5. Type **copy 835xps.dll c:/program files/common files/agilent/pna**
6. Type **copy 835x.tlb c:/program files/common files/agilent/pna**

If you will NOT be using [FCA commands](#), skip steps 7,.8, and 9.

7. Type **cd..**
8. Type **cd extensions/fca**
9. Type **copy fca.tlb c:/program files/common files/agilent/pna**
10. If it is not already there, copy **regtlib.exe** from **PNA:/WINNT** to your C:/<windows>/system32 directory (<windows> is OS-dependent- it is either windows or WINNT)
11. Type **regtlib C:/program files/common files/agilent/pna/835x.tlb**
12. Type **regsvr32 C:/program files/common files/agilent/pna/835xps.dll**
13. Type **regtlib C:/program files/common files/agilent/pna/fca.tlb**

Perform the [Access Procedure](#) after doing these steps.

## COM Fundamentals

---

The following terms are discussed in this topic:

- [Objects](#)
- [Interfaces](#)
- [Collections](#)
- [Methods](#)
- [Properties](#)
- [Events](#)
- [Visual Basic Syntax](#)

**Note:** The information contained in this topic is intended to help an experienced SCPI programmer transition to COM programming. This is NOT a comprehensive tutorial on COM programming.

### [Other Topics about COM Concepts](#)

#### Visual Basic Syntax

The examples in PNA Help use Visual Basic as the programming environment for COM, which uses 'dot' notation.

To set a property, follow the object reference with:

- a period (.)
- property or method
- an equal sign (=)
- the new value

For example:

```
object.property = value 'This Green text following an apostrophe (') is a comment.
```

To read a property, a variable to contain the returned value is followed with:

- an equal sign (=)
- an object, or reference to an object
- a period (.)
- property

For example:

```
variable = object.property
```

To execute a method, an object, or reference to an object is followed with:

- a period (.)
- the method
- a blank space
- any required parameters

For example:

```
object.method parameters
```

Some methods return values, such as methods that return data. To return data from a method, a variable to contain the returned data is followed with:

- an equal sign (=)
- an object, or reference to an object
- a period (.)
- the method
- any required parameters enclosed in parenthesis

```
variable = object.method (parameters)
```

## Objects

The objects of the Network Analyzer (Application) are arranged in a hierarchical order. The [PNA object model](#) lists the objects and their relationship to one another.

In SCPI programming, you must first select a measurement before making settings. With COM, you first get a handle to the object (or collection) and refer to that object in order to change or read settings (properties).

For more information on working with objects, see [Getting a Handle to an Object](#).

## Interfaces

A COM Interface is the connection to an object. When you get a handle to an object, you are actually using an interface to an object. This is important if you are developing PNA code that will run on multiple code versions. For more information, see [PNA Interfaces](#).

## Collections

A collection is an object that contains several other objects of the same type. For example, the **Channels** collection contains all of the channel objects.

**Note:** In the following examples, the collections are referred to as a variable. Before using a collection object, you must first get an instance of that object. For more information, see [Getting a Handle to an Object](#)

Generally, items in a collection can be identified by **number** or by **name**. The order for objects in a collection cannot be assumed. They are always unordered and begin with 1. For example, in the following procedure, chans(1) is used to set averaging on the **first** channel in the Channels collection (not necessarily channel 1).

```
Sub SetAveraging()  
    chans(1).AveragingFactor = 10  
End Sub
```

The following procedure uses the measurement string name to set the display format for a measurement in the measurements collection.

```
meass("CH1_S11_1").Format = 1
```

You can also manipulate an entire collection of objects if the objects share common methods. For example, the following procedure sets the dwell time on all of the segments in the collection.

```
Sub setDwell()  
For Each seg In segs  
    segs.DwellTime = 0.03  
Next  
End Sub
```

## Methods

A method is an action that is performed on an object. For example, **CreateSParameter** is a method on the [Application](#) object. The following procedure uses that method to create a new S21 measurement in channel 1 in a new window.

```
Sub CreatMeas  
app.CreateSParameter 1,2,1,1  
End Sub
```

## Properties

A property is an attribute of an object that defines one of the object's characteristics, such as size, color, or screen location. A property can also change an aspect of the object's behavior, such as whether the object is visible. In either case, to change the characteristics of an object, you change the values of its properties.

For example, the following statement sets the IF Bandwidth of a channel to 1 KHz.

```
Chan.IFBandwidth = 1e3
```

You can also read the current value of a property. The following statement reads the current IF Bandwidth of a channel into the variable **ifbw**.

```
Ifbw = Chan.IFBandwidth
```

Some properties cannot be set and some cannot be read. The Help topic for each property indicates if you can:

- Set and read the property (Write/Read)
- Only read the property (Read-only)
- Only set the property (Write-only)

## Events

A COM event is an action recognized by an object, such as clicking the mouse or pressing a key. Using events, your program can respond to a user action, program code, or triggered by the analyzer.

The SCPI equivalent of an event is a Service Request (SRQ).

For example:

```
OnChannelEvent
```

For more information, see [Working with the Analyzer's Events.](#)

## Getting a Handle to an Object

---

The following are discussed in this topic:

- [What Is a Handle](#)
- [Declaring an Object Variable](#)
- [Assigning an Object Variable](#)
- [Navigating the Object Hierarchy](#)
- [Getting a Handle to a Collection](#)

### [Other Topics about COM Concepts](#)

---

#### What Is a Handle

In SCPI programming, you must first select a measurement before changing or reading settings. With COM, you first get a handle to the object (or collection) and refer to that object to change or read its settings. The following analogy illustrates this:

A CAR could be called an object. More precisely, CAR is a class of objects. For example, one of the properties of the CAR class is "**Color**". You can read (by looking) or set (by painting) the color property of A car object. In other words, you can only read or set properties of a specific car object; not the entire car class. Therefore, to read or set a property, you need to get "a handle", or an instance of the object.

This process is also called "accessing an object", "getting an instance of an object", "returning an object". or "referring to an object". You can have handles to many instances of an object at the same time.

#### Accessing PNA Objects

The PNA Application object is the highest object in the PNA object model hierarchy. Because of that, it is the only object that must be 'created' before it, or any other objects, can be accessed and used. During the creation process, the application object assigned to a variable name, or handle. Throughout your program, that object is used by referring to that variable. All PNA objects can be assigned to a variable, and subsequently referred to, in this same manner.

The following example shows how to create the PNA Application object, as well as illustrate the general steps of get a handle to an object.

There are two steps in the process of getting a handle to analyzer objects:

1. Declaring a Variable
2. Assigning an Object to the Variable

#### 1. Declaring a Variable

**Note:** The examples in these topics use the Visual Basic Programming Language. See the short section regarding [Visual Basic syntax](#).

Use the Dim statement or one of the other declaration statements (Public, Private, or Static) to declare a variable.

The type of variable that refers to an object must be a Variant, an Object, or a specific type of object. Some programming languages, such as VBScript and Agilent VEE, do not allow you to specify variable types.

The following examples ALL declare the variable **pna**. Each subsequent statement is more specific than the previous:

- `Dim pna 'Variant data type.`
- `Dim pna As Object 'Object data type.`
- `Dim pna As AgilentPNA835x.Application ' Specific Application type`
- `Dim pna As AgilentPNA835x.IApplication ' Interface type`

1. If you use a variable without declaring it first, the data type of the variable is Variant. If you don't care about using automatic type checking, and willing to run code less efficiently, this method is very safe and is useable on all programming environments.
2. If you know the specific object type, and your programming environment allows it, you can declare the variable as an object.
3. Declaring a specific object type provides automatic type checking (Intellisense), faster code, and improved readability.
4. Declaring the interface is the most specific way and is beneficial when developing code for multiple firmware revisions. [Learn more about Interfaces.](#)

## 2. Assigning an Object to a Variable

To assign an object instance to a variable, use the **Set** keyword before the object variable that was declared previously. In the following line of code, we SET the current AgilentPNA835x Application to "pna".

```
Set pna = AgilentPNA835x.Application
```

As mentioned earlier, the AgilentPNA835x object is unique because it is the highest level of object in the PNA object model hierarchy. Therefore, we must use the **CreateObject** keyword with the (*classname,server name*) parameters.

- The **classname** for the analyzer object is always "AgilentPNA835x.Application".
- To find your analyzer's **server name**, see [View or change full computer name](#)

The following statements create an instance of the Analyzer object.

```
Dim pna AS AgilentPNA835x.Application
Set pna = CreateObject("AgilentPNA835x.Application", "Analyzer46")
```

**Note:** These statements will start the PNA application if it is not already running on your instrument.

## Navigating the Object Hierarchy

Once an instance of the PNA Application is "created", you access all of the PNA objects by navigating the object hierarchy. Navigating the object model hierarchy can be tricky. In addition, you also need to know how to refer to a specific instance of that object. For example, if you have three measurements present on the PNA, how do you refer to the channel 1 measurement? Each object on the PNA Object Model image is linked to an object page. At



the top of each object page is a **Description** section and another called "**Accessing the ... Object**". These sections together explain how to navigate the PNA hierarchy to access a specific instance of that object.

From the previous discussion, you may think that you must always declare and assign variables to an object before setting or reading its properties. While this method is best for objects that you will continue to reuse, such as a measurement, it is not always necessary. You can also refer to an object directly.

The [TriggerSetup](#) object, which is a child of the Application object. Because we will only need to refer to this object once to set a couple of properties, and it is easy to access, we will refer to it directly. From the previous example, we already have a handle to the Application object in the variable **pna**. The following example uses [Visual basic 'dot' notation](#) to refer to the TriggerSetup object, and then the Scope property.

```
pna.TriggerSetup.Scope = naChannelTrigger
```

By referring to the TriggerSetup object directly, we must type the same path whenever we refer to properties on the TriggerSetup object. The following method assigns the pna.TriggerSetup object to a variable that can be reused.

```
Dim trig As Object  
Set trig = pna.TriggerSetup
```

Once created, you can treat an object variable exactly the same as the object to which it refers. For example:

```
trig.Scope = naChannelTrigger  
trig.Source = naTriggerSourceInternal
```

### Getting a Handle to a Collection

The analyzer has several collections of objects which provide a convenient way of setting or reading all of the objects in the collection with a single procedure. Also, there are objects (limit lines for example) that can only be accessed through the collection.

To get a handle to an item in a collection, you can refer to the object by item number or sometimes by name. However, you first have to get a handle to the collection. To assign the collection to a variable, use the same two step process (1. declare the variable, 2. assign the variable using 'Set').

```
Dim meass As Measurements
```

```
Dim meas As Measurement
```

You can then iterate through the entire collection of measurements to read or set properties

```
Sub setFormat()  
For Each meas In meass  
    meas.Format = naDataFormat_LinMag  
Next  
End Sub
```

Or you can read or set a property on an individual object in the collection:

```
meass(1).Format = naLinMag
```

**Note:** Each object and collection has its own unique way of dealing with item names, and numbers. Refer to the [Analyzer Object Model](#) for details.

## Collections in the Analyzer

---

Collections are a gathering of similar objects. They are a convenience item used primarily to iterate through the like objects in order to change their settings. Collections generally provide the following generic methods and properties:

```
Item(n)
Count
Add(n)
Remove(n)
```

where **(n)** represents the number of the item in the collection. Some collections may have unique capabilities pertinent to the objects they collect.

---

### [Other Topics about COM Concepts](#)

---

#### Collections are Dynamic

**A collection does not exist until you ask for it.** When you request a Channels object (see Getting a Handle to an Object / [Collection](#)), handles to each of the channel objects are gathered and placed in an array.

For example, if channels 2 and 4 are the only channels that exist, then the array will contain only 2 items. The command 'channels.Count' will return the number 2, and:

- Channels(1) will contain the channel 2 object.
- Channels(2) will contain the channel 4 object.

**The ordering of objects within the collection should not be assumed.** If you add a channel to the previous example, as in:

```
Pna.Channels.Add(3)
```

'channels.Count' will now return 3 and:

- Channels(1) will contain the channel 2 object.
- Channels(2) will contain the channel 3 object.
- Channels(3) will contain the channel 4 object.

Primarily, collections are useful for making this type of iteration possible:

```
Dim ch as Channel
For each ch in pna.Channels
    Print ch.Number
    Print ch.StartFrequency
    Print ch.StopFrequency
Next ch
```

As soon as this for-each block has been executed, the Channels object goes out of scope.

---

## COM Data Types

---

The PNA uses several data types to communicate with the host computer. Before using a variable, it is best to declare the variable as the type of data it will store. It saves memory and is usually faster to access. The following are the most common data types:

- [Long Integer](#)
- [Single Precision \(Real\)](#)
- [Double Precision \(Real\)](#)
- [Boolean](#)
- [String](#)
- [Object](#)
- [Enumeration](#)
- [Variant](#)

---

### Other Topics about COM Concepts

---

**Long** (long integer) variables are stored as signed 32-bit (4-byte) numbers ranging in value from -2,147,483,648 to 2,147,483,647.

---

**Double** (double-precision floating-point) variables are stored as IEEE 64-bit (8-byte) floating-point numbers ranging in value from -1.79769313486232E308 to -4.94065645841247E-324 for negative values and from 4.94065645841247E-324 to 1.79769313486232E308 for positive values.

---

**Single** (single-precision floating-point) variables are stored as IEEE 32-bit (4-byte) floating-point numbers, ranging in value from -3.402823E38 to -1.401298E-45 for negative values and from 1.401298E-45 to 3.402823E38 for positive values.

---

**Boolean** variables are stored as 16-bit (2-byte) numbers, but they can only be True or False. Use the keywords True and False to assign one of the two states to Boolean variables.

When other numeric types are converted to Boolean values, 0 becomes False and all other values become True. When Boolean values are converted to other data types, False becomes 0 and True becomes -1.

In PNA release 5.26, the following properties were changed to return True rather than 1 to conform with this definition. This change may affect the functionality of your COM program:

- [Bandwidth Tracking Property](#)
- [ErrorCorrection Property](#)
- [IFGateEnable Property](#)

- [Interpolate Correction Property](#)
- [LimitTestFailed Property](#)

---

**String** variables hold character information. A String variable can contain approximately 65,535 bytes (64K), is either fixed-length or variable-length, and contains one character per byte. Fixed-length strings are declared to be a specific length. Variable-length strings can be any length up to 64K, less a small amount of storage overhead.

---

**Object** variables are stored as 32-bit (4-byte) addresses that refer to objects within the analyzer or within some other application. A variable declared as Object is one that can subsequently be assigned (using the Set statement) to refer to any actual analyzer object.

---

**Enumerations (Enum)** are a set of named constant values. They allow the programmer to refer to a constant value by name instead of by number. For example:

```
Enum DaysOfWeek
    Sunday = 0
    Monday = 1
    Tuesday = 2
    Wednesday = 3
    Thursday = 4
    Friday = 5
    Saturday = 6
End Enum
```

Given this set of enumerations, the programmer can then pass a constant value as follows:

```
SetTheDay(Monday)
```

rather than

```
SetTheDay(1)
```

where the reader of the code has no idea what the value 1 refers to.

However, the analyzer RETURNS a long integer, not the text.

```
Day = DaysOfWeek(today) 'Day = 1
```

---

**Variant** - If you don't declare a data type ("typed" data) the variable is given the Variant data type. The Variant data type is like a chameleon — it can represent many different data types in different situations.

The PNA provides and receives Variant data because there are programming languages that cannot send or receive "typed" data. Variant data transfers at a slower rate than "typed" data.

---

## PNA Interfaces

---

A COM interface is the connection to an object. When you [get a handle to an object](#), you are actually using an interface to an object. This subtle distinction is relevant to the COM programmer for the following two reasons:

- [Interface Inheritance \(Coding for Multiple PNA Versions\)](#)
- [Custom Interfaces.](#)

### [Other Topics about COM Concepts](#)

#### Interface Inheritance (Coding for Multiple PNA Versions)

The PNA continues to evolve and release [new firmware / software versions](#) that provide more functionality and features. New commands are added to existing objects, and with them new interfaces are added to support those commands. For example, new commands were added to the Measurement object in PNA release 3.0. These commands are accessible from the new IMeasurement2 interface. This can be important if you develop code using the type library in release 3.0, and run the code on a PNA with an older release, such as 2.0

When you use a command that was new with release 3.0, and you run that code on a PNA with release 2.0 firmware, errors will occur because that PNA does not recognize the new commands. However, even if you do NOT utilize new commands, errors can still occur. The following example shows how this occurs and how to avoid it.

The following Visual Basic statement dimensions the **meas** variable as an object.

```
Dim meas As Measurement
```

When the program compiles, Visual Basic figures out what interface to use to access that object. When dimensioning as an object, VB will use the default interface. As new interfaces are added to an object, they become the default interface. If this program was developed and compiled using the PNA 3.0 type library, the default Interface of the Measurement Object was IMeasurement2. However, if this program is run on an instrument with PNA 2.0 firmware, there was no IMeasurement2 Interface, and an E\_NOINTERFACE error will occur.

Therefore, the more robust approach would be to specify the interface instead of the object when declaring a variable.

```
Dim meas As IMeasurement
```

This code will ONLY use the IMeasurement interface; not the default interface.

However, regardless of how you declare a variable, errors will always occur if you use new commands, and run the code on an older instrument.

#### Custom Interfaces

The PNA object model contains three "custom" interfaces use "typed" variables, which is more efficient than using variant type variables. However, these interfaces are only usable from VB6, C, & C++. All other programming languages must use the other standard interfaces.

The custom interfaces are:

- [IArrayTransfer](#) - Measurement object
- [ICalData](#) - Calibrator object

- [ISourcePowerCalData](#) - Channel object

## Working with Events

---

- [What are Events?](#)
- [Using the Analyzer's Events](#)
- [Event ID's](#)
- [Filtering Events](#)
- [List of Events](#)
- [Out of Range Errors](#)
- [Troubleshooting Problems with Events](#)

### See Also

[Events Example](#)

[Errors and the SCPIStringParser Object](#)

### [Other Topics about COM Concepts](#)

### What are Events?

Windows applications work from user-initiated events such as mouse moves and mouse clicks. A mouse-click produces an event that the programmer can either ignore or "handle" by providing an appropriate subroutine like this:

```
Sub DoThis_onClick
    Perform something
End Sub
```

If this subroutine were in your program and the mouse-click event occurs on your PC, it would generate a "Callback" to the client and interrupt whatever it was doing and handle the event.

A more practical example of an event in the analyzer is Limit test. If limit test is on and the measurement fails, the analyzer produces a "Limit-failed" event. If the measurement passed, the analyzer produces a "Limit-succeeded" event.

The Analyzer has a very sophisticated Event structure. Your program **CAN** be notified when one or more events occur. However, it may not be necessary.

For example, the analyzer has an event that will notify your program when a sweep is complete. A simpler alternative is to use a synchronous command which waits for the sweep to complete.

```
sync = True
app.ManualTrigger sync
chan.StartFrequency = 4.5E6
```

This would NOT work if you want the controller to do other things while waiting, like setup a power meter or sort some data. In this case you would like a "callback" from the analyzer to let your program know that the sweep has completed. For an example of this see [Events Example](#).





- [AllowEventMessage](#) - monitor a specific event
- [AllowAllEvents](#) - monitor ALL events
- [DisallowAllEvents](#) - monitor NO events
- [AllowEventCategory](#) - monitor specific event categories (discussed later)
- [AllowEventSeverity](#) - monitor events having one or more of the following severity levels associated with them.

Code	Severity Enumeration
00	naEventSeveritySUCCESS - the operation completed successfully
01	naEventSeverityINFORMATIONAL - events that occur without impact on the measurement integrity
10	naEventSeverityWARNING - events that occur with potential impact on measurement integrity
11	naEventSeverityERROR - events that occur with serious impact on measurement integrity

### List of Events

The following is a list of categories and the general types of events they include. Click the link view the event details.

Category Enumeration	Callback
naEventCategory_PARSER	<a href="#">OnSCPIEvent</a>
naEventCategory_MEASURE	<a href="#">OnMeasurementEvent</a>
naEventCategory_CHANNEL	<a href="#">OnChannelEvent</a>
naEventCategory_HW	<a href="#">OnHardwareEvent</a>
naEventCategory_CAL	<a href="#">OnCalEvent</a>
naEventCategory_USER	<a href="#">OnUserEvent</a>
naEventCategory_DISPLAY	<a href="#">OnDisplayEvent</a>
naEventCategory_GENERAL	<a href="#">OnSystemEvent</a>

**Note:** Use the [MessageText](#) Method to get a text message describing the event.

### Out of Range Errors

When you attempt to set a value on an active function that is beyond the range (min or max) of the allowable values, the analyzer limits that value to an appropriate value (min or max) and sets the function to the limited value. From the front panel controls this is visually evident by the limited value in the edit box or by the annotation on the display. An example would be attempting to set the start frequency below 300kHz. The edit control doesn't allow the number to fall below 300kHz.

When the automation user programs a setting (such as start frequency below the allowable limits) the same behavior takes place. The analyzer accepts the limited value. However, in order to learn what setting took place, you have to read the HRESULT.

All automation calls return HRESULTs. By default the HRESULT returned when an overlimit occurs is S\_NA\_LIMIT\_OUTOFRANGE. This value is a success code, meaning that bit 31 in this 32 value is 0. Programmers should check the return code from all automation calls to determine success or failure.

Some C++ macros (like SUCCEEDED(hr) or FAILED(hr) ) only check bit 31. So if you are interested in trapping this outOfRange error you will have to check for S\_NA\_LIMIT\_OUTOFRANGE explicitly.

Alternatively, you can configure the analyzer to report outOfRange conditions with an error code. Use the method: App.[SetFailOnOverRange](#) (true). With this method set TRUE, any overrange error will return E\_NA\_LIMIT\_OUTOFRANGE\_ERROR.

This method is provided for the benefit of VB clients. VB users can't detect specific success codes because the VB runtime strips off the HRESULT and only raises a run time error if bit 31 is set, indicating a fail code.

---

### **Troubleshooting Problems with Callbacks**

When you do callbacks, the client PC becomes the server and the analyzer (server) becomes the client. Callbacks can only take place when both server and client are in the same workgroup or in the same domain. See [Configure for COM](#).

## Read and Write Calibration Data using COM

---

Calibration data in the PNA is stored in Cal Sets. [Learn more about Cal Sets](#)

You can read or write two types of Calibration data:

- **Error Terms** - calculated data using standard measurement data and the algorithms for the specified cal type.
- **Standard Measurement data** -raw data resulting from the measurement of a calibration standard.

Each of these data are available in the PNA in either variant data or typed data. [Learn more about variant and typed data](#)

### Other Topics about COM Concepts

#### Calibration / Cal Set Interfaces

There are several interfaces associated with Calibration.

##### [ICalibrator](#)

This interface is the original interface provided with the first version of the PNA. It provides remote access to the "Unguided" Calibration wizard. This interface can perform 1 and 2 port calibrations as well as response cals.

This interface can also read and write error terms from/to a Cal Set. However, ICalibrator is NOT recommended for this purpose. The [ICalSet2](#) Interface is better suited for reading and writing error terms.

See a vbscript example of [how to perform a 2-port Cal and read the cal data](#).

##### [IGuidedCalibration](#)

This interface provides the methods and properties used by the Guided Calibration wizard. With this interface you can perform multi-port calibrations (1 to 4 port cals), but no response cals.

##### [ICalSet2](#) and [ICalData3](#)

These interfaces provide access to the Cal Set contents. You can read and write error terms with both of these interfaces.

- ICalSet2 uses Variant data, which means it is usable from vbscript.
- ICalData3 uses "typed" data, which means it can be used from any automation engine that can read the type library (VEE, VB, C++, etc.). Typed arguments (such as float or single) are more efficient than variants, so use the ICalData3 interface where better performance is needed.

[See a vbscript example of how to read Cal Set data](#).

##### [ICalSet3](#)

This interface provides access to the stimulus attributes of the Cal data: frequency, power, number of points. These are the stimulus conditions under which the Cal Set was created.



## Programming the PNA with C++

---

The programming information contained in this Help system is aimed at the Visual Basic programmer. VB does a lot of work for the programmer when it comes to managing and accessing components. Using a lower level language like C++ requires a more thorough understanding of the underlying tenets of COM. It is not the intent of this section to teach COM programming. The following is intended to acquaint you with some of the basic concepts you need to know in order to program against COM.

- [Initializing COM](#)
- [Importing the Type Library](#)
- [Creating the Application Object](#)
- [Errors](#)
- [Events](#)
- [Additional Reading](#)
- [Example](#)

**Note:** The information in this section assumes development on a Windows OS using Microsoft tools.

### [Other Topics about COM Concepts](#)

---

#### Initializing COM

The first thing you must do before performing any COM transactions is to initialize the COM library. You can do this in a number of ways. The most basic of these is a call to **CoInitialize( )** or **CoInitializeEx( )**. Alternatively you can use the MFC (Microsoft Foundation Classes) **AfxOleInit( )**.

Conversely, before your program exits you must uninitialize COM. You can accomplish this with **CoUninitialize( )** or the MFC routine **AfxOleTerm( )**.

---

#### Importing the Type Library

To make a component available to the client, the server exports what is called the type library. For the PNA, this file is 835x.tlb. It is located on the PNA's hard drive at **C:/Program Files/ Agilent/ Network Analyzer/ Automation**. See [Configure for COM-DCOM Programming](#).

The type library can be read and deciphered using another COM interface called ITypeLib. VB uses this interface to present, for example, its object browser. Visual C++ can also read type libraries. This is done by importing the type library into your project with a compiler directive:

```
#import "C:/Program Files/Common Files/Agilent/Pna/835x.tlb", named_guids
```

When you compile your program with this statement in it, the compiler creates two other files: **835x.tlh** and **835x.tli**. The first is a header file that contains the type definitions for the PNA's COM interfaces and their methods. The second file contains inline functions that wrap the PNA's interface methods. The wrappers are beneficial in that they contain error reporting for each of the method calls.

The .tlh file defines a smart pointer which you can use to access the PNA's objects. The smart pointer definition

looks like this:

```
_com_smartptr_typedef(Iapplication, _uuidof(Iapplication))
```

A smart pointer is a term used for a C++ object that encapsulates a pointer used to refer to a COM object. All COM objects derive from the interface IUnknown. This interface has three methods: QueryInterface( ), AddRef( ), and Release( ). The function of the AddRef and Release methods is to maintain a reference count on the object and thus control the object's lifetime. Anytime you copy or create a reference to a COM object, you are responsible for incrementing its reference count. And likewise, when you are finished using that reference, it is your responsibility to Release it. Smart pointers do this work for you, as shown in the [example program](#). In addition, smart pointers will also perform the QueryInterface call when required. QueryInterface is a method that requests a specific interface from an object. In the example program we gain access to the IArrayTransfer interface of the Measurement object. In the ReadMethod routine, we see this:

```
PTransferData = pMeas;
```

The assignment operator is overloaded for the smart pointer and in reality, this simple statement does this:

```
HRESULT hr = pMeas->QueryInterface( IID_IArrayTransfer, (void**) &pTransferData);
```

Using the existing interface pointer (pMeas) to the object, this call asks the object if it supports the IArrayTransfer interface, and if so to return a pointer to it in pTransferData. Smart pointer makes life easier for the C++ programmer. Read more about smart pointers in Microsoft Developer's Network Library (*MSDN*).

---

## Creating the Application Object

The only createable object exported by the PNA is the [Application object](#). Typically this would be done with a call to CoCreateInstance:

```
STDAPI CoCreateInstance(  
    CLSID_IApplication, //Class identifier (CLSID) of the object  
    NULL, //Pointer to controlling IUnknown  
    CLSCTX_SERVER, //Context for running executable code  
    IID_IApplication, //Reference to the IID of the interface  
    (void*)&pNA //Address of output variable that receives  
    // the interface pointer requested in riid  
);
```

With the smart pointer, this is taken care of with the following call:

```
IApplicationPtr pNA; // declare the smart pointer  
pNA = IApplicationPtr("AgilentPNA835x.Application.1");
```

---

## Errors

All COM method calls are required to return an HRESULT. This is 32 bit long with a specific format.

- The most significant bit indicates success(0) or failure(1).
- The lower 16 bits indicate the specific failure.

Visual Basic strips off the returned HRESULT and raises an error object for non-successful returns. The C++ programmer must himself be diligent about handling errors. You must check the return value of each COM call to ensure its success.

---

## Events

The Application object sources the INetworkAnalyzerEvents interface. This object is the source for all events. To use events in C++, you must do two things:

1. Implement the INetworkAnalyzerEvents interface - derive an object from INetworkAnalyzerEvents and implement the methods described there.
2. Subscribe to the IconnectionPoint interface of the Application object. - obtain a pointer to the IConnectionPointContainer interface of the Application object and making the following request:

```
FindConnectionPoint( IID_INetworkAnalyzerEvents, &pConnection );
```

A successful call to this interface will return a valid pointer in pConnection. Use this pointer to subscribe to the Application object:

```
pConnect->Advise( IUnknown* punk, DWORD dwCookie);
```

This call provides the server object with a callback address. The Iunkown pointer in this call is the IUnkown pointer of the object that implements the INetworkAnalyzerEvents interface. This is the event sink. The application object needs a pointer to this object in order to call your interface when an event occurs. The **dwCookie** is your subscription key. Use it to unsubscribe (see Unadvise( ) ).

---

### Additional Reading

*"MSDN"* - Microsoft Developer's Network Library

*"Learning DCOM"*, by Thuan L. Thai, published by O'Reilly(1999)

*"Inside COM"*, by Dale Rogerson, published by Microsoft Press (1997)

*"Understanding ActiveX and OLE"*, by David Chappell, also published by Microsoft Press (1996)

*"Beginning ATL COM Programming"*, published by Wrox Press (1998)

---

### Example

The example uses the smart pointer created by Microsoft Visual Studio. The calls to CoInitialize and CoUninitialize open and close the COM libraries. In the example, notice that the pointers local to the main routine are explicitly released. When smart pointers go out of scope, they will perform this duty implicitly. However, we are calling CoUninitialize before they have the chance to be destroyed, so we are obliged to release them.

See the [example](#) program.

---

## Using COM from .NET

---

To communicate with the PNA from Microsoft .NET enabled languages such as C# and Visual Basic.NET perform the following steps:

1. [Configure your PC and PNA for COM-DCOM Programming](#).
2. Reference the type library within the development environment (see the following [exception for managed C++ projects](#).) In the process of referencing the type library, a .NET assembly is created that wraps the PNA type library with a .NET friendly interface. This .NET assembly is called an Interop Assembly.

**Exception for managed C++ projects:** To generate the Interop Assembly for managed C++ projects, you must use the `tlbimp.exe` utility. This utility is described in the MSDN documentation. On your PC, click Start then Run then type: `tlbimp.exe 835x.tlb` and click OK. After doing this you can use the `#using` directive to include the Interop Assembly on managed C++ projects.

### Example: Creating a .NET object from C#

The following is an example that shows how to create a .NET object that connects to the PNA over DCOM. In this example, *machineName* is either the DNS name or the IP address of the PNA to connect with.

```
Type pna = Type.GetTypeFromProgID("AgilentPNA835x.Application", machineName);
AgilentPNA835x.IApplication app =
(AgilentPNA835x.IApplication)Activator.CreateInstance(pna);
```

### See C# Example Programs:

[Perform a Guided Cal with C#](#)

[Using C#](#)

### Registering the PNA Primary Interop Assembly (PIA) (OPTIONAL)

The PIA is NOT necessary to communicate with the PNA. The following procedure is useful only when there are two .NET programs that want to share the same PNA interface definitions. Without the PIA, each .NET application would use its own Interop Assembly.

To register the PIA on a machine, you need to have the common language runtime (CLR) installed. This is included with Visual Studio.NET. Then perform the following steps:

**Note:** In the following steps, replace `<local directory>` with the full path name of the specified file on your PC.

1. Run the `PNAProxy.exe` program as described in [Configure for COM-DCOM Programming](#).
2. On the PNA, copy `C:/Program Files/Agilent/Network Analyzer/Automation/AgilentPNA835x.dll` to a local directory on your PC. Make a note of this directory.
3. On your PC, click **Start**, then **Run**, then type: `regasm <local directory> /AgilentPNA835x.dll` and click **OK** to register the dll.
4. Again, click **Start**, then **Run**, then type: `gacutil /i <local directory> /AgilentPNA835x.dll` and click **OK** to add the assembly to the Global Assembly Cache (GAC).



To **Uninstall the PIA**, perform the following:

1. On your PC, click **Start**, then **Run**, then type: **gacutil /u <local directory> /AgilentPNA835x.exe** and click **OK** to remove the assembly from the GAC.
2. On your PC, click **Start**, then **Run**, then type: **regasm /unregsite <local directory> / agilentpna835x.dll** and click **OK** to unregister the assembly.
3. To uninstall PNA Proxy.exe use the **Add/Remove Programs** utility in the control panel.

---

Last Modified:

2-Feb-2009    Updated by JE

## SCPI Command Tree

---

### See Also

- [Example Programs](#)
- [Find commands using a simulated PNA-X UI](#)
- [See list of all SCPI Errors.](#)
- [See Calibrating the PNA Using SCPI](#)
- [Synchronizing the PNA and Controller](#)
- [IEEE- 488.2 Common Commands](#)
- [Local Lockout](#)

---

<b>ABORt</b>	Stops all sweeps
<b>+ CALCulate</b>	Click to hide and show CALC branches
<b>:CORRection</b>	Electrical Delay and Phase Offset
<b>:CUSTom</b>	Custom measurements
<b>:DATA</b>	Sends and queries data.
<b>:EQUation</b>	Equation Editor
<b>:FILTer</b>	Time domain gating
<b>:FORMat</b>	Display format
<b>:FSIMulator</b>	Balanced measurements and Fixturing
<b>:FUNCTion</b>	Trace Statistics
<b>:GCData</b>	Read Gain compression data
<b>:GCMeas</b>	Gain Compression Analysis
<b>:GDElay</b>	Group Delay Aperture setting
<b>:LIMit</b>	Limit lines for pass / fail testing
<b>:MARKer</b>	Marker settings
<b>:MATH</b>	Math / Memory
<b>:MIXer</b>	X-axis display for FCA measurements
<b>:NORMalize</b>	Receiver power cal (Obsolete)
<b>:OFFSet</b>	Mag and Phase offset
<b>:PARAmeter</b>	Create and delete measurements
<b>:RDATa?</b>	Queries receiver data
<b>:SMOothing</b>	Point-to-point smoothing
<b>:TRANsform</b>	Time domain transform
<b>:X:VALues</b>	Returns X-Axis values for trace

<b>CONTRol</b>	Interface control and Rear-panel connector control.
<b>CSET:FIXTure</b>	Create a Cal Set that includes the effects of a fixture.
<b>DISPlay</b>	Display settings
<b>FORMat</b>	Format for data transfer
<b>HCOPY</b>	Hardcopy printing
<b>INITiate</b>	Continuous or manual triggering
<b>LXI</b>	LXI communications
<b>MMEMory</b>	Saves and recalls instrument states
<b>OUTPut</b>	Turns RF power ON and OFF
<b>ROUTE</b>	Controls internal switch to reference receiver. (Opt 81)

+ **SENSE** [Click to hide and show SENSE branches](#)

<b>:AVERage</b>	Sweep Averaging
<b>:BANDwidth</b>	IF Bandwidth
<b>:CLASs</b>	Returns measurement class name
<b>:CORRection</b>	Calibration and other correction settings
<b>:COUPle</b>	Chopped or Alternate sweep
<b>:FOM</b>	Frequency Offset (opt 080)
<b>:FREQuency</b>	Frequency sweep settings
<b>:GCSetup</b>	Gain Compression App (opt 086)
<b>:IF (E836xC)</b>	IF Access settings (opt H11)
<b>:IF (PNA-X)</b>	IF Access settings
<b>:IMD (PNA-X)</b>	Intermodulation Distortion (opt 087)
<b>:IMS (PNA-X)</b>	Intermodulation Spectrum (opt 087)
<b>:MIXer</b>	FCA measurements (opts 082 and 083)
<b>:MULTiplexer</b>	Controls external test sets.
<b>:NOISe (PNA-X)</b>	Noise Figure (opt 029)
<b>:PATH</b>	Provides access to hardware configuration
<b>:POWER</b>	Receiver attenuation and overpower protection
<b>:PULSe (PNA-X)</b>	Configure internal pulse generators
<b>:ROSCillator</b>	Returns the source of the reference oscillator.
<b>:SEGMeNt</b>	Segment sweep settings.
<b>:SWEep</b>	Sweep types
<b>:X:VALues</b>	Returns X-axis values for channel

<b>SOURce</b>	Source power to the DUT
<b>SOURce:POWer:CORR</b>	Source power Calibration
<b>STATus</b>	Reads the PNA status registers
<b>SYSTem</b>	Misc PNA capabilities
<b>SYSTem:CONF:EDEV</b>	Configure external devices
<b>TRIGger</b>	Trigger measurements

---

Last Modified:

13-Apr-2010	Added CSET (A.09.20)
23-Feb-2010	Added CALC:GDElay and LXI (A.09.20)
30-Jul-2009	Added syst:conf:edev and Calc:X?
16-Jan-2009	Moved class

## IEEE 488.2 Common Commands

---

[\\*CLS - Clear Status](#)

[\\*ESE - Event Status Enable](#)

[\\*ESE? - Event Status Enable Query](#)

[\\*ESR? - Event Status Enable Register](#)

[\\*IDN? - Identify](#)

[\\*OPC - Operation complete command](#)

[\\*OPC? - Operation complete query](#)

[\\*OPT? - Identify Options Query](#)

[\\*RST - Reset](#)

[\\*SRE - Service Request Enable](#)

[\\*SRE? - Service Request Enable Query](#)

[\\*STB? - Status Byte Query](#)

[\\*TST? - Result of Self-test Query](#)

[\\*WAI - Wait](#)

### See Also

- [Example Programs](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

### **\*CLS - Clear Status**

Clears the instrument status byte by emptying the error queue and clearing all event registers. Also cancels any preceding \*OPC command or query. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*ESE - Event Status Enable**

Sets bits in the standard event status enable register. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*ESE? - Event Status Enable Query**

Returns the results of the standard event enable register. The register is cleared after reading it. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*ESR - Event Status Enable Register**

Reads and clears event status enable register. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*IDN? - Identify**

Returns a string that uniquely identifies the analyzer. The string is of the form "Agilent Technologies",<model number>,<serial "number">,<software revision>" .

**Note:** Beginning with Rev 6.01, this command now returns the software revision with 6 digits instead of 4. For example, A.06.01.02.

---

### **\*OPC - Operation complete command**

Generates the OPC message in the standard event status register when all pending overlapped operations have been completed (for example, a sweep, or a Default). See [Understanding Command Synchronization](#).

---

### **\*OPC? - Operation complete query**

Returns an ASCII "+1" when all pending overlapped operations have been completed. See [Understanding Command Synchronization](#)

---

### **\*OPT? - Identify Options Query**

Returns a string identifying the analyzer option configuration.

---

### **\*RST - Reset**

Executes a device reset and cancels any pending \*OPC command or query, exactly the same as a [SYSTem:PRESet](#) with one exception: Syst:Preset does NOT reset [Calc:FORMAT](#) to ASCII. The contents of the analyzer's non-volatile memory are not affected by this command.

---

### **\*SRE - Service Request Enable**

Before reading a status register, bits must be enabled. This command enables bits in the service request register. The current setting is saved in non-volatile memory. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*SRE? - Service Request Enable Query**

Reads the current state of the service request enable register. The register is cleared after reading it. The return value can be decoded using the table in [Status Commands](#). See also [Reading the Analyzer's Status Registers](#).

---

### **\*STB? - Status Byte Query**

Reads the value of the instrument status byte. The register is cleared only when the registers feeding it are cleared. See [Status Commands](#) and [Reading the Analyzer's Status Registers](#).

---

### **\*TST? - Result of Self-test Query**

Returns the result of a query of the analyzer hardware status. An **0** indicates no failures found. Any other value indicates one or more of the following conditions exist. The value returned is the Weight (or sum of the Weights) of the existing conditions. For example:

- If **4** is returned from \*TST?, an **Overpower** condition exists.

- If **6** is returned, both **Unleveled** and **Overpower** conditions exists.

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	Phase Unlock	The source has lost phaselock. This could be caused by a reference channel open or a hardware failure.
1	2	Unleveled	The source power is unleveled. This could be a source is set for more power than it can deliver at the tuned frequency. Or it could be caused by a hardware failure.
2	4	Not used	
3	8	EE Write Failed	An attempted write to the EEPROM has failed. This is possibly caused by a hardware failure.
4	16	YIG Cal Failed	The analyzer was unable to calibrate the YIG. Either the phaselock has been lost or there has been a hardware failure.
5	32	Ramp Cal Failed	The analyzer was unable to calibrate the analog ramp generator due to a possible hardware failure.
6	64	Not used	

#### **\*WAI - Wait**

Prohibits the instrument from executing any new commands until all pending overlapped commands have been completed. See [Understanding Command Synchronization](#)

Last Modified:

17-Sep-2008    Added \*RST vs Syst:Pres note

## Abort Command

---

### ABORt

(Write-only) Stops all sweeps - then resume per current trigger settings. This command is the same as [INITtiate:IMMEDIATE](#) (restart) except if a channel is performing a single sweep, ABORt will stop the sweep, but not initiate another sweep.

Learn about [Synchronizing the PNA and Controller](#)

<b>Examples</b>	<b>ABOR</b> <b>abort</b>
<b>Query Syntax</b>	Not applicable
<b>Default</b>	Not applicable



## Calculate:Correction Commands

---

Controls error correction functions.

### CALCulate:CORRection

#### EDELay

| [DISTance](#)

| [TIME](#)

| [MEDium](#)

| [UNIT](#)

| [WGCutoff](#)

[[STATe](#)]

#### TYPE

#### OFFSet

| [[MAGNitude](#)]

| [PHASe](#)

Click on a [blue](#) keyword to view the command details.

[Red](#) keywords are superseded.

### See Also

- [Example Programs](#)
- New [Calibrating the PNA Using SCPI](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

**CALCulate<cnum>:CORRection:EDELay:DISTance <num>**

**(Read-Write)** Sets the electrical delay in physical length (distance) for the selected measurement.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<num> Electrical delay in distance.

First Specify units using [CALC:CORR:EDEL:UNIT](#)

Use [SENS:CORR:RVEL:COAX](#) <num> to set Velocity factor.

This parameter supports MIN and MAX as arguments. [Learn more.](#)

#### Examples

```
CALC1:CORR:EDEL:DIST 5
```

```
calculate2:correction:distance .003
```

**Query Syntax** CALCulate:CORRection:EDELay:DISTance?

**Return Type** Numeric

**Default** 0

---

### CALCulate<cnum>:CORRection:EDELay:MEDium <char>

**(Read-Write)** Sets the media used when calculating the electrical delay.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1.

<num> Choose from: **COAX** for coaxial medium, **WAVE**guide for waveguide medium.

#### Examples

```
CALC:CORR:EDEL:MED COAX
```

```
calc3:corr:edelay:medium waveguide
```

**Query Syntax** CALCulate<cnum>:CORRection:EDELay:MEDium?

**Return Type** Character

**Default** COAX

---

### CALCulate<cnum>:CORRection:EDELay:UNIT <char>

**(Read-Write)** Sets and returns the units for specifying electrical delay in physical length (distance).

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1.

<char> Units for delay in distance. Choose from:

- METer
- FEET
- INCH

**Examples**

```
CALC:CORR:EDEL:UNIT MET  
calc3:corr:edelay:unit inch
```

**Query Syntax** CALCulate<num>:CORRection:EDELay:UNIT?

**Return Type** Character

**Default** METer

---

**CALCulate<num>:CORRection:EDELay:TIME <num>**

**(Read-Write)** Sets the electrical delay for the selected measurement.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<num> Electrical delay in seconds. Choose any number between:  
**-10.00 and 10.00**

Use [SENS:CORR:RVEL:COAX](#) <num> to set Velocity factor.

This parameter supports MIN and MAX as arguments. [Learn more.](#)

**Examples**

```
CALC1:CORR:EDEL:TIME 1NS  
calculate2:correction:time 0.5e-12
```

**Query Syntax** CALCulate:CORRection:EDELay:TIME?

**Return Type** Numeric

**Default** 0 seconds

---

## CALCulate<cnum>:CORRection:EDELay:WGCutoff <num>

(Read-Write) Sets the waveguide cutoff frequency used when the electrical delay media is set to WAVEguide. (See [CALCulate:CORRection:EDELay:MEDium<char>](#).)

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1.  
<num> Waveguide cutoff frequency used with the electrical delay calculation.

This parameter supports MIN and MAX as arguments. [Learn more.](#)

### Examples

```
CALC:CORR:EDEL:WGC 18.067 GHz
```

```
calculate3:correction:edelay:wgcutoff 14.047 ghz
```

**Query Syntax** CALCulate<cnum>:CORRection:EDELay:WGCutoff?

**Return Type** Numeric

**Default** 45 MHz

---

## CALCulate<cnum>: CORRection:[STATe] <bool>

(Read-Write) Turns error correction ON or OFF for the selected measurement on the specified channel.

To turn error correction ON or OFF for a channel, use [SENS:CORR:STATe](#).

[See Critical Note](#)

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1  
<bool> Correction state. Choose from:

**0** - Correction OFF

**1** - Correction ON

### Examples

```
CALC:CORR ON
```

```
calculate:correction:state off
```

**Query Syntax** CALCulate<cnum>: CORRection: STATe?

**Return Type** Boolean

**Default** Not Applicable

---

## **CALCulate<cnum>:CORRection:TYPE <string>**

**(Read-Write)** Sets the Cal Type for the selected measurement on the specified channel. This is used when a Cal Set is applied. [Learn more about applying Cal Types.](#)

- Use [SENS:CORR:TYPE:CAT?](#) to list the Cal Types in the PNA.
- Use [SENS:CORR:CSET:TYPE:CAT?](#) to list the Cal Types contained in the active Cal Set for the channel.
- Use [SENS:CORR:COLL:METH](#) to set the Cal type to perform a new calibration,

### See Critical Note

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <string> **(String)** Cal type. Case sensitive. Use one of the following:

#### **For Full Calibrations:**

This command does not distinguish between TRL and SOLT. The same number of error terms is applied for both Cal Types.

"Full n Port(x,y,z...)"

where

n = the number of ports to calibrate

x,y,z = the port numbers to calibrate

For example:

```
"Full 7 Port (2,3,4,5,6,7,8) "
```

#### **For Response Calibrations:**

"Response(param)" OR

"ResponseAndIsolation(param)"

Where param =

- S-parameter. For example"

- ```
"Response (S21) "
```

- ```
"ResponseAndIsolation (A/R) "
```

- Single or ratioed receivers using either [logical receiver notation](#) or physical receiver notation. For example:

- `"Response (A) "`

- `"ResponseAndIsolation (a3/b4) "`

#### For FCA Calibrations:

- **"SMC\_2P"** (Response + Input + Output) All four sweeps required. Most accurate.
- **"SMCRsp+IN"** No Output match. All four sweeps required.
- **"SMCRsp+OUT"** No Output match. All four sweeps required.
- **"SMCRsp"** No Input or Output match. Saves two sweeps.

For VMC, multiple Cal types are not available.

#### For Gain Compression Cal

where r = receive port; s = source port

- "GCA 2P (r,s)" - full 2-port cal
- "GCA Enh Resp (r,s)" - Enhanced Response Cal

#### Examples

```
CALC:CORR:TYPE "Scalar Mixer Cal"
```

**Query Syntax** CALCulate<cnum>:CORRection:TYPE?

**Return Type** String

**Default** Not Applicable

---

**CALCulate<cnum>:CORRection:OFFSet[:MAGNitude] <num> Superseded**

**Note:** This command is replaced with [SENS:CORR:RPOWer:OFFSet\[:AMPLitude\]](#).  
To set data trace magnitude offset, use [CALC:OFFS:MAGN](#)  
This command does NOT function for FCA measurements.

See an example of a [Receiver Power Calibration](#).

(Read-Write)

**For Receiver Power Calibration**, specifies the power level to which the selected (unratioed) measurement data is to be adjusted. This command applies only when the selected measurement is of unratioed power.

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Cal power level in dBm. No limits are enforced on this value, but the PNA receivers themselves have maximum and minimum power specifications (that may differ between PNA models) which this value must comply with for a valid receiver power cal.

**Examples**

```
CALC:CORR:OFFS 10DBM  
calculat1:correction:offset:magnitude maximum
```

**Query Syntax** CALCulate<num>:CORRection:OFFSet[:MAGNitude]?

**Return Type** Numeric

**Default** 0dBm

---

**CALCulate<num>:CORRection:OFFSet:PHASe <num>[<char>]** **Superseded**

**Note:** This command is replaced with [CALC:OFFS:PHASe](#)

(Read-Write) Sets the phase offset for the selected measurement.

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Offset phase value. Choose any number between:  
**-360** and **360**
- <char> Units for phase. OPTIONAL. Choose either:  
**DEG** - Degrees (default)  
**RAD** - Radians

**Examples**

```
CALC:CORR:OFFS:PHAS 10  
calculate:correction:offset:phase 20rad
```

**Query Syntax** CALCulate:CORRection:OFFSet:PHASe?

**Return Type** Numeric, returned value always in degrees

**Default** 0 degrees

---

Last modified:

22-Sep-2009    Removed VMC from Corr:Type  
6-Feb-2009    Added two commands  
12-Feb-2008    Fixed typo  
9/12/06       MQ Modified Calc:Corr for multiport.



## Calculate:Custom Commands

---

Creates and modifies application measurements:

**CALCulate:CUSTom:**

[DEFine](#)

[MODify](#)

### See Also

- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### CALCulate<cnum>:CUSTom:DEFine <Mname>, <type> [,param]

**(Write-only)** Creates a custom measurement. The custom measurement is not automatically displayed. You must also do the following:

- Use [DISP:WIND:STATe](#) to create a window if it doesn't already exist.
- Use [DISP:WIND:TRAC:FEED](#) to display the measurement
- Select the measurement ([CALC:PAR:SEL](#)) before making additional settings.

See an example using this command to create a [VMC](#) and [SMC](#) measurement

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <Mname> Name of the measurement. Any non-empty, unique string, enclosed in quotes.
- <type> String. The type of custom measurement. Click each to see an example program. Choose from:
- ["Vector Mixer/Converter"](#)
  - ["Scalar Mixer/Converter"](#)
  - ["Noise Figure Cold Source"](#)
  - ["Noise Figure Converters"](#)
  - ["Gain Compression"](#)
  - ["Swept IMD"](#)
  - ["IM Spectrum"](#)

- "Swept IMD Converters"
- "IM Spectrum Converters"

[param] String. Optional parameter specifies the measurement to create:

Meas Class	Measurement Name	Description
Vector Mixer/Converter	"S11" "VC21" "S22"	<a href="#">Learn about VMC parameters</a>
Scalar Mixer/Converter	"S11" "SC21" "SC12" "S22" "Ipwr" "RevIPwr" "Opwr" "RevOPwr"	<a href="#">Learn about SMC parameters</a>
Gain Compression <a href="#">Learn more</a>	"CompIn21"  "CompOut21"  "CompGain21"  "CompS11"  "RefS21"  "DeltaGain21"  "S11", "S21", "S12", "S22"	Input power at the compression point.  Output power at the compression point.  Gain at the compression point.  Input Match at the compression point  Linear Gain  CompGain21 -Linear Gain  Standard S-parameters; measured at port 1 and port 2
	Noise Figure AND NFX:	
	"NF"	Noise figure
	"ENR"	Validate noise source measurements.

Noise Figure Cold Source

[Learn more](#)

Noise Figure Converters

[Learn more](#)

"T-Eff"	Effective noise temperature.
"DUTRNP" "DUTRNPI"	DUT noise power ratio. (Noise power expressed in Kelvin divided by 290).
"SYSRNP" "SYSRNPI"	System noise power ratio
"DUTNPD" "DUTNPDI"	DUT noise power density. (Noise power expressed in dBm/Hz).
"SYSNPD" "SYSNPDI"	System noise power density.
"OvrRng"	Indication that the noise receiver is being over powered.
"T-Rcvr"	Temperature reading (in Kelvin) of the noise receiver board.
<b>Noise Figure ONLY - NOT NFX:</b>	
"S11", "S21", "S12", "S22"	Standard S-parameters; measured with the port1 and port2 noise switches set for noise mode.
"A,1", "A,2" ...and so forth.	Unratioed parameters; with notation: "receiver, source port"
<b>NFX ONLY:</b>	
"S11" "SC21" "SC12" "S22" "Ipwr" "RevIPwr" "Opwr" "RevOPwr"	Mixer parameters

	<b>"ALO1"," BLO1" ...and so forth.</b>	Test port receiver at LO1 frequency
	<b>"R1_1", "B_2" ..and so forth.</b>	Unratioed parameters with notation: "receiver_source port"
Swept IMD Swept IMD Converters <a href="#">Learn more</a>	<p>There are over 150 possible Swept IMD parameters, too many to list here.</p> <p>Build the parameters with the <a href="#">Swept IMD Parameter</a> dialog, then copy the parameter name to the remote command.</p> <p>The following are a few example parameters:</p>	
	<b>"PwrMainLo"</b>	Absolute power of the Low tone at the DUT output.
	<b>"IM3"</b>	Power of the third product relative to the average power of the f1 and f2 tones measured at the DUT output.
IM Spectrum IM Spectrum Converters <a href="#">Learn more</a>	<b>"OIP3"</b>	Theoretical power level at which the third product will be the same power level as the average of the main tones at the output of the DUT.
	<b>"Output"</b>	View signals OUT of the DUT and into PNA port 2 (B receiver).
	<b>"Input"</b>	View signals IN to the DUT (R1 receiver).
	<b>"Reflection"</b>	View signals reflected off the DUT input and back into PNA port 1 (A receiver)

#### Examples

```
CALC:CUST:DEF 'My VC21', 'Vector Mixer/Converter', 'S22'
calculate2:custom:define 'MyNF', 'NoiseFigure', 'NF'
```

**Query Syntax** Not applicable

[Overlapped?](#) No

[Default](#) Not applicable

---

## CALCulate<cnum>:CUSTom:MODify <param>

(Write-only) Changes the selected custom measurement to a different parameter.

See an example using this command for a [VMC](#) and [SMC](#) measurement

### Parameters

<cnum> Channel of the custom measurement to be changed. First, select the measurement using [CALC:PAR:SEL](#).

<param> Parameter to change the custom measurement to. Select a parameter that is valid for the type of measurement. Choose from the same arguments as [Calc:Cust:Def](#).

### Examples

```
SYST:PRES  
CALC2:CUST:DEF 'My VC21', 'Vector Mixer/Converter'  
CALC:PAR:SEL 'My VC21'  
CALC2:CUST:MOD 'S22'
```

**Query Syntax** Not applicable

[Overlapped?](#) No

[Default](#) Not applicable

---

### Last Modified:

2-Mar-2010	Fixed IM spectrum converters
27-Feb-2009	Added IMD Converters and Noise
8-Sep-2008	Added IMD arguments
23-Aug-2007	Added Noise and GC Arguments

## Calculate:Data Commands

---

Controls writing and reading PNA measurement data.

<b>CALCulate:DATA</b>
<b>CUSTom</b>
<b>CATalog?</b>
<b>SNP?</b>
<b>PORTs?</b>
<b>SAVE</b>

Click on a [blue](#) keyword to view the command details.

**Red** is a superseded command.

### See Also

- [Example Programs](#)
- [Data Access Map](#)
- [Synchronizing the PNA and Controller](#)
- To read receiver data, use [CALC:RDATA?](#)
- To read error terms, use [SENS:CORR:CSET:DATA](#)
- To read SnP measurement data, use [CALC:DATA:SNP?](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

(Write) **CALCulate<cnum>:DATA <char>,<data>**

(Read) **CALCulate<cnum>:DATA? <char>**

Reads or writes Measurement data, Memory data, or Normalization Divisor data from the [Data Access Map](#) location.

- For Measurement data, use FDATA or SDATA
- For Memory data, use FMEM or SMEM. When querying memory, you must first store a trace into memory

using [CALC:MATH:MEMorize](#).

- For Normalization Divisor (Receiver Power Cal error term) data, use SDIV
- For write operations, data type depends on [FORM:DATA](#) command.

#### Equation Editor Notes:

- When equation editor is active on a trace in a standard S-parameter channel, Calc:Data returns the data from the parameter on the trace that was measured last. For example, for the equation "S22 + S33 + S11", then S33 is the last measured parameter because it uses source port 3.
- In [applications](#), if equation editor is active and the original parameter for the trace is not requested anywhere in the channel, then zeros are returned. If the original parameter is being measured within the channel, then data for the original parameter is returned.
- In general, if an equation contains no measurement parameters, then data for the original parameter is returned.

**Note:** The Calc:Data SCORR command to read / write error terms is **Superseded** with [SENS:CORR:CSET:DATA](#). SCORR commands do NOT accommodate greater than 12 error terms.

#### See Critical Note

#### Parameters

**<num>** Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**<char>** **FDATA** Formatted **measurement** data to or from [Data Access Map](#) location **Display** (access point 2).

- Corrected data is returned when correction is ON.
- Uncorrected data is returned when correction is OFF.
- Returns TWO numbers per data point for Polar and Smith Chart format.
- Returns one number per data point for all other formats.
- Format of the read data is same as the displayed format.

**SDATA** Complex measurement data.

**Writes** data to [Data Access Map](#) location **Raw Measurement** (access point 0).

- When writing corrected data, and correction is ON, it will be corrected again, resulting in meaningless data.

**Reads** data from **Apply Error Terms** (access point 1).

- Returns TWO numbers per data point.
- Corrected data is returned when correction is ON.
- Uncorrected data is returned when correction is OFF.

**FMEM** Formatted memory data to or from [Data Access Map](#) location **Memory result** (access point 4).

- Returns TWO numbers per data point for Polar and Smith Chart format.
- Returns one number per data point for all other formats.
- Format of the read data is same as the displayed format.
- Returned data reflects the correction level (On|OFF) when the data was stored into memory.

**SMEM** Complex measurement data to or from [Data Access Map](#) location **Memory** (access point 3).

- Returns TWO numbers per data point.
- Returned data reflects the correction level (On|OFF) when the data was stored into memory.
- Returned data reflects the correction level (On|OFF) when the data was stored into memory.

**SDIV** Complex data from [Data Access Map](#) location **Normalization (5)**.

- Returns TWO numbers per data point.
- If normalization interpolation is ON and the number of points changes after the initial normalization, the divisor data will then be interpolated.
- When querying the normalization divisor, you must first store a divisor trace using [CALC:NORMALize\[:IMMediate\]](#).

The following Calc:Data SCORR command to read / write error terms is **Superseded** with [SENS:CORR:CSET:DATA](#). These SCORR commands do NOT accommodate greater than 12 error terms.



For 2-Port SOLT and TRL calibrations	Specify this <char>	to get or put this Error Term...
	SCORR1	Forward Directivity
	SCORR2	Forward Source Match
	SCORR3	Forward Reflection Tracking
	SCORR4	Forward Isolation
	SCORR5	Forward Load Match
	SCORR6	Forward Transmission Tracking
	SCORR7	Reverse Directivity
	SCORR8	Reverse Source Match
	SCORR9	Reverse Reflection Tracking
	SCORR10	Reverse Isolation
	SCORR11	Reverse Load Match
	SCORR12	Reverse Transmission Tracking

## EXAMPLE

```
CALC:DATA FDATA,Data(x)
calculate2:data sdata,data(r,i)
```

See another [example](#) using this command.

**Return Type:** [Block data](#)

**Default** - Not Applicable

---

**CALCulate<cnum>:DATA:CUSTom <name>,<data>** **Superseded**

**Note:** This command has been replaced by [CALC:DATA:](#) which can now be used with all PNA applications.

**(Read-Write)** Reads or writes data from a custom-named measurement buffer.

[See Critical Note](#)

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <name> Name of the buffer to be read or written
- <data> Data to be read or written to the custom buffer. Format as one number per data point.

**Examples**

```
CALC:DATA:CUST 'VectorResult0',0,1,2,3,4,5 'Write  
CALC:DATA:CUST? 'VectorResult0' 'Read
```

**Query Syntax** CALCulate:DATA:CUSTom? <name>

**Return Type** Depends on [Form:Data](#)

**Default** Not Applicable

---

**CALCulate<num>:DATA:CUSTom:CATalog? Superseded**

**Note:** This command has been replaced by [CALC:DATA:CAT](#) which can now be used with all PNA applications.

**(Read-only)** Reads the list of buffer names (comma separated list of string values) available from the selected parameter. Specify the measurement using [CALCulate:PARameter:SElect](#).

[See Critical Note](#)

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

```
CALC:DATA:CUST:CAT?  
calculate:data:custom:catalog?
```

**Return Type** String

**Default** Not Applicable

---

## CALCulate<cnum>:DATA:SNP? <n> **Superseded**

**Note:** This command has been replaced by [CALC:DATA:SNP:PORTs?](#)

**(Read-only)** Reads SnP data from the selected measurement. [Learn more about SnP data.](#)

**Note:** This command returns SNP data without header information, and in columns, not in rows as .SnP files. This means that the data returned from this command sends all frequency data, then all Sx1 magnitude or real data, then all Sx1 phase or imaginary data, and so forth.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <n> Amount of data to return. If unspecified, <n> is set to 2. The number you specify must be less than or equal to the number of available ports on the PNA.

Choose from:

**1** (S1P) returns data for the active measurement.

**2** (S2P) returns data for the 2 port parameters associated with the current measurement. Default.

**3** (S3P) returns data for the 3 port parameters associated with the current measurement.

**4** (S4P) returns data for the 4 port parameters associated with the current measurement.

SnP data can be output using several data formatting options. See [MMEM:STORc:TRACc:FORMat:SNP](#).

See also [MMEM:STOR <file>.<snp>](#)

### Examples

```
CALC:PAR:DEF MyMeasurement, S11  
CALC:PAR:SEL MyMeasurement  
CALC:DATA:SNP? 1
```

**Return Type** Depends on [FORMat:DATA](#)

**Default** Not Applicable

---

**CALCulate<cnum>:DATA:SNP:PORTs? <"x,y,z">**

**Note:** This command replaces [CALC:DATA:SNP?](#). This command is more explicit regarding the data to be returned, and works for PNAs with multiport test sets.

**(Read-only)** Reads SNP data from the selected measurement for the specified ports. [Learn more about SnP data.](#)

**Note:** This command returns SNP data without header information, and in columns, not in rows as .SnP files. This means that the data returned from this command sends all frequency data, then all Sx1 magnitude or real data, then all Sx1 phase or imaginary data, and so forth.

[See Critical Note](#)

#### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<"x,y,z"> Comma or space delimited port numbers for which data is requested, enclosed in quotes.

SnP data can be output using several data formatting options. See [MMEM:STORe:TRACe:FORMat:SNP](#).

#### Examples

```
CALC:DATA:SNP:PORTs? "1,2,4,5,7" 'read data for these ports
```

#### Return Type

Depends on [FORMat:DATA](#)

#### Default

Not Applicable

---

### **CALCulate<cnum>:DATA:SNP:PORTs:SAVE <"x,y,z">,<filename>**

**Note:** This command replaces [MMEM:STOR sNP](#). This command is more explicit regarding the data to be saved, and works for PNAs with multiport test sets.

**(Write-only)** Saves sNP data from the selected measurement for the specified ports.

[Learn more about SNP data.](#)

[See Critical Note](#)

#### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<"x,y,z"> **String** Comma or space delimited port numbers for which data is requested, enclosed in quotes.

<filename> **String** Path, filename, and suffix of location to store the sNp data. The suffix is not checked for accuracy. If saving 2 ports, specify "filename.s2p"; If saving 4 ports, specify "filename.s4p.", and so forth.

SnP data can be output using several data formatting options. See [MMEM:STORe:TRACe:FORMat:SNP](#).

**Examples**

```
CALC:DATA:SNP:PORTs:Save '1,2,4','C:/Program  
Files/Agilent/Network Analyzer/Documents/MyData.s3p'
```

**Return Type** Depends on [FORMat:DATA](#)

**Default** Not Applicable

---

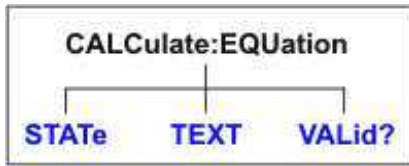
Last modified:

18-May-2009	Superseded Custom commands
24-Mar-2009	Edited SNP note
26-Apr-2007	Added clarification to Calc:Data SDATA
9/18/06	MQ Added two SNP Ports commands for multiport

## Calculate:Equation Commands

---

Controls Equation Editor capabilities.



Click on a [blue](#) keyword to view the command details.

### see Also

- [Example Programs](#)
- [Learn about Equation Editor](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### CALCulate<cnum>:EQUation:STATe <bool>

**(Read-Write)** Turns ON and OFF the equation on selected measurement for the specified channel. If the equation is not valid, then processing is not performed. Use [CALC:EQUation:VALid?](#) to ensure that the equation is valid.

[See Critical Note](#)

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <bool> **ON** (or 1) - turns equation ON.
- OFF** (or 0) - turns equation OFF.

#### Examples

```
CALC:EQU:STAT 1  
calculate2:equation:state 0
```

**Query Syntax** CALCulate<cnum>:EQUation:STATe?

**Return Type** Boolean

**Default** OFF (0)

---

### **CALCulate<cnum>:EQUation:TEXT <string>**

**(Read-Write)** Specifies an equation or expression to be used on the selected measurement for the specified channel.

[See Critical Note](#)

#### **Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<string> Any valid equation or expression. [See Equation Editor.](#)

#### **Examples**

**'Equation (includes '=')**

```
CALC:EQU:TEXT "foo=S11/S21"
```

**'Expression'**

```
calculate2:equation:text "S11/S21"
```

**Query Syntax** CALCulate<cnum>:EQUation:TEXT?

**Return Type** String

**Default** Not Applicable

---

### **CALCulate<cnum>:EQUation:VALid?**

**(Read-Write)** Returns a boolean value to indicate if the current equation on the selected measurement for the specified channel is valid. For equation processing to occur, the equation must be valid and ON ([CALC:EQU:STAT 1](#)).

[See Critical Note](#)

**Parameters**

<cnun> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnun> is set to 1.

**Examples**

```
CALC:EQU:VAL?
```

```
calculate2:equation:valid?
```

**Return Type**

Boolean

1 - equation is valid

0 - equation is NOT valid

**Default**

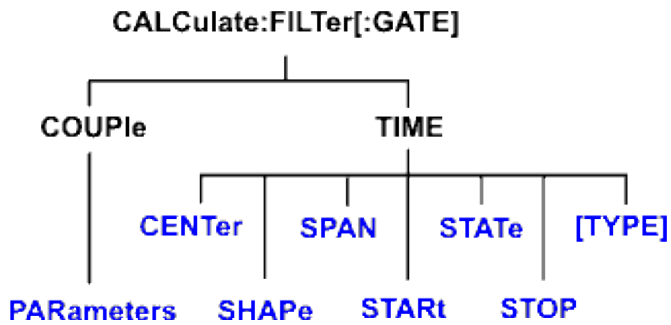
Not Applicable



## Calculate:Filter Commands

---

Controls the gating function used in time domain measurements. The gated range is specified with either (start / stop) or (center / span) commands.



Click on a [blue](#) keyword to view the command details.

### see Also

- [Example Programs](#)
- [Learn about Gating](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

**CALCulate<cnum>:FILTerf:GATEl:COUPLe:PARAMeters <num>**

---

**(Read-Write)** Specifies the time domain gating parameters to be coupled. The settings for those parameters will be copied from the selected measurement to all other measurements on the channel.

- To enable Trace Coupling, use [SENS:COUP:PAR](#)
- To specify Transform parameters to couple, use [CALC:TRAN:COUP:PAR](#)

Learn more about [Time Domain Trace Coupling](#)

[See Critical Note](#)

#### Parameters

- <cnun> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnun> is set to 1.
- <num> (Numeric) Parameters to couple. To specify more than one parameter, add the numbers.
- 1** - Gating Stimulus (Start, Stop, Center, and Span TIME settings.)
  - 2** - Gating State (ON / OFF)
  - 4** - Gating Shape (Minimum, Normal, Wide, and Maximum)
  - 8** - Gating Type (Bandpass and Notch)

#### Examples

**'To couple all parameters:**

```
CALC:FILT:COUP:PAR 15
```

**'To couple Stimulus and Type:**

```
calculate2:filter:gate:couple:parameters 9
```

**Query Syntax** CALCulate<cnun>:FILTer:GATE:COUPle:PARAmeters?

**Return Type** Numeric

**Default** 13 (All parameters except **2** - Gating State)

---

**CALCulate<cnun>:FILTer[:GATE]:TIME:CENTer <num>**

**(Read-Write)** Sets the gate filter center time.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Center time in seconds; Choose any number between:  
 $\pm (\text{number of points}-1) / \text{frequency span}$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
CALC:FILT:GATE:TIME:CENT -5 ns  
calculate2:filter:time:center maximum
```

**Query Syntax** CALCulate<num>:FILTer[:GATE]:TIME:CENTer?

**Return Type** Numeric

**Default** 0

---

### CALCulate<num>:FILTer[:GATE]:TIME:SHAPE <char>

**(Read-Write)** Sets the gating filter shape when in time domain.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <char> Choose from  
**MAXimum** - the widest gate filter available  
**WIDE** -  
**NORMal** -  
**MINimum** - the narrowest gate filter available

#### Examples

```
CALC:FILT:GATE:TIME:SHAP MAX  
calculate2:filter:time:shape normal
```

**Query Syntax** CALCulate<num>:FILTer[:GATE]:TIME:SHAPE?

**Return Type** Character

**Default** NORMal

## CALCulate<cnum>:FILTer[:GATE]:TIME:SPAN <num>

(Read-Write) Sets the gate filter span time.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Time span in seconds; Choose any number between:  
0 and  $2 * [(number\ of\ points - 1) / frequency\ span]$

**Note:** This command will accept MIN or MAX instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
CALC:FILT:GATE:TIME:SPAN 5 ns  
calculate2:filter:time:span maximum
```

**Query Syntax** CALCulate<cnum>:FILTer[:GATE]:TIME:SPAN?

**Return Type** Numeric

**Default** 20 ns

## CALCulate<cnum>:FILTer[:GATE]:TIME:STaTe <boolean>

(Read-Write) Turns gating state ON or OFF.

[See Critical Note](#)

**Note:** Sweep type must be set to LInear Frequency in order to use Transform Gating.

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <boolean> **ON** (or 1) - turns gating ON.  
**OFF** (or 0) - turns gating OFF.

### Examples

```
CALC:FILT:TIME:STAT ON  
calculate2:filter:gate:time:state off
```

**Query Syntax** CALCulate<cnum>:FILTer[:GATE]:TIME:STaTe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

## CALCulate<cnum>:FILTer[:GATE]:TIME:StARt <num>

(Read-Write) Sets the gate filter start time.

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<num> Start time in seconds; any number between:  
 $\pm (\text{number of points}-1) / \text{frequency span}$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
CALC:FILT:TIME:STAR 1e-8  
calculate2:filter:gate:time:start minimum
```

**Query Syntax** CALCulate<cnum>:FILTer[:GATE]:TIME:StARt?

**Return Type** Numeric

**Default** 10 ns

---

## CALCulate<cnum>:FILTer[:GATE]:TIME:StOP <num>

(Read-Write) Sets the gate filter stop time.

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<num> Stop time in seconds; any number between:  
 $\pm (\text{number of points}-1) / \text{frequency span}$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
CALC:FILT:TIME:STOP -1 ns  
calculate2:filter:gate:time:stop maximum
```

**Query Syntax** CALCulate<cnum>:FILTer[:GATE]:TIME:StOP?

**Return Type** Numeric

**Default** 10 ns

---

## CALCulate<cnum>:FILTer[:GATE]:TIME[:TYPE] <char>

(Read-Write) Sets the type of gate filter used.

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<char> Choose from:

**BPASs** - Includes (passes) the range between the start and stop times.

**NOTCh** - Excludes (attenuates) the range between the start and stop times.

### Examples

```
CALC:FILT:TIME BPAS  
calculate2:filter:gate:time:type notch
```

**Query Syntax** CALCulate<cnum>:FILTer[:GATE]:TIME[:TYPE]?

**Return Type** Character

**Default** BPAS

---

## Calculate:Format Commands

---

**CALCulate:**

**FORMat**

**UNIT**

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

### See Also

- [Example](#) using this command.
  - [Learn About Data Format](#)
  - [Synchronizing the PNA and Controller](#)
- 

## CALCulate<cnum>:FORMat <char>

(Read-Write) Sets the display format for the measurement.

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<char> Choose from:

- MLINear
- MLOGarithmic
- PHASe
- UPHase 'Unwrapped phase
- IMAGinary
- REAL
- POLar
- SMITH
- SADMittance 'Smith Admittance
- SWR
- GDElay 'Group Delay

- KELVin
- FAHRenheit
- CELSius

#### Examples

```
CALC:FORM MLIN
calculate2:format polar
```

**Query Syntax** CALCulate<cnum>:FORMat?

**Return Type** Character

**Default** MLINear

## CALCulate<cnum>:FORMat:UNIT <dataFormat>, <units>

**(Read-Write)** Sets and returns the units for the specified data format. Measurements with display formats other than those specified are not affected.

[See Critical Note](#)

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <dataFormat> Choose MLOG.
- <units> For unratioed measurements, choose from:
- **DBM** Units are displayed in dBm. 0 dBm = 0.001 watt
  - **DBMV** Units are displayed in dBmV. 0 dBmV = 0.001 volt

#### Examples

```
CALC:FORM MLOG, DBM
calculate2:format mlog, dbmv
```

**Query Syntax** CALCulate<cnum>:FORMat:UNIT? <dataFormat>

**Return Type** Character

**Default** DBM

Last Modified:

- 25-Aug-2008 Added Units
- 1-Oct-2007 Added temperature formats

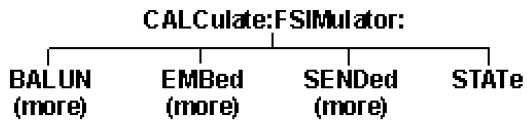




## Calculate:FSimulator Commands

---

Specifies settings and fixturing for Balanced Measurements.



Click a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
  - [SCPI Command Tree](#)
- 

### CALCulate<cnum>:FSIMulator:STATe <bool>

**(Read-Write)** Turns all three fixturing functions (de-embedding, port matching, impedance conversion) ON or OFF for all ports on the specified channel. Does not affect port extensions.

**Note:** This command affects ALL measurements on the specified channel.

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <bool> Choose from:
- ON or 1** - Turns Fixturing ON
- OFF or 0** - Turns Fixturing OFF

#### Examples

```
CALC:FSIM:STAT 1
calculate2:fsimulator:state 0
```

**Query Syntax** CALCulate<cnum>:FSIMulator:STATe?

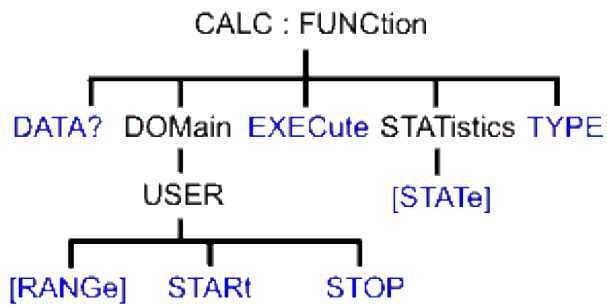
**Return Type** Boolean

**Default** OFF



## Calculate:Function Commands

---



Click on a [blue](#) keyword to view the command details.

### see Also

- [Example Programs](#)
- [Learn about Trace Statistics](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### CALCulate<cnum>:FUNCTION:DATA?

(Read-only) Returns the trace statistic data for the selected statistic type for the specified channel. Select the type of statistic with [CALC:FUNC:TYPE](#).

[See Critical Note](#)

#### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

**Return Type** Depends on [FORM:DATA](#)

**Example** CALCulate2:FUNCTION:DATA?

**Default** Not applicable

---

### CALCulate<cnum>:FUNCTION:DOMain:USER[:RANGe] <range>

**(Read-Write)** Sets the range used to calculate trace statistics. Each channel has 16 user ranges. The x-axis range is specified with the [CALC:FUNC:DOM:USER:START](#) and [STOP](#) commands.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<range> Range number. Choose from: **0** to **16**

**0** is Full Span of the current x-axis range

**1 to 16** are user-specified ranges

#### Examples

```
CALC:FUNC:DOM:USER 4  
calculate2:function:domain:user:range 0
```

**Query Syntax** CALCulate<num>:FUNCTION:DOMAIN:USER[:RANGE]?

**Return Type** Numeric

**Default** 0 - Full Span

---

### CALCulate<num>:FUNCTION:DOMAIN:USER:START <range>, <start>

**(Read-Write)** Sets the start of the specified user-domain range.

To apply this range, use [CALC:FUNC:DOM:USER](#)

To set the stop of the range, use [CALC:FUNC:DOM:USER:STOP](#).

[See Critical Note](#)

**Note:** This command does the same as [CALC:MARK:FUNC:DOM:USER:STAR](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<range> Range number that will receive the start value. Choose an integer between **1** and **16**

<start> Start value of the specified range. Choose a real number between: the analyzer's **Minimum** and **Maximum** x-axis value.

#### Examples

```
CALC:FUNC:DOM:USER:STAR 1,1e9  
calculate2:function:domain:user:start 2,2e9
```

---

<b>Query Syntax</b>	CALCulate<cnum>:FUNction:DOMain:USER:START? <range>
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	The analyzer's <b>Minimum</b> x-axis value

---

## **CALCulate<cnum>:FUNction:DOMain:USER:STOP <range>, <stop>**

**(Read-Write)** Sets the stop value of the specified user-domain range.

To apply this range, use [CALC:FUNC:DOM:USER](#).

To set the start of the range, use [CALC:FUNC:DOM:USER:START](#)

[See Critical Note](#)

**Note:** This command does the same as [CALC:MARK:FUNC:DOM:USER:STOP](#)

### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <range> Range number that will receive the stop value. Choose an integer between **1** and **16**
- <stop> Stop value of the specified range. Choose a real number between: the analyzer's **Minimum** and **Maximum** x-axis value.

### **Examples**

```
CALC:FUNC:DOM:USER:STOP 4,5e9
calculate2:function:domain:user:stop 3,8e9
```

<b>Query Syntax</b>	CALCulate<cnum>:FUNction:DOMain:USER:STOP? <range>
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	The analyzer's <b>Maximum</b> x-axis value

---

## **CALCulate<cnum>:FUNction:EXECute**

**(Write-only)** For the active trace of specified channel, executes the statistical analysis specified by the CALC:FUNC:TYPE command.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

```
CALC:FUNC:EXEC  
calculate2:function:execute
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**CALCulate<num>:FUNCTION:STATistics[:STATe] <ON|OFF>**

**(Read-Write)** Displays and hides the trace statistics (peak-to-peak, mean, standard deviation) on the screen.

The analyzer will display either measurement statistics or Filter Bandwidth statistics; not both.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<ON|OFF> ON - Displays trace statistics

OFF - Hides trace statistics

**Examples**

```
CALC:FUNC:STAT ON  
calculate2:function:statistics:state off
```

**Query Syntax** CALCulate<num>:FUNCTION:STATistics[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF (0)

---

**CALCulate<num>:FUNCTION:TYPE <char>**

(Read-Write) Sets statistic TYPE that you can then query using [CALC:FUNCtion:DATA?](#).

**Note:** In PNA releases 4.2 and prior, this command applied the statistic type to all measurements. Now, this command affects only the selected measurement on the specified channel.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<char> Choose from:

**PTPeak** - the difference between the max and min data points on the trace.

**STDEV** - standard deviation of all data points on the trace

**MEAN** - mean (average) of all data points on the trace

**MIN** - lowest data point on the trace

**MAX** - highest data point on the trace

#### Examples

```
CALC:FUNC:TYPE PTP  
calculate2:function:type stdev
```

**Query Syntax** CALCulate<num>:FUNCtion:TYPE?

**Return Type** Character

**Default** PTPeak

---



## Calc:GCData Commands

---

Reads Gain Compression data from the current 2D acquisition.

**CALCulate:GCData:**

[DATA](#)

[IMAG](#)

[ITERations](#)

[REAL](#)

Click on a [blue](#) keyword to view the command details.

### Other Gain Compression commands

The calibration commands listed in this topic are supplemental to the Guided Cal commands.

- [CALC:CUSTom:DEFine](#) - creates a gain compression measurement.
- [SENS:GCSetup](#) - Most Gain Compression settings.
- [CALC:GCMeas:ANAL](#) - Gain Compression Analysis settings
- Gain compression data can also be saved to a \*.csv file. [Learn how.](#)

### See Also

- [Example Programs](#)
- [Learn about Gain Compression Application](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

**CALCulate<ch>:GCData:DATA? <param>**

**(Read-Only)** Returns measurement data at all frequency and power data points for any GCA sweep. Previously 2D only.

The format of the data is the same as the format of the measurement that you select using [Calc:Par:Select](#). If the measurement is scalar, than one number is returned per data point. If complex (such as Smith Chart format) than both real and imaginary numbers are returned.

If correction is on, corrected data are returned. Otherwise, raw data are returned.

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <param> (String) Parameter to read. NOT Case-sensitive. The specified parameter need NOT be displayed or selected. Choose from:
- "pin" - input power at specified compression level.
  - "pout" - output power at specified compression level.
  - "gain" - device gain (S21) at specified compression level.
  - "inputmatch" - input match (S11)at specified compression level.
  - "DeltaGain" - Measured Gain (watts) / Ref Gain (watts). [Learn more.](#)
  - "AI1" and "AI2" - ADC measurements at the specified compression level.

[Learn more about GCA parameters.](#)

### Examples

```
data = CALC:GCD:DATA? "pin"
data = calculate:gcddata:data? "pout"
```

**Return Type** Array of data

**Default** Not Applicable

---

**CALCulate<ch>:GCDdata:IMAG? <char>, <dpoint>, <param>**

**(Read-Only)** Returns the imaginary part of the specified Gain Compression data. If correction is on, corrected data are returned. Otherwise, raw data are returned.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

- **FREQuency** - for the specified frequency data point, returns all of the measured data for each power stimulus.
- **POWer** - for the specified power data point, returns all of the measured data for each frequency stimulus.

<dPoint> Data point (FREQ or POWER) for which data is returned.

<param> (String) Parameter to read. NOT Case-sensitive. The specified parameter need NOT be displayed.

- "pin" - input power at each data point.
- "pout" - output power at each data point.
- "gain" - device gain (S21) at each data point.
- "inputmatch" - input match (S11) at each data point.

### Examples

For the fifth frequency data point, returns 'Power Output' imaginary (phase) data from all power stimulus values. If there are 30 power sweep points, 30 values are returned.

```
data = CALC:GCD:IMAG? FREQ, 5, "pout"
```

For the 30th stimulus power data point, returns 'Power Output' imaginary (phase) data from all frequency stimulus values. If there are 201 power sweep points, 201 values are returned.

```
data = calculate:gcddata:imag? power, 30, "pout"
```

**Return Type** Array of data

**Default** Not Applicable

---

**CALCulate<cnum>:GCDdata:ITERations?**

**(Read-only)** Returns the number of iterations made in a SMART sweep. Returns number of power points for a 2D sweep.

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

**CALC:GCD:ITER?**

**Return Type**

Numeric

**Default**

Not Applicable

---

**CALCulate<ch>:GCDData:REAL? <char>, <dpoint>, <param>**

**(Read-Only)** Returns the real part of the specified Gain Compression data. If correction is on, corrected data are returned. Otherwise, raw data are returned.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

- **FREQuency** - for the specified frequency data point, returns all of the measured data for each power stimulus.
- **POWer** - for the specified power data point, returns all of the measured data for each frequency stimulus.

<dPoint> Data point (FREQ or POWer) for which data is returned.

<param> (String) Parameter to read. NOT Case-sensitive. The specified parameter need NOT be displayed.

- **"pin"** - input power at each data point.
- **"pout"** - output power at each data point.
- **"gain"** - device gain (S21) at each data point.
- **"inputmatch"** - input match (S11) at each data point.

**Examples**

For the fifth frequency data point, returns 'Power Output' real data from all power stimulus values. If there are 30 power sweep points, 30 values are returned.

```
data = CALC:GCD:REAL? FREQ, 5, "pout"
```

For the 30th stimulus power data point, returns 'Power Output' real data from all frequency stimulus values. If there are 201 power sweep points, 201 values are returned.

```
data = calculate:gcddata:real? power, 30, "pout"
```

**Return Type**    Array of data

**Default**    Not Applicable

---

Last Modified:

- 30-Mar-2009    Added ADC and DeltaGain params
- 24-Apr-2008    Edited for any GCA sweep
- 18-Oct-2007    MX New topic

## Calc:Limit Commands

---

Controls the limit segments used for pass / fail testing.

### **CALCulate:LIMit:**

**DATA**

| **DELe**

**DISPlay**

| **[STATe]**

**FAIL?**

**SEGment**

| **AMPLitude**

| **STARt**

| **STOP**

| **STIMulus**

| **STARt**

| **STOP**

| **TYPE**

**SOUNd**

| **[STATe]**

**[STATe]**

Click on a [blue](#) keyword to view the command details.

### **see Also**

- [Example Programs](#)
- [Learn about Limit Lines](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one

measurement for each channel using [Calc:Par:Select](#)

---

## **CALCulate<cnum>:LIMit:DATA <block>**

(Read-Write) Sets data for limit segments.

[See Critical Note](#)

### **Parameters**

**<cnum>** Channel number of the measurement for which limit lines are to be set. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

**<block>** Data for all limit segments in REAL,64 format. The following is the data format for 1 segment:

**Type,BegStim, EndStim, BegResp,EndResp**

**Type** Type of limit segment. Choose from  
0 - Off  
1 - Max  
2 - Min

**BegStim** Start of X-axis value (freq, power, time)

**EndStim** End of X-axis value

**BegResp** Y-axis value that corresponds with Start of X-axis value

**EndResp** Y-axis value that corresponds with End of X-axis value

### **Examples**

The following writes three max limit segments for a bandpass filter.

```
CALC:LIM:DATA 1,3e5,4e9,-60,0,1,4e9,7.5e9,0,0,1,7.5e9,9e9,0,-30
```

**Query Syntax** CALCulate<cnum>:LIMit:DATA?

**Return Type** Depends on [FORM:DATA](#) - All 100 predefined limit segments are returned.

**Default** 100 limit segments - all values set to 0

---

## **CALCulate<cnum>:LIMit:DATA:DELeTe**

**(Write-only)** Deletes all limit line data for the selected measurement on the specified channel.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

```
CALC2:LIM:DATA:DEL
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**CALCulate<num>:LIMit:DISPlay[:STATe] <ON | OFF>**

**(Read-Write)** Turns the display of limit segments ON or OFF (if the data trace is turned ON).

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<ON | OFF> **ON** (or 1) - turns the display of limit segments ON.  
**OFF** (or 0) - turns the display of limit segments OFF.

**Examples**

```
CALC:LIM:DISP:STAT ON  
calculate2:limit:display:state off
```

**Query Syntax** CALCulate<num>:LIMit:DISPlay[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**CALCulate<num>:LIMit:FAIL?**



**(Read-only)** Returns the Pass / Fail status of the limit line test. Returns 1 (Fail) if any data point fails for any limit segment.

Limit display (CALC:LIM:DISP) does NOT have to be ON.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

```
CALC:LIM:FAIL?
```

**Return Type**

Boolean

- 0 is returned when **Pass**
- 1 is returned when **Fail**

**Default** Not Applicable

---

**CALCulate<num>:LIMit:SEGMent<snum>AMPLitude:STARt <num>**

**(Read-Write)** Sets the start (beginning) of the Y-axis amplitude (response) value.

[See Critical Note](#)

**Parameters**

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<snum> Segment number; if unspecified, value is set to 1.

<num> Choose any number between: **-500** and **500**

Display value is limited to the Maximum and Minimum displayed Y-axis values.

**Examples**

```
CALC:LIM:SEGM1:AMPL:STAR 10  
calculate2:limit:segment2:amplitude:start 10
```

**Query Syntax**

CALCulate<num>:LIMit:SEGMent<snum>AMPLitude:STARt?

**Return Type**

Numeric

**Default** 0

---

### **CALCulate<cnum>:LIMit:SEGMent<snum>AMPLitude:STOP <num>**

(Read-Write) Sets the stop (end) of the Y-axis amplitude (response) value.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <snum> Segment number; if unspecified, value is set to 1.
- <num> Choose any number between: **-500** and **500**

Display value is limited to the Maximum and Minimum displayed Y-axis values.

#### **Examples**

```
CALC:LIM:SEGM1:AMPL:STOP 10  
calculate2:limit:segment2:amplitude:stop 10
```

**Query Syntax** CALCulate<cnum>:LIMit:SEGMent<snum>AMPLitude:STOP?

**Return Type** Numeric

**Default** 0

---

### **CALCulate<cnum>:LIMit:SEGMent<snum>STIMulus:STARt <num>**

(Read-Write) Sets the start (beginning) of the X-axis stimulus value.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <snum> Segment number; if unspecified, value is set to 1.
- <num> Choose any number within the X-axis span of the analyzer.

#### **Examples**

```
CALC:LIM:SEGM1:STIM:STAR 10  
calculate2:limit:segment2:stimulus:start 10
```

**Query Syntax** CALCulate<cnum>:LIMit:SEGMent<snum>STIMulus:STARt?

**Return Type** Numeric

**Default** 0

---

### **CALCulate<cnum>:LIMit:SEGMent<snum>STIMulus:STOP <num>**

**(Read-Write)** Sets the stop (end) of the X-axis stimulus value.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <snum> Segment number; if unspecified, value is set to 1.
- <num> Choose any number within the X-axis span of the analyzer.

#### Examples

```
CALC:LIM:SEGM1:AMPL:STOP 10  
calculate2:limit:segment2:stimulus:stop 10
```

**Query Syntax** CALCulate<num>:LIMit:SEGMent<snum>STIMulus:STOP?

**Return Type** Numeric

**Default** 0

---

### CALCulate<num>:LIMit:SEGMent<snum>:TYPE <char>

**(Read-Write)** Sets the type of limit segment.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <snum> Segment number. Choose any number between:  
**1 and 100**  
If unspecified, value is set to 1.
- <char> Choose from:  
**LMAX** - a MAX limit segment. Any response data exceeding the MAX value will fail.  
**LMIN** - a MIN limit segment. Any response data below the MIN value will fail.  
**OFF** - the limit segment (display and testing) is turned OFF.

#### Examples

```
CALC:LIM:SEGM:TYPE LMIN  
calculate2:limit:segment3:type lmax
```

**Query Syntax** CALCulate<num>:LIMit:SEGMent<snum>:TYPE?

**Return Type** Character

**Default** OFF

---

### **CALCulate<num>:LIMit:SOUNd[:STATe] <ON | OFF>**

(Read-Write) Turns limit testing fail sound ON or OFF.

[See Critical Note](#)

#### **Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <ON | OFF> **ON** (or 1) - turns sound ON.  
**OFF** (or 0) - turns sound OFF.

#### **Examples**

```
CALC:LIM:SOUN ON  
calculate2:limit:sound:state off
```

**Query Syntax** CALCulate<num>:LIMit:SOUNd[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

### **CALCulate<num>:LIMit:STATe <ON | OFF>**

(Read-Write) Turns limit segment **testing** ON or OFF.

- Use [CALC:LIM:DISP](#) to turn ON and OFF the **display** of limit segments.
- If using [Global Pass/Fail](#) status, trigger the PNA AFTER turning Limit testing ON.

[See Critical Note](#)

#### **Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <ON | OFF> **ON** (or 1) - turns limit testing ON.  
**OFF** (or 0) - turns limit testing OFF.

#### **Examples**

```
CALC:LIM:STAT ON  
calculate2:limit:state off
```

**Query Syntax** CALCulate<num>:LIMit:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

Last Modified:

31-Oct-2008 Added Delete and Fail commands (8.33)

## Calculate:Marker Commands

---

Controls the marker settings used to remotely output specific data to the computer.

### CALCulate:MARKer:

**AOFF**

**BWIDth**

**COMPression**

| **LEVel**

| **PIN?**

| **POUT?**

**COUPling**

| **[STATe]**

**DELta**

**DISCrete**

**DISTance**

**FORMat**

**FUNction**

| **APEak**

| **EXCursion**

| **THReshold**

| **DOMain**

| **USER**

| **START**

| **STOP**

| **EXECute**

| **SElect**

| **TRACking**

**PNOP more commands**

## PSATuration more commands

### REference

[|\[STATe\]](#)

[|X](#)

[|Y?](#)

### SET

[\[STATe\]](#)

### TARGet

### TYPE

[X](#)

[Y?](#)

Click on a [blue](#) keyword to view the command details.

#### See Also

- [Marker example program](#)
- Marker Readout [number](#) and [size](#) commands.
- [Learn about Markers](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

**Note:** The Reference Marker is Marker Number 10

---

## CALCulate<cnum>:MARKer:AOff

(Write-only) Turns all markers off for selected measurement.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

#### Examples

```
CALC:MARK:AOFF  
calculate2:marker:aoff
```

**Query Syntax** Not applicable

**Default** Not applicable

---

### CALCulate<num>:MARKer:BWIDth <num>

(Read-Write) Turns on and sets markers 1 through 4 to calculate filter bandwidth. The <num> parameter sets the value below the maximum bandwidth peak that establishes the bandwidth of a filter. For example, if you want to determine the filter bandwidth 3 db below the bandpass peak value, set <num> to -3.

To turn off the Bandwidth markers, either turn them off individually or turn them [All Off](#).

The analyzer screen will show either Bandwidth statistics OR Trace statistics; not both.

To search a User Range with the bandwidth search, first activate marker 1 and set the desired [User Range](#). Then send the CALC:MARK:BWID command. The user range used with bandwidth search only applies to marker 1 searching for the max value. The other markers may fall outside the user range.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<num> Target value below filter peak. Choose any number between :-500 and 500

#### Examples

```
CALC:MARK:BWID -3  
calculate2:marker:bwid -2.513
```

**Query Syntax** CALCulate<num>:MARKer:BWIDth?  
Returns the results of bandwidth search:

**Return Type** Numeric - Four Character values separated by commas: bandwidth, center Frequency, Q, loss.



### **CALCulate<cnum>:MARKer<mkr>:COMPression:LEVel <num>**

(Read-Write) Set and read the marker compression level. A compression marker must already exist. Use [CALC:MARK ON](#) and [CALC:MARK:FUNC COMP](#) to create compression markers.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <num> Compression level. Choose any number between: -500 dB to 500 dB
- Standard gain compression values are positive.

#### **Examples**

```
CALC:MARK:COMP:LEV 1  
calculate2:marker:compression:level 1.5
```

**Query Syntax** CALCulate<cnum>:MARKer:COMPression:LEVel?

**Return Type** Numeric

Default +1

---

### **CALCulate<cnum>:MARKer<mkr>:COMPression:PIN?**

(Read-only) Reads the input power at the marker compression level. First send [CALC:MARK:FUNC:EXEC COMP](#) or [CALC:MARK:FUNC:TRAC ON](#)

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.

#### **Examples**

```
CALC:MARK:COMP:PIN?  
calculate2:marker:compression:pin?
```

**Return Type** Numeric

Default Not applicable

---

## CALCulate<cnum>:MARKer:COMPression:POUT?

(Read-only) Reads the output power at the marker compression level. First send [CALC:MARK:FUNC:EXEC COMP](#) or [CALC:MARK:FUNC:TRAC ON](#)

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.

### Examples

```
CALC:MARK:COMP:POUT?  
calculate2:marker:compression:pout?
```

**Return Type** Numeric

**Default** Not applicable

---

## CALCulate<cnum>:MARKer<mkr>:COUPling[:STATe]<ON|OFF>

(Read-Write) Sets and reads the state of Coupled Markers (ON and OFF).

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <ON|OFF> **OFF (0)** - Turns Coupled Markers OFF
- ON (1)** - Turns Coupled Markers ON

### Examples

```
CALC:MARK:COUP ON  
calculate2:marker8:coupling off
```

**Query Syntax** CALCulate<cnum>:MARKer<mkr>:COUPling:[STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## CALCulate<cnum>:MARKer<mkr>:DELTA <ON|OFF>

**(Read-Write)** Specifies whether marker is relative to the Reference marker or absolute.

**Note:** The reference marker must already be turned ON with [CALC:MARK:REF:STATE](#).

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <ON|OFF> **ON** (or 1) - Specified marker is a Delta marker  
**OFF** (or 0) - Specified marker is an ABSOLUTE marker

#### Examples

```
CALC:MARK:DELT ON  
calculate2:marker8:delta off
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:DELTa?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

### **CALCulate<num>:MARKer<mkr>:DIScrete <ON|OFF>**

**(Read-Write)** Makes the specified marker display either a calculated value between data points (interpolated data) or the actual data points (discrete data).

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <ON|OFF> **ON** (or 1) - Specified marker displays the actual data points  
**OFF** (or 0) - Specified marker displays calculated data between the actual data points.

#### Examples

```
CALC:MARK:DISC ON  
calculate2:marker8:discrete off
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:DIScrete?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## CALCulate<num>:MARKer<mkr>:DISTance <num>

(Read-Write) Set or query marker distance on a time domain trace.

The Write command moves the marker to the specified distance value. Once moved, you can [read the Y axis](#) value or [read the X-axis time](#) value. (Distance is calculated from the X-axis time value.)

The Read command reads the distance of the marker.

If the marker is set as delta, the WRITE and READ data is relative to the reference marker.

[See Critical Note](#)

### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <num> Marker distance in the unit of measure specified with [CALC:TRAN:TIME:MARK:UNIT](#)

### Examples

```
CALC:MARK:DIST .1  
calculate2:marker8:distance 5
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:DISTance?

**Return Type** Numeric

**Default** Not Applicable

---

## CALCulate<num>:MARKer<mkr>:FORMat <char>

(Read-Write) Sets the format of the data that will be returned in a marker data query CALC:MARK:Y? and the displayed value of the marker readout. The selection does not have to be the same as the measurement's display format.

[See Critical Note](#)

### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <char> Choose from:  
**Default** - The format of the selected measurement

**MLINear** - Linear magnitude

**MLOGarithmic** - Logarithmic magnitude

**IMPedance** -  $(R+jX)$

**ADMittance** -  $(G+jB)$

**PHASe** - Phase

**IMAGinary** - Imaginary part (Im)

**REAL** - Real part (Re)

**POLar** - (Re, Im)

**GDELay** - Group Delay

**LINPhase** - Linear Magnitude and Phase

**LOGPhase** - Log Magnitude and Phase

**KELVin** - temperature

**FAHRenheit** - temperature

**CELSius** - - temperature

**Examples**

```
CALC:MARK:FORMat MLIN  
calculate2:marker8:format Character
```

**Query Syntax** CALCulate<cnum>:MARKer<mkr>:FORMat?

**Return Type** Character

**Default** DEFault

---

**CALCulate<cnum>:MARKer<mkr>:FUNCTION:APEak:EXCursion <num>**

**(Read-Write)** Sets amplitude peak excursion for the specified marker. The Excursion value determines what is considered a "peak". This command applies to marker peak searches (Next peak, Peak Right, Peak Left).

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any existing marker number from 1 to 10; if unspecified, value is set to 1.
- <num> Excursion value. Choose any number between **-500** and **500**.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
CALC:MARK:FUNC:APE:EXC 10
calculate2:marker8:function:apeak:excursion maximum
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:FUNCtion:APEak:EXCursion?

**Return Type** Numeric

**Default** 3

---

### **CALCulate<num>:MARKer<mkr>:FUNCtion:APEak:THReshold <num>**

**(Read-Write)** Sets peak threshold for the specified marker. If a peak (using the criteria set with :EXCursion) is below this reference value, it will not be considered when searching for peaks. This command applies to marker peak searches (Next peak, Peak Right, Peak Left).

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <num> Threshold value. Choose any number between **-500** and **500**.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
CALC:MARK:FUNC:APE:THR -40
calculate2:marker8:function:apeak:threshold -55
```

---

<b>Query Syntax</b>	CALCulate<cnum>:MARKer<mkr>:FUNction:APeak:THReshold?
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	-100

---

### **CALCulate<cnum>:MARKer<mkr>:FUNction:DOMain:USER <range>**

**(Read-Write)** Assigns the specified marker to a range number. The x-axis travel of the marker is constrained to the range's span. The span is specified with the [CALC:MARK:FUNC:DOM:USER:START](#) and [STOP](#) commands, unless range 0 is specified which is the full span of the analyzer.

Each channel has **16** user ranges. (Trace statistics use the same ranges.) More than one marker can use a domain range.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <span> User span. Choose any Integer from **0 to 16**
  - 0** is Full Span of the analyzer
  - 1 to 16** are available for user-defined x-axis span

#### **Examples**

```
CALC:MARK:FUNC:DOM:USER 1
calculate2:marker8:function:domain:user 1
```

<b>Query Syntax</b>	CALCulate<cnum>:MARKer<mkr>:FUNction:DOMain:USER?
	Returns the user span number that the specified marker is assigned to.
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	0 - Full Span

---

### **CALCulate<cnum>:MARKer<mkr>:FUNction:DOMain:USER:START <start>**

**(Read-Write)** Sets the start of the span that the specified marker's x-axis span will be constrained to.

Use [CALC:MARK:FUNC:DOM:USER<range>](#) to set range number

Use [CALC:MARK:FUNC:DOM:USER:STOP](#) to set the stop value.

**Note:** If the marker is assigned to range 0 (full span), the USER:START and STOP commands generate an error. You cannot set the START and STOP values for "Full Span".

**Note:** This command does the same as [CALC:FUNC:DOM:USER:STAR](#)

[See Critical Note](#)

#### Parameters

- <cnm> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnm> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <start> The analyzer's **Minimum** x-axis value

#### Examples

```
CALC:MARK:FUNC:DOM:USER:START 500E6  
calculate2:marker8:function:domain:user:start 1e12
```

**Query Syntax** CALCulate<cnm>:MARKer<mkr>:FUNCtion:DOMain:USER:START?

**Return Type** Numeric

**Default** The analyzer's **Minimum** x-axis value

---

### **CALCulate<cnm>:MARKer<mkr>:FUNCtion:DOMain:USER:STOP <stop>**

**(Read-Write)** Sets the stop of the span that the marker's x-axis travel will be constrained to.

Use [CALC:MARK:FUNC:DOM:USER<range>](#) to set range number

Use [CALC:MARK:FUNC:DOM:USER:START](#) to set the stop value.

**Note:** If the marker is assigned to range 0 (full span), the USER:START and STOP commands generate an error. You cannot set the START and STOP values for "Full Span".

**Note:** This command does the same as [CALC:FUNC:DOM:USER:STOP](#)

[See Critical Note](#)

#### Parameters



- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <stop> Stop value of x-axis span; Choose any number between the analyzer's **MINimum** and **MAXimum** x-axis value.

**Examples**

```
CALC:MARK:FUNC:DOM:USER:STOP 500e6  
calculate2:marker8:function:domain1:user:stop 1e12
```

**Query** CALCulate<cnum>:MARKer<mkr>:FUNctio:DOMain:USER:STOP?

**Syntax**

**Return Type** Numeric

**Default** The analyzer's **MAXimum** x-axis value.

---

**CALCulate<cnum>:MARKer<mkr>:FUNctio:EXECute <func>**

(Write-only) Immediately executes (performs) the specified search function.

[Learn more about Marker Search](#)

[See Critical Note](#)

**Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <func> The function to be performed. Choose from:
- **MAXimum** - finds the highest value
  - **MINimum** - finds the lowest value
  - **RPEak** - finds the next valid peak to the right
  - **LPEak** - finds the next valid peak to the left
  - **NPEak** - finds the next highest value among the valid peaks
  - **TARGet** - finds the target value to the right, wraps around to the left
  - **LTARget** - finds the next target value to the left of the marker
  - **RTARget** - finds the next target value to the right of the marker

- **COMP**ression - finds the compression level on a Power Swept S21 trace.

#### Examples

```
CALC:MARK:FUNC:EXEC MAX
calculate2:marker2:function:execute maximum
```

**Query Syntax** Not applicable

**Default** Not applicable

## CALCulate<cnum>:MARKer<mkr>:FUNCtion[:SElect] <char>

(Read-Write) Sets the search function that the specified marker will perform when executed. Use [CALC:MARK:FUNC:TRAC ON](#) to automatically execute the search every sweep.

[Learn more about Marker Search](#)

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <char> Marker function. Choose from:
  - **MAX**imum - finds the highest value
  - **MIN**imum - finds the lowest value
  - **RPE**ak - finds the next valid peak to the right
  - **LPE**ak - finds the next valid peak to the left
  - **NPE**ak - finds the next highest value among the valid peaks
  - **TAR**get - finds the target value to the right, wraps around to the left
  - **LTAR**get - finds the next target value to the left of the marker
  - **RTAR**get - finds the next target value to the right of the marker
  - **COM**pression - finds the compression level on a power-swept S21 trace.

#### Examples

```
CALC:MARK:FUNC MAX
calculate2:marker8:function:select ltarget
```

**Query Syntax** CALCulate<cnum>:MARKer<mkr>:FUNCtion[:SElect]?

**Return Type** Character

**Default** MAX

---

### **CALCulate<cnum>:MARKer<mkr>:TARGet <num>**

**(Read-Write)** Sets the target value for the specified marker when doing Target Searches with [CALC:MARK:FUNC:SEL](#) <TARGet | RTARget | LTARget>

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <num> Target value to search for; Units are NOT allowed.

#### **Examples**

```
CALC:MARK:TARG 2.5  
calculate2:marker8:target -10.3
```

**Query Syntax** CALCulate<cnum>:MARKer<mkr>:TARGet?

**Return Type** Numeric

**Default** 0

---

### **CALCulate<cnum>:MARKer<mkr>:FUNCtion:TRACking <ON | OFF>**

**(Read-Write)** Sets the tracking capability for the specified marker. The tracking function finds the selected search function every sweep. In effect, turning Tracking ON is the same as doing a [CALC:MARK:FUNC:EXECute](#) command every sweep.

[Learn more about Marker Search](#)

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <ON | OFF> **ON** (or 1) - The specified marker will "Track" (find) the selected function every sweep.
- OFF** (or 0) - The specified marker will find the selected function **only** when

the CALC:MARK:FUNC:EXECute command is sent.

**Examples**

```
CALC:MARK:FUNC:TRAC ON  
calculate2:marker8:function:tracking off
```

**Query Syntax**

CALCulate<cnum>:MARKer<mkr>:FUNCTion:TRACking?

**Return Type**

Boolean (1 = ON, 0 = OFF)

**Default**

OFF

---

**CALCulate<cnum>:MARKer:REFerence[:STATe] <ON | OFF>**

(Read-Write) Turns the reference marker (marker 10) ON or OFF. When turned OFF, existing Delta markers revert to absolute markers.

[See Critical Note](#)

**Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<ON | OFF> **ON** (or 1) - turns reference marker ON

**OFF** (or 0) - turns reference marker ON

**Examples**

```
CALC:MARK:REF ON  
calculate2:marker:reference:state OFF
```

**Query Syntax**

CALCulate<cnum>:MARKer:REFerence[:STATe]?

**Return Type**

Boolean (1 = ON, 0 = OFF)

**Default**

OFF

---

**CALCulate<cnum>:MARKer:REFerence:X <num>**

**(Read-Write)** Sets and returns the absolute x-axis value of the reference marker (marker 10).

[See Critical Note](#)

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> X-axis value. Choose any number within the operating domain of the reference marker.

**Examples**

```
CALC:MARK:REF:X 1e9  
calculate2:marker:reference:x 1e6
```

**Query Syntax** CALCulate<num>:MARKer:REFerence:X?

**Return Type** Numeric

**Default** If the first Marker, turns ON in the middle of the X-axis span. If not, turns ON at the position of the active marker.

---

### **CALCulate<num>:MARKer:REFerence:Y?**

**(Read-only)** Returns the absolute Y-axis value of the reference marker.

[See Critical Note](#)

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

**Examples**

```
CALC:MARK:REF:Y?  
calculate2:marker:reference:y?
```

**Return Type** Character

**Default** Not applicable

---

### **CALCulate<num>:MARKer<mkr>:SET <char>**

(Write-only) Sets the selected instrument setting to assume the value of the specified marker.

Marker Functions CENT, SPAN, START, and STOP do not work with channels that are in [CW](#) or [Segment Sweep](#) mode.

[See Critical Note](#)

### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <char> Choose from:
- **CENTer** - changes center frequency to the value of the marker
  - **SPAN** - changes the sweep span to the span that is defined by the delta marker and the marker that it references. Unavailable if there is no delta marker.
  - **START** - changes the start frequency to the value of the marker
  - **STOP** - changes the stop frequency to the value of the marker
  - **RLEVel** - changes the reference level to the value of the marker
  - **DElay** - changes the line length at the receiver input to the phase slope at the active marker stimulus position.
  - **CWFReq** - Sets the CW frequency to the frequency of the active marker. Does NOT change sweep type. NOT available in CW or Power Sweep. Use this argument to first set the CW Frequency to a value that is known to be within the current calibrated range, THEN set [Sweep.Type](#) to POWer or CW.

### Examples

```
CALC:MARK:SET CENT  
calculate2:marker8:set span
```

Query Syntax Not Applicable

[Default](#) Not Applicable

---

**CALCulate<num>:MARKer<mkr>[:STATe] <ON|OFF>**

**(Read-Write)** Turns the specified marker ON or OFF. **Marker 10 is the Reference Marker.** To turn all markers off, use CALC:MARK:AOff.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <ON|OFF> **ON** (or 1) - turns marker ON.  
**OFF** (or 0) - turns marker OFF.

#### Examples

```
CALC:MARK ON  
calculate2:marker8 on
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** Off

---

### CALCulate<num>:MARKer<mkr>:TYPE <char>

**(Read-Write)** Sets the type of the specified marker.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1
- <char> Choose from:

- **NORMal** - a marker that stays on the assigned X-axis position unless moved or searching.
- **FIXed** - a marker that will not leave the assigned X or current Y-axis position.

#### Examples

```
CALC:MARK:TYPE NORM  
calculate2:marker2:type fixed
```

**Query Syntax** CALCulate<num>:MARKer<mkr>:TYPE?

**Return Type** Character

**Default** NORMal

---

## CALCulate<cnum>:MARKer<mkr>:X <num>

**(Read-Write)** Sets the marker's X-axis value (frequency, power, or time). If the marker is set as delta, the SET and QUERY data is relative to the reference marker.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <mkr> Any marker number from 1 to 10; if unspecified, value is set to 1.
- <num> Any X-axis position within the measurement span of the marker.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
CALC:MARK:X 100Mhz  
calculate2:marker8:x maximum
```

**Query Syntax** CALCulate<cnum>:MARKer<mkr>:X?

**Return Type** Numeric

**Default** First Marker turns ON in the middle of the X-axis span. Subsequent markers turn ON at the position of the active marker.

---

## CALCulate<cnum>:MARKer<mkr>:Y?

**(Read-only)** Reads the marker's Y-axis value. The format of the value depends on the current CALC:MARKER:FORMAT setting. If the marker is set as delta, the data is relative to the reference marker. The query always returns two numbers:

- Smith and Polar formats - (Real, Imaginary)
- LINPhase and LOGPhase - (Real, Imaginary)
- All other formats - (Value,0)

**Note:** To accurately read the marker Y-axis value with [trace smoothing](#) applied, the requested format must match the [displayed format](#). Otherwise, the returned value is un-smoothed data. For example, to read the smoothed marker value when measuring group delay, both the display format and the marker format must be set to (Group) Delay.

[See Critical Note](#)



**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <mr> Any marker number from 1 to 10; if unspecified, value is set to 1.

**Examples**

```
CALC:MARK:Y?  
calculate2:marker3:y?
```

**Query Syntax** CALCulate<num>:MARKer<mr>:Y?

**Return Type** Numeric

**Default** Not applicable

---

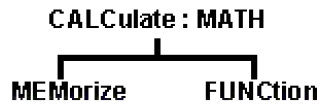
**Last modified:**

9-Mar-2010	Added links to PNOP and PSAT
24-Feb-2009	Replace True/False
12-Feb-2009	Added compression markers (8.50)
2-Oct-2008	Added Calc:Mark:Set CWFR (8.33)
1-Oct-2007	Added temperature formats
27-Mar-2007	Corrected Set?
4-Dec-2006	Added smoothing note to Y?

## Calculate:Math Commands

---

Controls math operations on the currently selected measurement and memory.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Math Operations](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

## CALCulate<cnum>:MATH:FUNCTION <char>

**(Read-Write)** Sets math operations on the currently selected measurement and the trace stored in memory. (There MUST be a trace stored in Memory. See [CALC:MATH MEM](#))

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<char> The math operation to be applied. Choose from the following:

<b>NORMal</b>	Trace data only
<b>ADD</b>	Data + Memory
<b>SUBTract</b>	Data - Memory
<b>MULTiply</b>	Data * Memory
<b>DIVide</b>	Data / Memory

**Examples**

```
CALC:MATH:FUNC NORM  
calculate2:math:function subtract
```

**Query Syntax** CALCulate<cnum>:MATH:FUNCTION?

**Return Type** Character

**Default** NORMal

---

**CALCulate<cnum>:MATH:MEMorize**

(Write-only) Puts the currently selected measurement trace into memory. (Data-> Memory).

[See Critical Note](#)

**Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

**Examples**

```
CALC:MATH:MEM  
calculate2:math:memorize
```

**Query Syntax** Not applicable

**Default** Not applicable

---

## Calculate:Mixer Command

---

### CALCulate<ch>:MIXer:XAXis <char>

**(Read-Write)** Sets or returns the swept parameter to display on the X-axis for the selected FCA measurement. [Learn more about X-axis display.](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1.
- <char> Parameter to display on the X-axis. Choose from:
- INPUT** - Input frequency span
  - OUTPUT** - Output frequency span
  - LO\_1** - First LO frequency span
  - LO\_2** - Second LO frequency span

#### Examples

```
CALC:MIX:XAX INPUT  
calc2:mixer:xaxis output
```

See an example that creates, selects, and calibrates an [SMC](#) and [VMC](#) measurement using SCPI.

**Query Syntax** CALCulate<ch>:MIXer:XAXis?

**Return Type** Character

**Default** OUTPUT

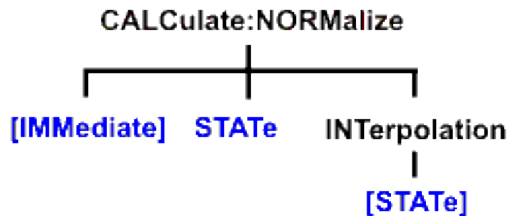
## Calculate:Normalize Commands

---

Specifies the normalization features used for a receiver power calibration.

These commands are [Superseded](#) (Sept 2004).

See the replacement commands in a new [Receiver Power Cal example](#).



Click on a [blue](#) keyword to view the command details.

See Also

- [Example Programs](#)
- [Learn about Receiver Cal](#)
- [SCPI Command Tree](#)

Save and recall your receiver power calibration (which use .CST file commands):

- [SENS:CORR:CSET:SAVE](#)
- [SENS:CORR:CSET\[:SEL\]](#)

Or use these two commands and specify either .STA or .CST file extensions:

- [MMEM:LOAD](#)
- [MMEM:STOR](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

**CALCulate<cnum>:NORMAlize[:IMMediate]** **Superseded**

**Note:** This command is replaced with [SENS:CORR:COLL:METH RPOWer](#) and [SENS:CORR:COLL\[:ACQ\] POWer](#)

See an example of a [Receiver Power Calibration](#).

**(Write only)** Stores the selected measurement's data to that measurement's "divisor" buffer for use by the Normalization data processing algorithm. This command is not compatible with ratioed measurements such as S-parameters. It is intended for receiver power calibration when the selected measurement is of an unratioed power type.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

#### Examples

```
CALC:NORM  
calculate1:norm:immediate
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### CALCulate<num>:NORMalize:STATe <ON | OFF> **Superseded**

**Note:** This command is replaced with [SENS:CORR\[:STATe\] ON/OFF](#)

**(Read-Write)** Specifies whether or not normalization is applied to the measurement. Normalization is enabled only for measurements of unratioed power where it serves as a receiver power calibration.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<ON | OFF> **ON (or 1)** - normalization is applied to the measurement.

**OFF (or 0)** – normalization is NOT applied to the measurement.

#### Examples

```
CALC:NORM:STAT ON  
calculate2:norm:state off
```

**Query Syntax** CALCulate<num>:NORMalize:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

## **CALCulate<cnum>:NORMalize:INTerpolate[:STATe] <ON | OFF>** **Superseded**

**Note:** This command is replaced with [SENS:CORR:INT\[:STATe\] ON|OFF](#)

**(Read-Write)** Turns normalization interpolation ON or OFF. Normalization is enabled only for measurements of unratioed power, where it serves as a receiver power calibration.

[See Critical Note](#)

### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <ON | OFF> **ON (or 1)** – turns interpolation ON.
- OFF (or 0)** – turns interpolation OFF.

### **Examples**

```
CALC:NORM:INT ON  
calculate2:normalize:interpolate:state off
```

**Query Syntax** CALCulate<cnum>:NORMalize:INTerpolate[:STATe]?

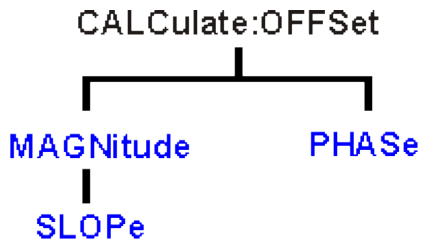
**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

## Calculate:Offset Commands

---

Allows the data trace magnitude and phase to be offset.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Magnitude Offset](#)
- [Learn about Phase Offset](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

**CALCulate<cnum>:OFFSet:MAGNitude <num>**



**(Read-Write)** Offsets the data trace magnitude by the specified value.

To offset the data trace magnitude to a slope value that changes with frequency, use

[CALC:OFFS:MAGN:SLOP](#)

[See Critical Note](#)

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Offset value in dB.

**Examples**

```
CALC:OFFS:MAGN:4  
calculate1:offset:magnitude -2
```

**Query Syntax** CALCulate<num>:OFFSet:MAGNitude?

**Return Type** Numeric

**Default** 0

---

**CALCulate<num>:OFFSet:MAGNitude:SLOPe <num>**

**(Read-Write)** Offsets the data trace magnitude to a value that changes linearly with frequency. The offset slope begins at 0 Hz.

[See Critical Note](#)

**Parameters**

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Offset slope value in dB/ 1GHz.

**Examples**

```
CALC:OFFS:MAGN:SLOP 1 'Offset slope set to 1dB/GHz  
calculate1:offset:magnitude:slope -2 'Offset slope set to -  
2dB/GHz
```

**Query Syntax** CALCulate<num>:OFFSet:MAGNitude:SLOPe?

**Return Type** Numeric

**Default** 0

---

**CALCulate<num>:OFFSet:PHASe <num>[<char>]**

**(Read-Write)** Sets the phase offset for the selected measurement.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Offset phase value. Choose any number between:  
**-360** and **360**
- <char> Units for phase. OPTIONAL. Choose either:  
**DEG** - Degrees (default)  
**RAD** - Radians

#### Examples

```
CALC:OFFS:PHAS 10  
calculate:offset:phase 20rad
```

**Query Syntax** CALCulate:OFFSet:PHASe?

**Return Type** Numeric, returned value always in degrees

**Default** 0 degrees

---

## Calculate:Parameter Commands

---

Lists, creates, selects and deletes measurements.

### **CALCulate:PARameter:**

**CATalog**

**DEFine**

| **EXTended**

**DElete**

| **ALL**

**MNUMber**

| **[SElect]**

**MODify**

| **EXTended**

**SElect**

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Measurement Parameters](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### **CALCulate<cnum>:PARameter:CATalog?**

**(Read-only)** Returns the names and parameters of existing measurements for the specified channel.

**Note:** For Balanced Measurements: CALC:PAR:CAT? may have an unexpected behavior. [Learn more.](#)

[See Critical Note](#)

#### Parameters

<cnum> Channel number of the measurements to be listed. If unspecified, <cnum> is set to 1.

#### Examples

```
CALC:PAR:CAT?  
calculate2:parameter:catalog?
```

**Return Type** String - "<measurement name>,<parameter>,<measurement name>,<parameter>..."

**Default** "CH1\_S11\_1,S11"

---

### CALCulate<cnum>:PARAmeter:DEFine <Mname>,<param>[,port] **Superseded**

**Note:** This command is replaced with [CALC:PAR:DEFine:EXTended](#). This command will continue to work for up to 4 port parameters.

**(Write-only)** Creates a measurement but does NOT display it.

There is no limit to the number of measurements that can be created. However, there is a limit to the number of measurements that can be displayed. See [Traces, Channels, and Windows on the PNA](#).

- Use [DISP:WIND:STATE](#) to create a window if it doesn't already exist.
- Use [DISP:WIND<wnum>:TRAC<tnum>:FEED <Mname>](#) to display the measurement.

For FCA Measurements see [CALC:CUST:DEF](#)

You must select the measurement (CALC<cnum>:PAR:SEL <mname>) before making additional settings.

[See Critical Note](#)

#### Parameters

<cnum> Channel number of the new measurement. If unspecified, value is set to 1.

<Mname> Name of the measurement. Any non-empty, unique string, enclosed in quotes.

<param> **For S-parameters:**

Any S-parameter available in the PNA

**For ratioed measurements:**

Any two receivers that are available in the PNA. (See the [block diagram](#) showing the receivers in YOUR PNA.)

For example: AR1 (this means A/R1)

**For non-ratioed measurements:**

Any receiver that is available in the PNA. (See the [block diagram](#) showing the receivers in YOUR PNA.)

For example: A

**For Balanced Measurements:**

First create an S-parameter measurement, then change the measurement using [CALC:FSIM:BAL](#) commands. [See an example.](#)

**For FCA Measurements** see [CALC:CUST:DEF](#)

[port] Optional argument;

For multi-port reflection S-parameter measurements: specifies the PNA port which will provide the load for the calibration. This argument is ignored if a transmission S-parameter is specified.

For all non S-parameter measurements: specifies the source port for the measurement.

**Examples**

```
CALC4:PAR:DEF 'ch4_S33',S33,2 'Defines an S33 measurement with a load on port2 of the analyzer.'
```

```
calculate2:parameter:define 'ch1_a', a, 1 'unratioed meas
```

```
calculate2:parameter:define 'ch1_a', ar1,1 'ratioed meas
```

**Query Syntax** Not Applicable; see [Calc:Par:Cat?](#)

**Default** Not Applicable

---

**CALCulate<cnum>:PARAmeter:DEFine:EXTended <Mname>,<param>**

**Note:** This command replaces [CALC:PAR:DEF](#) as it allows the creating of measurements using [external multiport testsets](#).

(Write-only) Creates a measurement but does NOT display it.

There is no limit to the number of measurements that can be created. However, there is a limit to the number of measurements that can be displayed. See [Traces, Channels, and Windows on the PNA](#).

- Use [DISP:WIND:STATE](#) to create a window if it doesn't already exist.
- Use [DISP:WIND<wnum>:TRAC<tnum>:FEED <Mname>](#) to display the measurement.

You must select the measurement (CALC<cnum>:PAR:SEL <mname>) before making additional settings.

### [See Critical Note](#)

#### Parameters

- <cnum> Channel number of the new measurement. If unspecified, value is set to 1.
- <Mname> **(String)** Name of the measurement. Any non-empty, unique string, enclosed in quotes.
- <param> **(String )** Measurement Parameter to create. Case sensitive.

#### For S-parameters:

Any S-parameter available in the PNA

Single-digit port numbers CAN be separated by "\_" (underscore). For example: "S21" or "S2\_1"

Double-digit port numbers MUST be separated by underscore. For example: "S10\_1"

#### For ratioed measurements:

Any two PNA physical receivers separated by forward slash '/' followed by comma and source port.

For example: "A/R1, 3"

[Learn more about ratioed measurements](#)

See a [block diagram](#) showing the receivers in YOUR PNA.

#### For non-ratioed measurements:

Any PNA physical receiver followed by comma and source port.

For example: "A, 4"

[Learn more about unratioed measurements.](#)

See the [block diagram](#) showing the receivers in YOUR PNA.

With PNA Rev 6.2, **Ratioed** and **Unratioed** measurements can also use **logical receiver notation** to refer to receivers. This notation makes it easy to refer to receivers with an [external test set](#) connected to the PNA. You do not need to know which physical receiver is used for each test port. [Learn more.](#)

#### For ADC measurements:

Any ADC receiver in the PNA followed by a comma, then the source port.

For example: "AI1,2" indicates the Analog Input1 with source port of 2.

[Learn more about ADC receiver measurements.](#)

#### For Balanced Measurements:

First create an S-parameter measurement, then change the measurement using [CALC:FSIM:BAL](#) "define" commands. [See an example.](#)

#### Examples

```
CALC4:PAR:DEF:EXT 'ch4_S33', 'S33' 'Defines an S33 measurement
```

```
calculate2:parameter:define:extended 'ch1_a', 'b9, 1' 'logical  
receiver notation for unratioed meas of test port 9 receiver with  
source port 1.
```

```
calculate2:parameter:define:extended 'ch1_a', 'b9/a10,1' 'logical  
receiver notation for ratioed meas of test port 9 receiver divided  
by the reference receiver for port 10 using source port 1
```

Query Syntax Not Applicable; see [Calc:Par:Cat?](#)

Default Not Applicable

---

**CALCulate<cnum>:PARAmeter:DELeTe[:NAME] <Mname>**

(Write-only) Deletes the specified measurement.

[See Critical Note](#)

**Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<Mname> String - Name of the measurement

**Examples**

```
CALC:PAR:DEL 'TEST'  
calculate2:parameter:delete 'test'
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**CALCulate:PARAmeter:DElete:ALL**

(Write-only) Deletes all measurements on the PNA.

[See Critical Note](#)

**Parameters**

**Examples**

```
CALC:PAR:DEL:ALL
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**CALCulate<cnum>:PARAmeter:MNUMber?**

(Read-only) Returns the measurement number of the selected measurement on the specified channel. This is useful when needing to identify a measurement by number, such as with [Status:Ques:Lim](#) or [Status:Oper:Aver](#) commands.

[See Critical Note](#)

**Parameters**

<cnum> Channel number of the measurement. If unspecified, <cnum> is set to 1.

**Examples**

```
CALC:PAR:MNUM?  
calculate2:parameter:mnumber?
```

**Return Type** Numeric

**Default** Not Applicable

---



## CALCulate<cnum>:PARAmeter:MNUM[:SElect] <n>

(Read-Write) Sets the selected measurement for the channel. Most CALC: commands require that this, or [CALC:PAR:SEL](#), be sent before a setting change is made to that measurement. Each channel can have one selected measurement.

These are the same numbers you see in the “Tr1”, “Tr2” annotation next to the parameter name on the PNA screen.

### Parameters

<cnum> Channel number of the measurement to be selected. If unspecified, <cnum> is set to 1.

<n> Numeric - Measurement number.

### Examples

```
CALC:PAR:MNUM 2  
calculate2:parameter:mnumber:select 3
```

**Query Syntax** CALCulate:PARAmeter:MNUMber[:SElect]?

**Return Type** String

**Default** The preset measurement number is 1

---

## CALCulate<cnum>:PARAmeter:MODify <param> **Superseded**

**Note:** This command is replaced with [CALC:PAR:MOD:EXT](#). This command will continue to work for up to 4 port parameters.

(Write-only) Modifies a standard measurement using the same arguments as [CALC:PAR:DEF](#). To modify an FCA measurement, use [CALC:CUST:MOD](#).

### [See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. The selected measurement on that channel will be changed. If unspecified, <cnum> is set to 1.

<param> Measurement parameter to change to. Use the same <param> arguments as [CALC:PAR:DEF](#).

### Examples

```
SYST:PRESET  
CALC:PAR:DEF "MyMeas", S11  
CALC:PAR:SEL "MyMeas"  
CALC:PAR:MOD AR1 'changes the selected S11 measurement to an A/R1
```

measurement

Query Syntax Not Applicable

Default Not Applicable

---

### CALCulate<cnum>:PARameter:MODify:EXTended <param>

**Note:** This command replaces [CALC:PAR:MOD](#) as it allows modification of measurements using [external multiport testsets](#).

**(Write-only)** Modifies a standard measurement using the same arguments as [CALC:PAR:DEF:EXT](#). To modify an FCA measurement, use [CALC:CUST:MOD](#).

[See Critical Note](#)

#### Parameters

- <cnum> Channel number of the measurement. The selected measurement on that channel will be changed. If unspecified, <cnum> is set to 1.
- <param> **(String)** New measurement parameter. Use the same <param> arguments as [CALC:PAR:DEF:EXT](#).

#### Examples

```
SYST:PRESET  
  
CALC:PAR:DEF:EXT "MyMeas", "S10_1"  
  
CALC:PAR:SEL "MyMeas"  
  
CALC:PAR:MOD:EXT "a4b4,1" 'changes the selected S10_1  
measurement to an a4/b4 measurement with source port 1
```

Query Syntax Not Applicable

Default Not Applicable

---

### CALCulate<cnum>:PARameter:SElect <Mname>

**(Read-Write)** Sets the selected measurement. Most CALC: commands require that this command be sent before a setting change is made. One measurement on each channel can be selected at the same time. To obtain a list of currently named measurements, use CALC:PAR:CAT?

**Parameters**

<cnum> Channel number of the measurement to be selected. If unspecified, <cnum> is set to 1.

<Mname> String - Name of the measurement. (Do NOT include the parameter name.)

**Examples**

```
CALC:PAR:SEL 'TEST'  
calculate2:parameter:select 'test'
```

**Query Syntax** CALCulate:PARameter:SElect?

**Return Type** String

**Default** The preset measurement name is "CH1\_S11\_1"

---

Last modified:

31-Oct-2008	Added Mnum select (8.33)
19-Apr-2007	Added ADC meas
9/12/06	New Extended commands.

## Calculate:RData? Command

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

### CALCulate<cnum>:RDATA? <char>

(Read-only) Returns receiver data for the selected measurement. To query measurement data, see [CALC:DATA?](#)

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <char> Choose from any physical receiver in the PNA.

For example: "A"

Also, **REF** - returns data for either R1 or R2 data depending on the source port of the selected measurement.

See the [block diagram](#) showing the receivers in YOUR PNA.

**Note:** Logical receiver notation is NOT allowed with this command. [Learn more.](#)

#### Example

```
GPIOB.Write "INITiate:CONTinuous OFF"  
GPIOB.Write "INITiate:IMMediate;*wai"  
GPIOB.Write "CALCulate:RDATA? A"
```

```
GPIOB.Write "CALCulate:RDATA? REF"
```

**Return Type** Depends on [FORM:DATA](#) - Two numbers per data point

**Default** Not Applicable

#### Notes:

Generally when you query the analyzer for data, you expect that the number of data values returned will be consistent with the number of points in the sweep.

However, if you query **receiver** data while the instrument is sweeping, the returned values may contain zeros. For example, if your request for receiver data is handled on the 45th point of a 201 point sweep, the first 45 values will be valid data, and the remainder will contain complex zero.

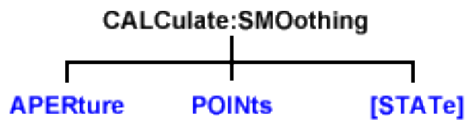
This can be avoided by synchronizing this request with the end of a sweep or putting the channel in hold mode.

[Learn about Unratioed Measurements](#)

## Calculate:Smoothing Commands

---

Controls point-to-point smoothing. Smoothing is a noise reduction technique that averages adjacent data points in a measurement trace. Choose the amount of smoothing by specifying either the number of points or the aperture. Smoothing is not the same as CALC:AVERage which averages each data point over a number of sweeps.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Smoothing](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### CALCulate<cnum>:SMOothing:APERture <num>

**(Read-Write)** Sets the amount of smoothing as a percentage of the number of data points in the channel.

[See Critical Note](#)

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Percentage value. Choose any number between:  
**1 and 25**

#### Examples

```
CALC:SMO:APER 2
calculate2:smoothing:aperture 20.7
```

**Query Syntax** CALCulate<cnum>:SMOothing:APERture?

**Return Type** Numeric

### **CALCulate<cnum>:SMOothing:POINts <num>**

(Read-Write) Sets the number of adjacent data points to average.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Number of points from 1 point to maximum of 25% of data points in the channel. For example: if number of points in a data trace = 401, the maximum value for points = 100. The points value is always rounded to the closest odd number.

#### **Examples**

```
CALC:SMO:POIN 50  
calculate2:smoothing:points 21
```

**Query Syntax** CALCulate<cnum>:SMOothing:POINts?

**Return Type** Numeric

**Default** 3

---

### **CALCulate<cnum>:SMOothing[:STATe] <ON | OFF>**

(Read-Write) Turns data smoothing ON or OFF.

[See Critical Note](#)

#### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <ON | OFF> **ON** (or 1) - turns smoothing ON.  
**OFF** (or 0) - turns smoothing OFF.

#### **Examples**

```
CALC:SMO ON  
calculate2:smoothing:state off
```

**Query Syntax** CALCulate<cnum>:SMOothing[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

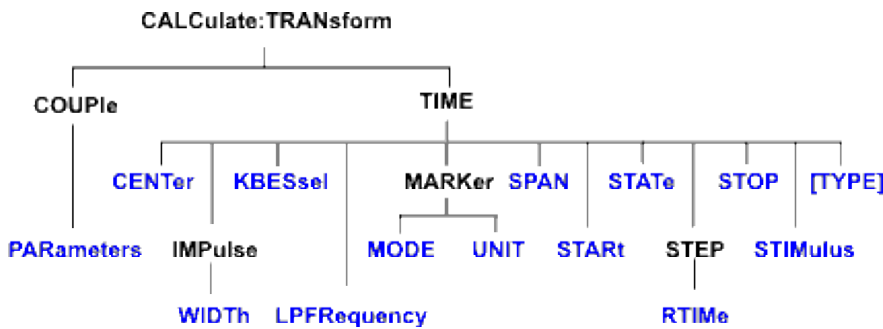




## Calculate:Transform Commands

---

Specifies the settings for time domain transform.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Time Domain](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Critical Note:** CALCulate commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### CALCulate<cnum>:TRANSform:COUPle:PARAmeters <num>

**(Read-Write)** Specifies the time domain transform parameters to be coupled. The settings for those parameters will be copied from the selected measurement to all other measurements on the channel.

- To turn coupling ON and OFF, use [SENS:COUP:PAR](#)
- To specify Gating parameters to couple, use [CALC:FILT:COUP:PAR](#)

Learn more about [Time Domain Trace Coupling](#)

### [See Critical Note](#)

#### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> (Numeric) Parameters to couple. To specify more than one parameter, add the

numbers.

**1** - Transform Stimulus (Start, Stop, Center, and Span TIME settings.)

**2** - Transform State (ON / OFF)

**4** - Transform Window (Kaiser Beta / Impulse Width)

**8** - Transform Mode (Low Pass Impulse, Low Pass Step, Band Pass)

**16** - Transform Distance Marker Units

**Examples**

'To couple all parameters:

```
CALC:TRAN:COUP:PAR 31
```

'To couple Stimulus and Mode:

```
calculate2:transform:couple:parameters 9
```

**Query Syntax** CALCulate<cnum>:TRANSform:COUPle:PARameters?

**Return Type** Numeric

**Default** 29 (All parameters except **2** - Transform State)

---

**CALCulate<cnum>:TRANSform:TIME:CENTer <num>**

(Read-Write) Sets the center time for time domain measurements.

[See Critical Note](#)

**Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<num> Center time in seconds; any number between:  
 $\pm$  (number of points-1) / frequency span

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
CALC:TRAN:TIME:CENT 1e-8
```

```
calculate2:transform:time:center 15 ps
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:CENTer?

**Return Type** Numeric

**Default** 0

---

## CALCulate<cnum>:TRANSform:TIME:IMPulse:WIDTh <num>

(Read-Write) Sets the impulse width for the transform window.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Impulse width in seconds; Choose any number between:  
**.6 / frequency span** and **1.39 / frequency span**

### Examples

```
CALC:TRAN:TIME:IMP:WIDTh 10  
calculate2:transform:time:impulse:width 13
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:IMPulse:WIDTh?

**Return Type** Numeric

**Default** .98 / Default Span

---

## CALCulate<cnum>:TRANSform:TIME:KBESsel <num>

(Read-Write) Sets the parametric window for the Kaiser Bessel window.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Window width for Kaiser Bessel in seconds; Choose any number between:  
**0.0** and **13.0**

### Examples

```
CALC:TRAN:TIME:KBES 10  
calculate2:transform:time:kbessel 13
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:KBESsel?

**Return Type** Numeric

**Default** 6

---

## CALCulate<cnum>:TRANSform:TIME:LPFREQuency

(Write-only) Sets the start frequencies in LowPass Mode.

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

#### Examples

```
CALC:TRAN:TIME:LPFR  
calculate2:transform:time:lpfrequency
```

Query Syntax Not applicable

Default Not applicable

---

### CALCulate<num>:TRANSform:TIME:MARKer:MODE <char>

(Read-Write) Specifies the measurement type in order to determine the correct marker distance.

- Select Auto for S-Parameter measurements.
- Select Reflection or Transmission for arbitrary ratio or unratiod measurements.

This setting affects the display of ALL markers for only the ACTIVE measurement.

Learn more about [Distance Markers](#).

[See Critical Note](#)

#### Parameters

<num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.

<char> Choose from:

**AUTO** If the active measurement is an S-Parameter, automatically chooses reflection or transmission. If non S-Parameter measurements, reflection is chosen.

**REFlection** Displays the distance from the source to the receiver divided by two (to compensate for the return trip.)

**TRANsmision** Displays the distance from the source to the receiver.

#### Examples

```
CALC:TRAN:TIME:MARK:MODE REFL  
calculate2:transform:time:marker:mode auto
```

Query Syntax CALCulate<num>:TRANSform:TIME:MARKer:MODE?

**Return Type** Character

**Default** Auto

---

### **CALCulate<cnum>:TRANSform:TIME:MARKer:UNIT <char>**

**(Read-Write)** Specifies the unit of measure for the display of marker distance values. This settings affects the display of ALL markers for only the ACTIVE measurement (unless Distance Maker Units are coupled using [CALC:TRAN:COUP:PAR](#)).

Learn more about [Distance Markers](#).

[See Critical Note](#)

#### **Parameters**

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<char> Choose from:

**METR**s

**FEET**

**INCH**es

#### **Examples**

```
CALC:TRAN:TIME:MARK:UNIT INCH  
calculate2:transform:time:marker:unit feet
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:MARKer:UNIT?

**Return Type** Character

**Default** METRs

---

### **CALCulate<cnum>:TRANSform:TIME:SPAN <num>**

**(Read-Write)** Sets the span time for time domain measurements.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Span time in seconds; any number between:  
0 and  $2 * [(number\ of\ points - 1) / frequency\ span]$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
CALC:TRAN:TIME:SPAN 1e-8  
calculate2:transform:time:span maximum
```

**Query Syntax** CALCulate<num>:TRANSform:TIME:SPAN?

**Return Type** Numeric

**Default** 20 ns

---

### CALCulate<num>:TRANSform:TIME:STARt <num>

**(Read-Write)** Sets the start time for time domain measurements.

[See Critical Note](#)

#### Parameters

- <num> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <num> is set to 1.
- <num> Start time in seconds; any number between:  
 $\pm (number\ of\ points - 1) / frequency\ span$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
CALC:TRAN:TIME:STAR 1e-8  
calculate2:transform:time:start minimum
```

**Query Syntax** CALCulate<num>:TRANSform:TIME:STARt?

**Return Type** Numeric

**Default** -10 ns

## CALCulate<cnum>:TRANSform:TIME:STATe <ON | OFF>

(Read-Write) Turns the time domain transform capability ON or OFF.

[See Critical Note](#)

**Note:** [Sweep type](#) must be set to Linear Frequency in order to use Time Domain Transform.

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<ON|OFF> **ON** (or 1) - turns time domain ON.  
**OFF** (or 0) - turns time domain OFF.

### Examples

```
CALC:TRAN:TIME:STAT ON  
calculate2:transform:time:state off
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## CALCulate<cnum>:TRANSform:TIME:STOP <num>

(Read-Write) Sets the stop time for time domain measurements.

[See Critical Note](#)

### Parameters

<cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.

<num> Stop time in seconds; any number between:  
 $\pm (\text{number of points}-1) / \text{frequency span}$

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
CALC:TRAN:TIME:STOP 1e-8  
calculate2:transform:time:stop maximum
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:STOP?

**Return Type** Numeric

**Default** 10 ns

## CALCulate<cnum>:TRANSform:TIME:STEP:RTIME <num>

(Read-Write) Sets the step rise time for the transform window.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <num> Rise time in seconds; Choose any number between:  
**.45 / frequency span** and **1.48 / frequency span**

### Examples

```
CALC:TRAN:TIME:STEP:RTIM 1e-8  
calculate2:transform:time:step:rtime 15 ps
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:STEP:RTIME?

**Return Type** Numeric

**Default** .99 / Default Span

---

## CALCulate<cnum>:TRANSform:TIME:STIMulus <char>

(Read-Write) Sets the type of simulated stimulus that will be incident on the DUT.

[See Critical Note](#)

### Parameters

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <char> Choose from:  
**STEP** - simulates a step DUT stimulus  
**IMPulse** - simulates a pulse DUT stimulus

STEP can ONLY be used when [CALC:TRAN:TIME:TYPE](#) is set to LPASs (Lowpass). (STEP **cannot** be used with TYPE = BPASs.)

**:STIM STEP** will set :TYPE to **LPASs**

**:TYPE BPASs** will set :STIM to **IMPulse**

### Examples

```
CALC:TRAN:TIME:STIM STEP  
calculate2:transform:time:stimulus impulse
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME:STIMulus?

**Return Type** Character



## **CALCulate<cnum>:TRANSform:TIME[:TYPE] <char>**

(Read-Write) Sets the type of time domain measurement.

[See Critical Note](#)

### **Parameters**

- <cnum> Channel number of the measurement. There must be a selected measurement on that channel. If unspecified, <cnum> is set to 1.
- <char> Type of measurement. Choose from:  
**LPASs** - Lowpass; Must also send [CALC:TRAN:TIME:LPFRequency](#) before calibrating.  
**BPASs** - Bandpass;

BPASs can **only** be used when [CALC:TRAN:TIME:STIM](#) is set to IMPulse.  
(BPASs **cannot** be used with :STIM = STEP)

:STIM **STEP** will set :TYPE to **LPASs**

:TYPE **BPASs** will set :STIM to **IMPulse**

### **Examples**

```
CALC:TRAN:TIME LPAS  
calculate2:transform:time:type bpas
```

**Query Syntax** CALCulate<cnum>:TRANSform:TIME[:TYPE]?

**Return Type** Character

Default BPAS

## Control Commands

---

Specifies the settings to remotely control the rear panel connectors and ECAL Module state.

### **CONTRol**

#### **AUXiliary - More Commands**

#### **CHANnel:INTerface:CONTRol:**

| **CONFig:RECall**

| **[STATe]**

#### **ECAL:MODule:**

| **PATH:**

| **COUNt?**

| **STATe**

| **STATe**

#### **EXTernal:TESTset - More Commands**

#### **HANDler - More Commands**

#### **NOISe:SOURce[:STATe]**

#### **SIGNal:**

| **TRIGger**

| **ATBA**

| **OUTP**

Click on a **blue** keyword to view the command details.

**Red** command is superseded.

#### **See Also**

- [Example Programs](#)

- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)
- See a pinout and detailed description of the rear panel connectors:
  - [Auxilliary IO connector](#)
  - [External Test Set IO connector](#)
  - [Material Handler IO connector](#)

---

## CONTrol:CHANnel:INTerface:CONTrol:CONFig:RECall <string>

(Write-only) Recalls an Interface Control configuration file. [Learn more about Interface Control.](#)

### Parameters

<string> File name and extension (.xml) of the configuration file to recall. Files are typically stored in the default folder "C:/Program Files/Agilent/Network Analyzer/Documents". To recall from a different folder, specify the full path name.

### Examples

```
CONT:CHAN:INT:CONT:CONF:REC 'MyConfigFile.xml'
```

```
control:channel:interface:control:config:recall 'C:/Program
Files/Agilent/Network Analyzer/Documents/MyFile.xml'
```

Query Syntax Not Applicable

Default Not Applicable

---

## CONTrol:CHANnel:INTerface:CONTrol[:STATe] <bool>

**(Read-Write)** Enables and disables ALL Interface Control settings. To send data, the individual interfaces must also be enabled. [Learn more about Interface Control.](#)

#### Parameters

<bool> Boolean

**OFF (0)** - Interface Control is disabled;NO control data is sent.

**ON (1)** - Interface Control is enabled.

#### Examples

```
CONT:CHAN:INT:CONT 1
```

```
control:channel:interface:control:state 0
```

**Query Syntax** CONTrol:CHANnel:INTerface:CONTrol[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

### CONTrol:ECAL:MODule[num]:PATH:COUNT? <name>

**(Read-only)** Returns the number of unique states that exist for the specified path name on the selected ECal module.

This command performs exactly the same function as [SENS:CORR:CKIT:ECAL:PATH:COUNT?](#)

Use the [CONT:ECAL:MOD:PATH:STAT](#) command to set the module into one of those states.

Use [SENS:CORR:CKIT:ECAL:PATH:DATA?](#) to read the data for a state.

#### Parameters

[num] Optional argument. USB number of the ECal module. If unspecified (only one ECal module is connected to the USB), <num> is set to 1. If two or more modules are connected, use [SENS:CORR:CKIT:ECAL:LIST?](#) to determine how many, and [SENS:CORR:CKIT:ECAL:INF?](#) to verify their identities.

<name> Name of the path for which to read number of states. Choose from:

Reflection paths

- **A**
- **B**
- **C** (4-port modules)
- **D** (4-port modules)

## Transmission paths

- **AB**
- **AC** (4-port modules)
- **AD** (4-port modules)
- **BC** (4-port modules)
- **BD** (4-port modules)
- **CD** (4-port modules)

Note: For each transmission path, the first of the available states is the through state, the second is the confidence (attenuator) state.

### Examples

```
CONT:ECAL:MOD:PATH:COUNT? A
control:ecal:module2:path:count? cd
```

**Return Type** Integer

**Default** Not Applicable

---

## CONTrol:ECAL:MODule[num]:PATH:STATe <path>, <stateNum>

(Write-only) Sets the internal state of the selected ECAL module. This command supersedes [CONT:ECAL:MOD:STAT](#).

- Use [CONT:ECAL:MOD:PATH:COUN?](#) to read the number of unique states that exist for the specified path name on the module.
- Use [SENS:CORR:CKIT:ECAL:PATH:DATA?](#) to read the data for a state (from the module memory) corresponding to the stimulus values of a channel.

### Parameters

- [num] Optional argument. USB number of the ECAL module. If unspecified (only one ECAL module is connected to the USB), <num> is set to 1. If two or more modules are connected, use [SENS:CORR:CKIT:ECAL:LIST?](#) to determine how many, and [SENS:CORR:CKIT:ECAL:INF?](#) to verify their identities.
- <path> Path name for which to set a state.

**Note:** The impedance paths are not independent. For example, changing the impedance presented on path A will cause a change to the impedance on path B.

Choose from:

Reflection paths

- **A**
- **B**
- **C** (4-port modules)
- **D** (4-port modules)

Transmission paths

- **AB**
- **AC** (4-port modules)
- **AD** (4-port modules)
- **BC** (4-port modules)
- **BD** (4-port modules)
- **CD** (4-port modules)

<stateNum> Number of the state to set. Refer to the following table to associate the <stateNum> with a state in your ECal module.

In addition, [CONT:ECAL:MOD:PATH:COUNT?](#) returns the number of states in the specified ECal module.

<stateNum>	N4432A and N4433A States	N4431A States	N469x States**	8509x States
<b>One-Port Reflection States</b>				
1	Open	Open	Impedance 1	Open
2	Short	Short	Impedance 2	Short
3	Impedance 1	Impedance 1	Impedance 3	Impedance 1
4	Impedance 2	Impedance 2	Impedance 4	Impedance 2

5			Impedance 5	
6			Impedance 6	
7			Impedance 7	
<b>Two-Port Transmission States</b>				
1	Thru	Thru	Thru	Thru
2	Confidence	Confidence	Confidence	Confidence

\*\* The following modules have only FOUR Impedance states (1, 2, 3, 4):  
N4690B ,N4691B ,N4692A ,N4696B.

#### Examples

```
CONT:ECAL:MOD:PATH:STATe A,5
control:ecal:module2:state BC,1
```

**Query Syntax** Not Applicable

**Default** Not Applicable

### CONTrol:ECAL:MODULE[num]:STATe <value> **Superseded**

This command is replaced with [CONT:ECAL:MOD:PATH:STATe](#).

**(Write-only)** Sets the internal state of the selected ECAL module.

#### Parameters

[num] Optional argument. USB number of the ECal module. If unspecified (only one ECal module is connected to the USB), <num> is set to 1. If two or more modules are connected, use [SENS:CORR:COLL:CKIT:INF?](#) to verify their identity.

<value> Integer code for switching the module. The following are codes for Agilent ECal modules.

<b>Agilent 8509x Modules</b>		
State	Port A	Port B
Open	0	0
Short	43	43
Load	33	33
Mismatch	4	16
Thru	84	
Confidence	20	

Agilent N469x Modules		
State	Port A	Port B
Open	36	33
Short	39	45
Load	37	37
Mismatch (Offset short)	53	53
Impedance 5 (Offset open)	5	5
Impedance 6 (Offset short)	21	21
Impedance 7 (Offset short)	38	41
Thru	42	
Confidence	40	

Agilent N4431A Modules				
State	Port A	Port B	Port C	Port D
Open	-1398	-1384	-2774	-2654
Short	-1350	-1381	-2582	-2642
Load	26985	-26986	-26986	26985
Mismatch	-26986	26985	26985	-26986
Path	Thru		Confidence	
AB Path	-2590		-598	
AC Path	-4011		85	
AD Path	-2517		16042	
BC Path	-1650		-598	
BD Path	-4011		85	
CD Path	-1352		16042	



Agilent N4432A and N4433A Modules				
State	Port A	Port B	Port C	Port D
Open	-6971	-11835	-14895	-14876
Short	-14395	-12859	-14899	-14905
Load	-14907	-14907	-14907	-14907
Offset Short	-9787	-6459	-14874	-14887
Path		Thru	Confidence	
AB Path		13765	30069	
AC Path		-10519	-2327	
AD Path		-10538	-2346	
BC Path		-5655	-1559	
BD Path		-5674	-1578	
CD Path		-15051	30069	

#### Examples

```
CONT:ECAL:MOD:STAT 36
control:ecal:module2:state 38
```

Query Syntax Not Applicable

Default Not Applicable

### CONTrol:NOISe:SOURce[:STATe] <bool>

(Read-Write) Set and read the noise source ON | OFF state. For [Noise Figure measurements](#).

#### Parameters

<bool> Boolean

**OFF (0)** - Noise Source OFF

**ON (1)** - Noise Source ON

#### Examples

```
CONT:NOIS:SOUR 1
control:noise:source:state 0
```

Query Syntax CONTrol:NOISe:SOURce[:STATe]?

Return Type Boolean

Default OFF (0)

### CONTrol:SIGNal <conn>,<char>

**(Read-Write)** Configures external triggering in the PNA.

- To control BNC1 and BNC2 with this command, then you **MUST** have [TRIG:PREF:AIGLobal = ON](#). [Learn more](#)
- [Trigger:Sequence:Source](#) is automatically set to External when **CONTROL:SIGNal** is sent.
- Edge triggering is only available on some Microwave PNA models.
- For more information, see [External Triggering](#) in the PNA.

### Parameters

<conn> Rear Panel connector to send or receive trigger signals. Choose from:

**BNC1** Trigger IN from rear-panel [Trigger IN BNC connector](#)

**AUXT** Trigger IN from [AUX IO connector Pin 19](#)

**Note:** Only one of the input connectors is active at a time. When a command is sent to one, the PNA automatically makes the other INACTIVE.

**BNC2** Trigger OUT to rear-panel [Trigger OUT BNC connector](#).

**MATHtrigger** - Trigger IN from rear-panel [Material Handler connector Pin 18](#)

**RDY** - Ready for trigger OUT. (Not available on E836x models)

- PNA-X: [Meas Trig RDY](#)
- PNA-L: [AUX I/O p18](#), [Handler I/O p21](#) ([Some models](#))

<char> **INACTIVE** - Disables the specified connector <conn>.

**Choose from ONLY the following** when <conn> is set to **BNC1** or **AUXT** or **MATHtrigger**:

- **TIENEGATIVE** - (Trigger In Edge Negative) - Triggers the PNA when receiving a negative going signal
- **TIEPOSITIVE** - (Trigger In Edge Positive) - Triggers the PNA when receiving a positive going signal
- **TILLOW** - (Trigger In Level Low) - Triggers the PNA when receiving a low level signal
- **TILHIGH** - (Trigger In Level High) - Triggers the PNA when receiving a High-level signal

Choose from **ONLY the following** when <conn> is set to **BNC2**:

Use [CONTrol:SIGNal:TRIGger:OUTP](#) to enable the BNC2 output.

The following selections send a positive or negative pulse before or after each trigger acquisition. This normally occurs each sweep unless a channel is in [point trigger](#) mode.

- **TOPPAFTER** - (Trigger Out Pulse Positive After) - Sends a POSITIVE going TTL pulse at the END of each trigger acquisition.
- **TOPPBEFORE** - (Trigger Out Pulse Positive Before) - Sends a POSITIVE going TTL pulse at the START of each trigger acquisition.
- **TOPNAFTER** - (Trigger Out Pulse Negative After) - Sends a NEGATIVE going TTL pulse at the END of each trigger acquisition.
- **TOPNBEFORE** - (Trigger Out Pulse Negative Before) - Sends a NEGATIVE going TTL pulse at the START of each trigger acquisition.

Choose from **ONLY the following** when <conn> is set to **RDY**:

- **LOW** Outputs a TTL low when the PNA is ready for trigger. (Default setting)
- **HIGH** Outputs a TTL high when the PNA is ready for trigger.

#### Examples

```
CONT:SIGN BNC1,TIENEGATIVE
control:signal bnc2,toppbefore

CONT:SIGN RDY,LOW
```

**Query Syntax** CONTrol:SIGNal? <conn>

In addition to the arguments listed above, the following is also a possible returned value:

**NAVAILABLE** - This feature is not available on this PNA

**Return Type** Character

#### Default

At Preset:

BNC1 = INACTIVE  
BNC2 = INACTIVE  
AUXT = TILHIGH

When [Output is enabled](#):

BNC1 = INACTIVE

BNC2 = TOPPAFTER  
AUXT = TILHIGH

---

### CONTrol:SIGNal:TRIGger:ATBA <bool>

**(Read-Write) Accept Trigger Before Armed** Determines what happens to an EDGE trigger signal if it occurs before the PNA is ready to be triggered. (LEVEL trigger signals are always ignored.) For more information, see [External triggering](#).

#### Parameters

<bool> Boolean

OFF (0) - A trigger signal is ignored if it occurs before the PNA is ready to be triggered.

ON (1) - A trigger signal is remembered and then used when the PNA becomes armed (ready to be triggered). The PNA remembers only one trigger signal.

#### Examples

```
CONT:SIGN:TRIG:ATBA 0  
control:signal:trigger:atba ON
```

**Query Syntax** CONTrol:SIGNal:TRIGger:ATBA?

**Return Type** Boolean

**Default** OFF

---

### CONTrol:SIGNal:TRIGger:OUTP <bool>

**(Read-Write) Output Enabled** The PNA can be enabled to send trigger signals out the rear-panel [TRIGGER OUT BNC](#) connector. Use [CONTrol:SIGNal](#) to configure for output triggers.

For more information, see [External triggering](#).

#### Parameters

<bool> Boolean

OFF (0) - PNA does NOT output trigger signals.

ON (1) - PNA DOES output trigger signals.

#### Examples

```
CONT:SIGN:TRIG:OUTP 1  
control:signal:trigger:outp OFF
```

**Query Syntax** CONTrol:SIGNal:TRIGger:OUTP?

**Return Type** Boolean

Default   OFF

---

Last Modified:

24-Feb-2009	Replaced True/False
14-Mar-2008	Added RDY argument to CONT:SIGN
25-Feb-2008	Clarified CONT:SIGN
30-Jan-2008	Added ECal states note
22-Aug-2007	Added noise command (8.0)
18-Jan-2007	Fixed count? example

## Display Commands

---

Controls the settings of the front panel screen.

### DISPlay:

#### ANNotation

|FREQuency[:STATE]

|MESSage:STATe

|STATus

#### ARRange

#### CATalog?

#### COLor [More Commands](#)

#### ENABle

#### FSIGn

#### TMAX

#### TILE

#### WINDow

|ANNotation

|TRACe:STATe

|MARKer

|SINGle[:STATe]

|SIZE

|STATe

|CATalog?

|ENABle

|SIZE

|[STATe]

|TABLe

|TITLe

|DATA

||[STATe]

|TRACe

|DELeTe

|FEED

|GRATicule:GRID:LTYPE

|MEMory[:STATe]

|SElect

|[STATe]

|TITLe

|DATA

||[:STATe]

|Y[:SCALe]

|AUTO

<b>COUPle</b>
<b>METHod</b>
<b>[STATe]</b>
<b>PDIVision</b>
<b>RLEVel</b>
<b>RPOSition</b>
<b>VISible</b>

Click on a [blue](#) keyword to view the command details.

#### See Also

- See an [example](#) using some of these commands
- [Synchronizing the PNA and Controller](#)
- [Learn about Screen Setup](#)
- [SCPI Command Tree](#)

---

## DISPlay:ANNotation:FREQuency[:STATe] <ON | OFF>

**(Read-Write)** Turns frequency information on the display title bar ON or OFF for all windows.

#### Parameters

<ON | OFF>    **ON** (or 1) - turns frequency annotation ON.  
                  **OFF** (or 0) - turns frequency annotation OFF.

#### Examples

```
DISP:ANN:FREQ ON  
display:annotation:frequency:state off
```

**Query Syntax**    DISPlay:ANNotation:FREQuency[:STATe]?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    ON (1)

---

## DISPlay:ANNotation:MESSage:STATe <ON | OFF>

**(Read-Write)** Enables and disables error pop-up messages on the display.

**Parameters**

<ON | OFF>    **ON** (or 1) - enables error pop-up messages  
                  **OFF** (or 0) - disables error pop-up messages

**Examples**

```
DISP:ANN:MESS:STAT ON  
display:annotation:message:state off
```

**Query Syntax**    DISPlay:ANNotation:MESSage:STATe?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    ON (1)

---

## DISPlay:ANNotation:STATus <ON | OFF>

**(Read-Write)** Turns the status bar at the bottom of the screen ON or OFF. The status bar displays information for the active window.

**Parameters**

<ON | OFF>    **ON** (or 1) - turns status bar ON.  
                  **OFF** (or 0) - turns status bar OFF.

**Examples**

```
DISP:ANN:STAT ON  
display:annotation:status off
```

**Query Syntax**    DISPlay:ANNotation:STATus?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    Last state that was set

---

## DISPlay:ARRange <char>



**(Write-only)** Places EXISTING measurements into pre-configured window arrangements. Overlay, Stack(2), Split(3), and Quad(4) creates new windows. To learn more, see [Arrange Existing Measurements](#).

**Parameters**

<char> Window arrangement. Choose from:

- TILE - tiles existing windows
- CASCade - overlaps existing windows
- OVERlay - all traces placed in 1 window
- STACK - 2 windows
- SPLit - 3 windows
- QUAD - 4 windows

**Examples**

```
DISP:ARR CASC  
display:arrange cascade
```

**Query Syntax** Not Applicable

**Default** TILE

---

**DISPlay:CATalog?**

**(Read-only)** Returns the existing Window numbers.

**Return Type** String of Character values, separated by commas

**Example**

```
Two windows with numbers 1 and 2 returns:  
"1,2"
```

**Default** Not applicable

---

**DISPlay:ENABLE <ON | OFF>**

**(Read-Write)** Specifies whether to disable or enable all analyzer display information **in all windows** in the analyzer application. Marker data is not updated. More CPU time is spent making measurements instead of updating the display.

**Parameters**

<ON | OFF>    **ON** (or 1) - turns the display ON.  
                  **OFF** (or 0) - turns the display OFF.

**Examples**

```
DISP:ENAB ON  
display:enable off
```

**Query Syntax**    DISPlay:ENABle?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    ON

---

**DISPlay:FSIGn <ON | OFF>**

**(Read-Write)** Shows or hides the window which displays global pass/fail results.

**Parameters**

<ON | OFF>    **ON** (or 1) - displays the pass/fail dialog  
                  **OFF** (or 0) - hides the pass/fail dialog

**Examples**

```
DISP:FSIG ON  
display:fsign off
```

**Query Syntax**    DISPlay:FSIGn?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    OFF

---

**DISPlay:TMAX <bool>**

**(Read-Write)** Maximizes (isolates) or restores the active trace in the active window. When turned ON, the active trace is the **ONLY** trace on the display. All other traces are hidden. [Learn more.](#)

**Parameters**

<bool> **ON** (or 1) - Maximize / isolates the active trace.

**OFF** (or 0) - Restores other traces to the normal window setting.

**Examples**

```
DISP:TMAX ON
display:tmax 0
```

**Query Syntax** DISPlay:TMAX?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**DISPlay[:TILE] - Superseded**

This command is replaced by [DISP:ARRange](#)

**(Write-only)** Tiles the windows on the screen.

**Examples**

```
DISP
display:tile
```

**Default** Not Applicable

---

**DISPlay:WINDow<wnum>:ANNotation:MARKer:SINGle[:STATe] <bool>**

**(Read-Write)** Either shows marker readout of only the active trace or other traces simultaneously.

See other SCPI [Marker](#) commands. Learn more about [Marker readout](#).

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<bool> **ON** (or 1) - Shows the readout of only the active marker for each trace.

**OFF** (or 0) - Shows up to 5 marker readouts per trace, up to 20 total readouts.

**Examples**

```
DISP:WIND:ANN:MARK:SING ON
display>window:annotation:marker:single off
```

**Query Syntax** DISPlay:WINDow:ANNotation:MARKer:SINGle?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## DISPlay:WINDow<wnum>:ANNotation:MARKer:SIZE <char>

(Read-Write) Specifies the size of the marker readout text. See other SCPI [Marker](#) commands. Learn more about [Marker readout](#).

### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <char> Readout text size. Choose from: **NORMal** | **LARGE**

### Examples

```
DISP:WIND:ANN:MARK:SIZE LARG
display:window:annotation:marker:size normal
```

**Query Syntax** DISPlay:WINDow:ANNotation:MARKer:SIZE?

**Return Type** Character

**Default** NORMal

---

## DISPlay:WINDow<wnum>:ANNotation:MARKer:STATe <ON | OFF>

(Read-Write) Specifies whether to show or hide the Marker readout (when markers are ON) on the selected window. See other SCPI [Marker](#) commands. Learn more about [Marker readout](#).

### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <ON | OFF> **ON** (or 1) - turns marker readout ON.  
**OFF** (or 0) - turns marker readout OFF.

### Examples

```
DISP:WIND:ANN:MARK:STAT ON
display:window:annotation:marker:state off
```

**Query Syntax** DISPlay:WINDow:ANNotation:MARKer:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

## DISPlay:WINDow<wnum>:ANNotation:TRACe:STATe <ON | OFF>

**(Read-Write)** Specifies whether to show or hide the Trace Status buttons on the left of the display.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.  
<ON | OFF> **ON** (or 1) - turns the buttons ON.  
**OFF** (or 0) - turns the buttons OFF.

**Examples**

```
DISP:WIND:ANN:TRAC:STAT ON  
display:window:annotation:trace:state off
```

**Query Syntax** DISPlay:WINDow:ANNotation:TRACe:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**DISPlay:WINDow<wnum>:CATalog?**

**(Read-only)** Returns the trace numbers for the specified window.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

**Example**

```
Window 1 with four traces:  
DISPlay:WINDow1:CATalog?  
Returns:  
"1, 2, 3, 4"
```

**Return Type** String of Character values separated by commas

**Default** Not applicable

---

**DISPlay:WINDow<wnum>:ENABLE <ON | OFF>**

**(Read-Write)** Specifies whether to disable or enable all analyzer display information **in the specified window**. Marker data is not updated. More CPU time is spent making measurements instead of updating the display.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.  
<ON | OFF> **ON** (or 1) - turns the display ON.  
**OFF** (or 0) - turns the display OFF.

**Examples**

```
DISP:WIND:ENABLe ON  
display:window1:enable off
```

**Query Syntax** DISPlay:WINDow<wnum>:ENABLe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**DISPlay:WINDow<wnum>:SIZE <char>**

**(Read-Write)** Sets or returns the window setting of Maximized, Minimized, or Normal. To arrange all of the windows, use [DISP:ARR](#).

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1  
<char> Window size. Choose from:

**MIN | MAX | NORM**

**Examples**

```
DISP:WIND:SIZE MAX  
display:window:size norm
```

**Query Syntax** DISPlay:WINDow:SIZE?

**Default** Not Applicable

---

**DISPlay:WINDow<wnum>[:STATe] <ON | OFF>**

**(Read-Write) Write** to create or delete a window on the screen or **Read** whether a window is present.

**Parameters**

- <wnum> Window number to create; choose any integer between **1** and the [maximum number of windows allowed in the PNA](#).
- <ON | OFF> **ON** (or 1) - The window <wnum> is created.  
**OFF** (or 0) - The window <wnum> is deleted.

**Examples**

```
DISP:WIND ON  
display:window2:state off
```

**Query Syntax** DISPlay:WINDow<wnum>[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** Window number "1" **ON**

---

**DISPlay:WINDow<wnum>:TABLE <char>**

**(Read-Write) Write** to show the specified table at the bottom of the analyzer screen or **Read** to determine what table is visible.

**Parameters**

- <wnum> Any existing window number. If unspecified, value is set to 1
- <char> Table to show. Choose from:  
**OFF | MARKer | LIMit | SEGMENT**

**Examples**

```
DISP:WIND:TABLE SEGM  
display:window:table off
```

**Query Syntax** DISPlay:WINDow:TABLE?

**Default** OFF

---

**DISPlay:WINDow<wnum>:TITLe:DATA <string>**

**(Read-Write)** Sets data in the window title area. The title is turned ON and OFF with [DISP:WIND:TITL:STAT OFF](#).

**Parameters**

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <string> Title to be displayed. Any characters, enclosed with quotes. If the title string exceeds 50 characters, an error will be generated and the title not accepted. Newer entries replace (not append) older entries.

**Examples**

```
DISP:WIND:TITL:DATA 'hello'
display:window2:title:data 'hello'
```

**Query Syntax** DISPlay:WINDow<wnum>:TITLe:DATA?

**Return Type** String

**Default** NA

---

**DISPlay:WINDow<wnum>:TITLe[:STATe] <ON | OFF>**

**(Read-Write)** Turns display of the title string ON or OFF. When OFF, the string remains, ready to be redisplayed when turned back ON.

**Parameters**

- <wnum> Any existing window number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - turns the title string ON.  
**OFF** (or 0) - turns the title string OFF.

**Examples**

```
DISP:WIND:TITL ON
Display:window1:title:state off
```

**Query Syntax** DISPlay:WINDow<wnum>:TITLe[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**DISPlay:WINDow<wnum>:TRACe<tnum>:DELeTe**



**(Write-only)** Deletes the specified trace from the specified window. The measurement parameter associated with the trace is not deleted.

#### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <tnum> The number of the trace to be deleted; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used ONLY for remote programs.

#### Examples

```
DISP:WIND:TRAC:DEL  
display>window2:trace2:delete
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### DISPlay:WINDow<wnum>:TRACe<tnum>:FEED <name>

**(Write-only)** Creates a new trace <tnum> and associates (feeds) a measurement <name> to the specified window<wnum>. This command should be executed immediately after creating a new measurement with [CALC:PAR:DEF<name>,<parameter>](#).

To feed the same measurement to multiple traces, create another measurement with the same <name>,<parameter> using the CALC:PAR:DEF command. The analyzer will collect the data only once.

#### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <tnum> Trace number to be created. Choose any Integer between 1 and the PNA [maximum number of traces per window](#) allowed.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used ONLY for remote programs.

- <name> Name of the measurement that was defined with  
CALC:PAR:DEF<name>,<parameter>

#### Examples

```
DISP:WIND:TRAC:FEED 'test'  
display>window2:trace2:feed 'test'
```

**Query Syntax** Not applicable

**Default** "CH1\_S11"

---

## DISPlay:WINDow:TRACe:GRATicule:GRID:LTYPE <value>

**(Read-Write)** Sets and returns the grid line type (solid | dotted) for all open windows. Grid is returned to solid when the PNA is Preset. [Learn more.](#)

### Parameters

<value> Line type. Choose from:

**SOLid** - solid lines

**DOTTed** - dotted lines

### Examples

```
DISP:WIND:TRAC:GREAT:GRID:LTYPE SOL
display>window:trace:graticule:grid:ltype dotted
```

**Query Syntax** DISPlay:WINDow:TRACe:GRATicule:GRID:LTYPE?

**Return Type** Character

**Default** SOLID

---

## DISPlay:WINDow<wnum>:TRACe<tnum>:MEMory[:STATe] <ON | OFF>

**(Read-Write)** Turns the memory trace ON or OFF.

### Parameters

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<ON | OFF> **ON** (or 1) - turns the memory trace ON.  
**OFF** (or 0) - turns the memory trace OFF.

### Examples

```
DISP:WIND:TRAC:MEM ON
display>window2:trace2:memory:state off
```

**Query Syntax** DISPlay:WIND<wnum>:TRACe<tnum>:MEMory[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## DISPlay:WINDow<wnum>:TRACe<tnum>:SElect

**(Write-only)** Activates the specified trace in the specified window for front panel use.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

**Examples**

```
DISP:WIND:TRAC:SEL  
display:window2:trace2:select
```

**Query Syntax** Not applicable

**Default** NA

---

**DISPlay:WINDow<wnum>:TRACe<tnum>[:STATe] <ON | OFF>**

**(Read-Write)** Turns the display of the specified trace in the specified window ON or OFF. When OFF, the measurement behind the trace is still active.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<ON | OFF> **ON** (or 1) - turns the trace ON.  
**OFF** (or 0) - turns the trace OFF.

**Examples**

```
DISP:WIND:TRAC ON  
display:window2:trace2:state off
```

**Query Syntax** DISPlay:WIND<wnum>:TRACe<tnum>[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**DISPlay:WINDow<wnum>:TRACe<tnum>TITLe:DATA <string>**

**(Read-Write)** Writes and read data to the trace title area. The trace title is embedded in the [trace status field](#). [Learn more](#).

Newer entries replace (not append) older entries. The title is turned ON and OFF with [DISP:WIND:TRAC:TITL:STAT](#).

#### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1.
- <tnum> Trace number of the specified window. If unspecified, value is set to 1. Use [Display:Cat?](#) to read the window numbers. Use [Disp:Window:Cat?](#) to read the trace numbers of the specified window.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

- <string> Title to be displayed. Any characters (not spaces) enclosed with quotes.

#### Examples

```
DISP:WIND:TRAC:TITL:DATA 'MyNewMeas'  
display>window2:trace3:title:data 'hello'
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe<tnum>TITLe:DATA?

**Return Type** String

**Default** Not Applicable

---

### DISPlay:WINDow<wnum>:TRACe<tnum>:TITLe:STATe <bool>

**(Read-Write)** Turns display of the Trace Title ON or OFF. When turned OFF, the previous trace title returns. Set a new trace title using [DISP:WIND:TRAC:TITL:DATA](#)

[Learn more](#) about Trace Titles

#### Parameters

- <wnum> Any existing window number. If unspecified, value is set to 1
- <tnum> Trace number of the specified window. If unspecified, value is set to 1. Use [Display:Cat?](#) to read the window numbers. Use [Disp:Window:Cat?](#) to read the trace numbers of the specified window.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<bool> **ON** (or 1) - turns the title ON.  
**OFF** (or 0) - turns the title OFF.

**Examples**

```
DISP:WIND:TRAC:TITL ON  
Display:window2:trace3:title:state off
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe<tnum>:TITLe[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

## DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALE]:AUTO

(Write-only) Performs an **Autoscale** on the specified trace in the specified window, providing the best fit display. Autoscale is performed only when the command is sent; it does NOT keep the trace autoscaled indefinitely.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used ONLY for remote programs.

**Examples**

```
DISP:WIND:TRAC:Y:AUTO  
display:window2:trace2:y:scale:auto
```

**Query Syntax** Not applicable

**Default** Not applicable

---

## DISPlay:WINDow:TRACe:Y[:SCALE]:COUPle:METHod <char>

**(Read-Write)** Sets and returns the method of scale coupling. [Learn more](#) about Scale coupling.

**Parameters**

<char> **OFF** - NO scale coupling for any windows.

**WINDow** - Scale settings are coupled for traces in each window.

**ALL** - Scale settings are coupled for traces in ALL selected windows.

Enable the selected windows using [DISP:WIND:TRAC:Y:COUP ON](#)

**Examples**

```
DISP:WIND:TRAC:Y:COUP:METH ALL
```

```
Display>window2:trace:y:scale:method window
```

**Query Syntax** DISPlay:WINDow:TRACe:Y[:SCALe]:COUPle:METHod?

**Return Type** Character

**Default** OFF

---

**DISPlay:WINDow<wnum>:TRACe:Y[:SCALe]:COUPle[:STATe] <bool>**

**(Read-Write)** Enables and disables scale coupling for the specified window. [Learn more](#) about Scale coupling.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1

<bool> **ON** (or 1) - Scale coupling enabled for specified window.

**OFF** (or 0) - Scale coupling disabled for specified window.

**Examples**

```
DISP:WIND:TRAC:Y:COUP ON
```

```
Display>window2:trace:y:scale:couple:state off
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe:Y[:SCALe]:COUPle[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALe]:PDIVision <num>**

**(Read-Write)** Sets the Y axis **Per Division** value of the specified trace in the specified window.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<num> Units / division value. The range of acceptable values is dependent on format and domain.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
DISP:WIND:TRAC:Y:PDIV 1
display>window2:trace2:y:scale:pdivision maximum
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALE]:PDIVision?

**Return Type** Numeric

**Default** 10

---

**DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALE]:RLEVel <num>**

**(Read-Write)** Sets the Y axis Reference Level of the specified trace in the specified window.

**Parameters**

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<num> Reference level value. The range of acceptable values is dependent on format and domain.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
DISP:WIND:TRAC:Y:RLEV 0
display>window2:trace2:y:scale:rlevel minimum
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALE]:RLEVel?

**Return Type** Numeric  
**Default** Not Applicable

---

## DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALe]:RPOSition <num>

(Read-Write) Sets the **Reference Position** of the specified trace in the specified window

### Parameters

<wnum> Any existing window number. If unspecified, value is set to 1.

<tnum> Any existing trace number; if unspecified, value is set to 1.

**Note:** This is **NOT** the trace number of the channel which appears as the [Tr annotation](#) on the Trace Status display. This <tnum> is the trace number within the specified window, and is used **ONLY** for remote programs.

<num> Reference position on the screen measured in horizontal graticules from the bottom. The range of acceptable values is dependent on format and domain.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
DISP:WIND:TRAC:Y:RPOS 0
display:window2:trace2:y:rposition maximum
```

**Query Syntax** DISPlay:WINDow<wnum>:TRACe<tnum>:Y[:SCALe]:RPOSition?

**Return Type** Numeric

**Default** 5

---

## DISPlay:VISible <ON | OFF>



**(Read-Write)** Makes the PNA application visible or not visible. In the Not Visible state, the analyzer cycle time for making measurements, and especially data transfer, can be significantly faster because the display does not process data.

**Parameters**

<ON | OFF>    **ON** (or 1) - PNA app is visible  
                  **OFF** (or 0) - PNA app is NOT visible

**Examples**

```
DISP:VIS ON  
display:visible off
```

**Query Syntax**    DISPlay:VISible?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    ON

---

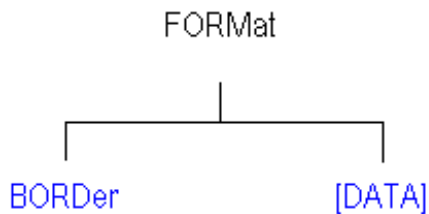
Last modified:

16-Mar-2010	Added grid line (9.2)
23-Mar-2009	Added <tnum> note
26-Aug-2008	Added TraceMax
9/12/06	Modified for number of windows.

## Format Commands

---

Specifies the way that data will be transferred when moving large amounts of data.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### FORMat:BORDER <char>

**(Read-Write)** Set the byte order used for GPIB data transfer. Some computers read data from the analyzer in the reverse order. This command is only implemented if FORMAT:DATA is set to :REAL.

If FORMAT:DATA is set to :ASCII, the swapped command is ignored.

#### Parameters

<char> Choose from:

**NORMAL** - Use when your controller is anything other than an IBM compatible computers.

**SWAPped** - for IBM compatible computers.

**Note:** Use **NORMAL** if you are using VEE, LabView, or T&M Tool kit.

#### Examples

```
FORM:BORD SWAP
format:border normal
```

**Query Syntax** FORMat:BORDER?

**Return Type** Character

Default Normal

---

### FORMat[:DATA] <char>

(Read-Write) Sets the data format for data transfers.

- To transfer measurement data, use [CALC:DATA](#).
- To transfer Cal Set data, use [SENS:CORR:CSET:DATA](#)
- To transfer Source Power correction data, use:
  - [SOURce:POWer:CORRection:COLLect:TABLE:DATA](#)
  - [SOURce:POWer:CORRection:COLLect:TABLE:FREQuency](#)
  - [SOURce:POWer:CORRection:DATA](#)

### Parameters

<char> Choose from:

- **REAL,32** - (default value for REAL) Best for transferring large amounts of measurement data.
- **REAL,64** - Slower but has more significant digits than REAL,32. Use REAL,64 if you have a computer that doesn't support REAL,32.
- **ASCii,0** - The easiest to implement, but very slow. Use when you have small amounts of data to transfer.

### Notes:

- The higher frequencies used on PNA can exceed the maximum value that can be represented by a 32-bit floating point number.
- The **REAL,32** and **REAL,64** arguments transfer data in block format as explained in [Transferring Measurement Data](#).

### Examples

```
FORM REAL, 64
format:data ascii
```

**Query Syntax** FORMat:DATA?

**Return Type** Character,Character

**Default** **ASCii,0**

Syst:Preset does NOT reset this command.

However, \*RST does reset this command to ASCii,0

---

Last Modified:

18-May-2009    Added Real 64 note

17-Sep-2008    Added \*RST vs Syst:Pres note.

## Hardcopy Command

---

Controls printing of the PNA screen and optional data to a printer or a file.

### HCOPY:

[DPrinter](#)

[FILE](#)

[\[IMMediate\]](#)

**ITEM**

| [AWINdow](#)

| [CTABle](#)

| [GPFail](#)

| [LOGO](#)

| [MKRData](#)

| [PNUMber](#)

| [SEGData](#)

| [SWINdow](#)

| [TIME](#)

| [TTABle](#)

| [WFRaction](#)

| [WINDowns](#)

**PAGE**

| [DIMension](#)

| [LLEFt](#)

| [URIGHt](#)

| [ORIENTATION](#)

| [SIZE](#)

**SDUMP**

| [DATA?](#)

| [FORMat](#)

[PRINters?](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [Learn more about PNA Printing](#)
- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)

## HCOPy:DPRinter <string>

(Read-Write) Sets the default printer and selects as the current printer. Use [HCOPy:PRINters?](#) to return a list of locally installed printers.

This setting survives instrument preset and PNA application restart.

### Parameters

<string> Name of the printer to become the default.

### Examples

```
HCOP:DPR "MyPrinter"
hcopy:dprinter "YourPrinter"
```

Query Syntax HCOPy:DPRinter?

Return Type String

Default Not Applicable

---

## HCOPy:FILE <filename>

(Write-only) Saves the screen image to a file. The image does NOT include the optional print data invoked by many HCOPy commands.

### Parameters

<filename> Name of the file to save the screen to. The file is saved to the current working directory unless a valid full path name is specified.

Use one of the following suffixes:

.bmp - not recommended due to large file size

.jpg - not recommended due to poor quality

.png - recommended

### Examples

```
HCOPY:FILE "myFile.png"
hcopy:file "c:/data/myfile.png"
```

Query Syntax Not Applicable

Default Not Applicable

---

## HCOPy[:IMMediate]

(Write-only) Prints the screen to the default printer.

**Examples**

```
HCOP
hcopy:immediate
```

**Query Syntax** Not applicable

**Default** Not Applicable

---

## HCOPY:ITEM:AWINDOW[:STATE] <bool>

(Read-Write) When ON, prints only the Active window. When OFF, prints all windows.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Active window state. Chose from:

**OFF** or (0) - Print ALL windows.

**ON** or (1) - Print Active window only.

**Examples**

```
HCOP:ITEM:AWIN 1
hcopy:item:awindow:state off
```

**Query Syntax** HCOPI:ITEM:AWINDOW[:STATE]?

**Return Type** Boolean

**Default** OFF (0)

---

## HCOPY:ITEM:CTABLE[:STATE] <bool>

**(Read-Write)** When ON, prints the channel settings table.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Channel table print state. Chose from:

**OFF** or (0) - Does NOT print the channel settings table.

**ON** or (1) - Prints channel settings table.

**Examples**

```
HCOP:ITEM:CTAB 1
```

```
hcopy:item:ctable:state off
```

**Query Syntax** HCOPy:ITEM:CTABle[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

**HCOPy:ITEM:GPFail[:STATe] <bool>**

**(Read-Write)** When ON, prints the [Global Pass/Fail](#) status in the page header.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Pass / Fail print state. Chose from:

**OFF** or (0) - Does NOT print Pass / Fail status.

**ON** or (1) - Print Pass / Fail status

**Examples**

```
HCOP:ITEM:GPF 1
```

```
hcopy:item:gpfail:state off
```

**Query Syntax** HCOPy:ITEM:GPFail[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

**HCOPy:ITEM:LOGO[:STATe] <bool>**



**(Read-Write)** When ON, prints the Agilent Technologies logo in the page header.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Agilent logo print state. Chose from:

**OFF** or (0) - Prints the Agilent logo.

**ON** or (1) - Does NOT print the Agilent logo.

**Examples**

```
HCOPY:ITEM:LOGO 1
```

```
hcopy:item:logo:state off
```

**Query Syntax** HCOpy:ITEM:LOGO[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

**HCOPY:ITEM:MKRData[:STATe] <bool>**

**(Read-Write)** When ON, includes marker data as part of the [trace attributes table](#).

To print marker data, [HCOPY:ITEM:TTable](#) must also be set to ON.

This setting does not affect the limited [marker readout data](#) that can be displayed in the measurement window.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Marker data print state. Chose from:

**OFF** or (0) - Does NOT print Marker data.

**ON** or (1) - Print Marker data.

**Examples**

```
HCOPY:ITEM:MKRD 1
```

```
hcopy:item:mkrdata:state off
```

**Query Syntax** HCOpy:ITEM:MKRData[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

## HCOPy:ITEM:PNUMber[:STATe] <bool>

(Read-Write) When ON, prints page numbers (1 of n) in the header at the top of each page.

This setting survives instrument preset and PNA application restart.

### Parameters

<bool> Page number print state. Chose from:

**OFF** or (0) - Does NOT print page numbers.

**ON** or (1) - Print page numbers.

### Examples

```
HCOP:ITEM:PNUM 1
```

```
hcopy:item:pnumber:state off
```

**Query Syntax** HCOPy:ITEM:PNUMber[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

## HCOPy:ITEM:SEGData[:STATe] <bool>

(Read-Write) When ON, includes ALL segment data as part of the [channel settings table](#).

To print ALL segment data, [HCOP:ITEM:CTAB](#) must also be set to ON.

This setting survives instrument preset and PNA application restart.

### Parameters

<bool> Expanded segment data print state. Chose from:

**OFF** or (0) - Does NOT print expanded segment data, but summary data is printed.

**ON** or (1) - Print expanded segment data.

### Examples

```
HCOP:ITEM:SEGD 1
```

```
hcopy:item:segdata:state off
```

**Query Syntax** HCOPy:ITEM:SEGData[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

### HCOPy:ITEM:SWINdow[:STATe] <bool>

(Read-Write) When ON, prints a single measurement window per page. When OFF, prints up to four measurement windows per page.

This setting survives instrument preset and PNA application restart.

#### Parameters

<bool> Single window print state. Chose from:  
  
**OFF** or (0) - Print up to four windows per page.  
  
**ON** or (1) - Print only one window per page.

#### Examples

```
HCOP:ITEM:SWIN 1  
hcopy:item:swindow:state off
```

**Query Syntax** HCOPy:ITEM:SWINdow[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

### HCOPy:ITEM:TIME[:STATe] <bool>

(Read-Write) When ON, prints the PNA computer date and time in the header.

This setting survives instrument preset and PNA application restart.

#### Parameters

<bool> Time stamp print state. Chose from:  
  
**OFF** or (0) - Does NOT print time stamp.  
  
**ON** or (1) - Print time stamp.

#### Examples

```
HCOP:ITEM:TIME 1  
hcopy:item:time:state off
```

**Query Syntax** HCOPy:ITEM:TIME:[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

### HCOPy:ITEM:TTABLE[:STATe] <bool>

**(Read-Write)** When ON, prints the trace attributes table.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Trace attributes table print state. Chose from:

**OFF** or (0) - Does NOT print the trace attributes table.

**ON** or (1) - Print the trace attributes table.

**Examples**

```
HCOP:ITEM:TTable 1
```

```
hcopy:item:ttable:state off
```

**Query Syntax** HCOPy:ITEM:TTable[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

**HCOPy:ITEM:WFRaction[:STATe] <value>**

**(Read-Write)** Sets the vertical amount of a page that is filled by the measurement windows.

This setting survives instrument preset and PNA application restart.

**Parameters**

<value> Window size as a fraction of the page. Chose a value from .4 (40%) to 1.0 (100%)

**Examples**

```
HCOP:ITEM:WFR .8
```

```
hcopy:item:wfraction .5
```

**Query Syntax** HCOPy:ITEM:WFRaction?

**Return Type** Numeric

**Default** .4

---

**HCOPy:ITEM:WINDows[:STATe] <bool>**

**(Read-Write)** When ON, prints measurement windows.

Use [HCOpy:ITEM:AWINdow](#) to specify all windows or only the active window.

This setting survives instrument preset and PNA application restart.

**Parameters**

<bool> Windows print state. Chose from:

**OFF** or (0) - Does not print measurement windows.

**ON** or (1) - Print measurement windows.

**Examples**

```
HCOP:ITEM:WIND 1
```

```
hcopy:item:windows:state off
```

**Query Syntax** HCOPy:ITEM:WINDows[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

**HCOpy:PAGE:DIMension:LLEft <left, lower>**

**(Read-Write)** Sets the left and lower page margins.

This setting survives instrument preset and PNA application restart.

**Parameters**

<left> Left page margin as a percentage of entire page width. Value must be between 0 and 1.

<lower> Lower page margin as a percentage of entire page length. Value must be between 0 and 1.

**Examples**

```
HCOP:PAGE:DIM:LLEF .10, .10
```

```
hcopy:page:dimension:lleft .5, .7
```

**Query Syntax** HCOPy:PAGE:DIMension:LLEft?

**Return Type** Numeric, Numeric

**Default** Depends on selected page size

---

**HCOpy:PAGE:DIMension:URIGht <right, upper>**

**(Read-Write)** Sets the right and upper page margins.

This setting survives instrument preset and PNA application restart.

**Parameters**

<right> Right page margin as a percentage of entire page width. Value must be between 0 and 1.

<upper> Upper page margin as a percentage of entire page length. Value must be between 0 and 1.

**Examples**

```
HCOPY:PAGE:DIM:URIG .10,.10
```

```
hcopy:page:dimension:uright .5,.7
```

**Query Syntax** HCOpy:PAGE:DiMension:URiGht?

**Return Type** Numeric, Numeric

**Default** Depends on selected page size

---

## HCOPY:PAGE:ORientation <char>

**(Read-Write)** Sets the page orientation.

This setting survives instrument preset and PNA application restart.

**Parameters**

<char> Choose from:

- PORTrait
- LANDscape

**Examples**

```
HCOPY:PAGE:ORI PORT
```

```
hcopy:page:orientation landscape
```

**Query Syntax** HCOpy:PAGE:ORientation?

**Return Type** Character

**Default** PORTrait

---

## HCOPY:PAGE:SIZE <int>

**(Read-Write)** Sets the paper type, which implies the page size.

This setting survives instrument preset and PNA application restart.

### Parameters

<int> Choose from: I

Integer	Description
1	Letter 8 1/2 x 11 in
2	Letter Small 8 1/2 x 11 in
3	Tabloid 11 x 17 in
4	Ledger 17 x 11 in
5	Legal 8 1/2 x 14 in
6	Statement 5 1/2 x 8 1/2 in
7	Executive 7 1/4 x 10 1/2 in
8	A3 297 x 420 mm
9	A4 210 x 297 mm
10	A4 Small 210 x 297 mm
11	A5 148 x 210 mm
12	B4 (JIS) 250 x 354
13	B5 (JIS) 182 x 257 mm

For more paper type choices, see Microsoft's "wingdi.h" file, which can be downloaded as part of the Platform SDK.

### Examples

```
HCOP:PAGE:SIZE 2
```

```
hcopy:page:size 5
```

**Query Syntax** HCOPy:PAGE:SIZE?

**Return Type** Integer

**Default** 1

---

**HCOPy:SDUMp:DATA?**

**(Read-only)** Returns the display image in a definite-length arbitrary binary block. The format of the data is PNG by default. Use [HCOP:SDUMp:DATA:FORMat](#) to change the format.

This command is equivalent to saving an image to the PNA ([HCOPy:FILE](#)) and then using [MMEM:TRAN](#) to transfer the file to the computer.

**Examples** `HCOP : SDUM?`

`hcopy : sdump?`

**Return Type** A definite-length arbitrary binary block

**Default** Not Applicable

---

### **HCOPy:SDUMp:DATA:FORMat <char>**

**(Read-Write)** Sets the graphic format for [HCOPy:SDUMp:DATA?](#)

#### **Parameters**

<char> Choose from: **JPG** | **BMP** | **PNG**

**Examples** `HCOP : SDUMp : DATA : FORMat BMP`

**Query Syntax** `HCOPy:SDUMp:DATA:FORMat?`

**Return Type** Character

**Default** PNG

---

### **HCOPy:PRINters?**

**(Read-only)** Returns a comma-separated list of printers installed on the PNA. Select a printer using [HCOPy:DPRinter](#).

This setting survives instrument preset and PNA application restart.

**Examples** `HCOP : PRIN?`

`hcopy : printers?`

**Query Syntax** `HCOPy:PRINters?`

**Return Type** String

**Default** Not Applicable

---

Last modified:

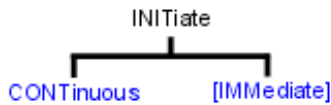


15-Mar-2010 Added SDUMP commands (A.09.20)  
Nov. 1, 2006 Added new commands

## Initiate Commands

---

Controls triggering signals



Click on a [blue](#) keyword to view the command details.

### See Also

- **Example** [Triggering the PNA](#)
  - [Learn about Triggering](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

### INITiate:CONTInuous <boolean>

**(Read-Write)** Specifies whether the PNA trigger source is set to Internal (continuous) or Manual.

This command is a subset of [TRIG:SEQ:SOURce](#), which can also set the trigger source to External.

To set how a channel responds to trigger signals, use [SENS:SWE:MODE](#).

See a [map of user interface to SCPI triggering commands](#).

For more information on triggering, see the [PNA Trigger Model](#).

**See the Example program:** [Triggering the PNA using SCPI](#).

#### Parameters

<boolean> **ON** (or 1) - Internal (continuous) trigger.

**OFF** (or 0) - Manual sweep. Use [INIT:IMMediate](#) to send a trigger signal

#### Examples

```
INIT:CONT ON  
initiate:continuous off
```

**Query Syntax** INITiate:CONTInuous?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

## INITiate<cnum>[:IMMediate]

(Write-only) Stops the current sweeps and immediately sends a trigger. (Same as [Trigger!](#) on the PNA front panel).

See the Example program: [Triggering the PNA using SCPI](#)

**Note:** An SMC Fixed Output measurement cannot be triggered using this command. For more information, see the [example program](#).

This command requires [Trigger:Source](#) to be set to Manual. This causes ONE trigger signal to be SENT each time INIT:IMM is issued.

[Sens<ch>:Sweep:Mode](#) sets the number of trigger signals each channel will ACCEPT (Continuous - unlimited, Groups - a specific number, or HOLD - none.)

### To trigger ALL channels in turn:

Set ALL channels to Sens<ch>:Sweep:Mode Continuous. The <ch> argument in INIT<ch>:IMM is ignored.

Then...

- TRIG:SCOP ALL triggers ALL channels (in sequence) each time Init:Imm is sent.
- TRIG:SCOP CURRent triggers ONLY the NEXT channel each time Init:Imm is sent.

### To trigger ONLY a specified channel:

1. Set ALL channels to Sens<ch>:Sweep:Mode HOLD
2. Send TRIG:SCOP CURRent
3. Send Init<ch>:Imm where <ch> is the channel to be triggered.

**Advanced** Situations that require some channels to be in CONT and others in HOLD are rare. The following describes the behavior of the Init:Imm command in these situations:

### When [Trigger:Scope](#) = Global:

- If the SPECIFIED <cnum> channel is in hold mode, it is put in single trigger (accepts 1 trigger signal) and goes to the end of the queue of channels to be triggered. The other 'non-hold' channels are triggered. The next Init:Imm triggers the specified channel first.

For example: ch1 is in Hold, ch2 and ch3 are in CONT and we send INIT1:IMM

- On the first INIT:IMM, ch2 and ch3 is triggered.
- next INIT:IMM, ch1, ch2, ch3 is triggered.
- next INIT:IMM, ch2 and ch3 is triggered.
- next INIT:IMM, ch1, ch2, ch3 is triggered, and so forth.

### When Trigger:Scope = Channel

- Only ONE channel is triggered for each issued INIT<ch>:IMM command.
- If the specified channel is in hold, it is put in single trigger (accepts 1 trigger signal) and goes the end of the queue of channels to be triggered as in the 'Global' example.

This is one of the PNA overlapped commands. [Learn more.](#)

#### Parameters

<cnm> Any existing channel number. If unspecified, value is set to 1

#### Examples

```
INIT
initiate2:immediate
```

#### Query Syntax

Not applicable

#### Default

Not applicable

Last modified:

April 23, 2007    Updated Init:Imm

## Memory Commands

---

The memory commands control saving and loading instrument states and measurement trace data to the hard drive. To read and write trace data in GPIB format, see [CALC:DATA](#).

### MMEMory:

[CATalog?](#)

[CDIRectory](#)

[COPY](#)

[DELeTe](#)

[LOAD](#)

[MDIRectory](#)

[MOVE](#)

[RDIRectory](#)

[STORe](#)

| [CITI](#)

| [DATA](#)

| [FORMat](#)

| [DATA](#)

| [ENR](#)

| [TRACe](#)

| [CONTents](#)

| [CITIfile](#)

| [FORMat](#)

| [CITIfile](#)

| [SNP](#)

[TRANsfer](#)

Click on a [blue](#) keyword to view the command details.

**Red** commands are [superseded](#).

### See Also

- [Example Programs](#)
- [Learn about Save / Recall and File Types](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

## Specifying Path Names

The MMEM commands use the following rules to specify path names:

- The default folder is "C:/Program Files/Agilent/Network Analyzer/Documents"
  - You can change the active directory using [MMEMory:CDIRectory](#).
  - Specify only the file name if using the active directory.
  - You can also use an absolute path name to specify the folder and file.
- 

## MMEMory:CATalog[:<char>]? [<folder>]

**(Read-only)** Returns a comma-separated string of file names that are in the specified folder. If there are no files of the specified type, "NO CATALOG" is returned. [Learn about File Types](#)

### Parameters

<char> The type of files to list. Choose from:

- **STATe** - Instrument states (.sta)
- **CORRection** - Calibration Data (.cal)
- **CSARchive** - Instrument state and calibration data (.csa)
- **CSTate** - Instrument state and link to Calibration data (.cst)

If unspecified then ALL file types (even unknown types) are listed.

<folder> String - Any existing folder name. See [Specifying Path Names](#)

### Examples

```
MMEM:CAT? 'lists all files from the current folder  
mmemory:catalog:correction? 'C:/Program Files/Agilent/Network  
Analyzer/Documents' 'lists .cal files from the specified folder
```

Default Not applicable

---

## MMEMory:CDIRectory <folder>

(Read-Write) Changes the folder name.

#### Parameters

<folder> Any drive and folder name that already exists.  
If the same level as "C:/Program Files/Agilent/Network Analyzer/Documents", then no punctuation is required

```
MMEM:CDIR Service
```

If the new folder is at a different level than "C:/Program Files/Agilent/Network Analyzer/Documents", use a slash (/) before the folder name and enclose in quotes.

```
mmemory:cdirectory '/automation' 'changes default directory up one level.'
```

You can use an absolute path to specify the new folder.

```
mmemory:cdirectory 'c:/automation/service'
```

**Query Syntax** MMEMory:CDIRectory? 'Returns the current folder name

**Return Type** String

**Default** 'C:/Program Files/Agilent/Network Analyzer/Documents'

---

#### MMEMory:COPY <file1>,<file2>

(Write-only) Copies file1 to file2. Extensions must be specified.

#### Parameters

<file1> String - Name of the file to be copied. See [Specifying Path Names](#)

<file2> String - Name of the file to be created from file1.

**Examples**

```
MMEM:COPY 'MyFile.cst', 'YourFile.cst'
```

**Query Syntax** Not applicable

**Default** Not applicable

---

#### MMEMory:DELeTe <file>

(Write-only) Deletes file. Extensions must be specified.

#### Parameters

<file> String - Name of the file to be deleted. See [Specifying Path Names](#)

#### Examples

```
MMEM:DEL 'MyFile.cst'
```

**Query Syntax** Not applicable

**Default** Not applicable

---

### MMEMory:LOAD[:<char>] <file>

(Write-only) Loads the specified file. [Learn about File Types](#)

#### Parameters

<char> The type of file to load. Choose from:

- **STATe** - Instrument states (.sta)
- **CORRection** - Calibration Data (.cal)
- **CSARchive** - Instrument state and calibration data (.csa)
- **CSTate** - Instrument state and link to Calibration data (.cst)
- **ENR** - Excess Noise Source data ([Noise Figure App only](#))

If <char> is unspecified, the extension must be included in the filename.

If an extension is specified in <file> that does not agree with <char> then no action is taken.

<file> String - Name of the file to be loaded. See [Specifying Path Names](#)

When <char> is **ENR**, then include **CAL**, See example below.

#### Examples

```
MMEM:LOAD 'MyFile.cst'  
mmemory:load:state 'MyInstState'  
  
MMEM:LOAD:ENR CAL, C:/data/calset/346C_16500.enr"
```

**Query Syntax** Not applicable

**Default** Not applicable

---

### MMEMory:MDIRectory <folder>



(Write-only) Makes a folder.

**Parameters**

<folder> String - Name of the folder to make. See [Specifying Path Names](#)

**Examples**

```
MMEM:MDIR 'MyFolder'  
mmemory:mdirectory 'c:/NewFolder'
```

**Query Syntax** Not applicable

**Default** Not applicable

---

**MMEMory:MOVE <file1>,<file2>**

(Write-only) Renames <file1> to <file2>. File extensions must be specified.

**Parameters**

<file1> String - Name of the file to be renamed. See [Specifying Path Names](#)

<file2> String - Name of the new file.

**Examples**

```
MMEM:MOVE 'MyFile.cst', 'YourFile.cst'
```

**Query Syntax** Not applicable

**Default** Not applicable

---

**MMEMory:RDIRectory <folder>**

(Write-only) Removes the specified folder.

**Parameters**

<folder> String - Name of the folder to remove. See [Specifying Path Names](#)

**Examples**

```
MMEM:RDIR 'MyFolder'
```

**Query Syntax** Not applicable

**Default** Not applicable

---

**MMEMory:STORe[:<char>] <file>**

(Write-only) Saves instrument / calibration state files.

To save data files, use [MMEM:STOR:DATA](#).

To save ENR files, use [MMEMory:STORe:ENR](#)

### Parameters

<char> Optional argument. The type of file to store. Choose from:

- **STATe** - Instrument states (.sta)
- **CORRection** - Calibration Data (.cal)
- **CSARchive** - Instrument state and calibration data (.csa)
- **CSTate** - Instrument state and link to Calibration data (.cst)

Include either <char> or the file extension. If both <char> and the extension are specified, they must agree or an error is returned and no action is taken. See examples below.

[Learn about File Types](#)

<file> String - Name of any valid file that does not already exist. See [Specifying Path Names](#)

### Examples

```
MMEM:STOR:STAT 'myState'  
mmemory:store 'c:/bin/myState.sta'
```

Query Syntax Not applicable

Default Not applicable

---

**MMEMory:STORe:CITifile:DATA <filename> - Superseded**

This command is replaced with [MMEMory:STORe:DATA](#).

(Write only) Saves UNFORMATTED trace data to .cti file. [Learn more](#).

#### Parameters

<filename> Any drive and folder name that already exists.  
If the same level as "C:/Program Files/Agilent/Network Analyzer/Documents", then no punctuation is required

```
MMEM:STOR:CIT:DATA MYCTIFile
```

If the new folder is at a different level than "C:/Program Files/Agilent/Network Analyzer/Documents", use a slash (/) before the folder name and enclose in quotes.

```
mmemory:cdirectory '/automation' 'changes default directory up  
one level.'
```

You can use an absolute path to specify the new folder.

```
mmemory:cdirectory 'c:/automation/service'
```

Query Syntax Not Applicable

Default C:/Program Files/Agilent/Network Analyzer/Documents'

---

### MMEMory:STORe:CITfile:FORMat <filename> - Superseded

This command is replaced with [MMEMory:STORe:DATA](#).

(Write only) Saves FORMATTED trace data to .cti file. [Learn more](#).

#### Parameters

<filename> Any drive and folder name that already exists.  
If the same level as "C:/Program Files/Agilent/Network Analyzer/Documents", then no punctuation is required

```
MMEM:STOR:CIT:DATA MYFile
```

If the new folder is at a different level than "C:/Program Files/Agilent/Network Analyzer/Documents", use a slash (/) before the folder name and enclose in quotes.

```
mmemory:cdirectory '/automation' 'changes default directory up  
one level.'
```

You can use an absolute path to specify the new folder.

```
mmemory:cdirectory 'c:/automation/service'
```

**Query Syntax** Not Applicable

**Default** C:/Program Files/Agilent/Network Analyzer/Documents'

---

## **MMEMory:STORe:DATA <filename>,<type>,<scope>,<format>,<selector>**

(Write-only) Stores trace data to the following file types: \*.prn, \*.cti, \*.csv, \*.mdf

To save snp files, use [Calc:Data:SNP:PORTs:SAVE](#)

To save state and calibration files, use [MMEM:STORe](#)

This command replaces the following:

- [MMEMory:STORe:CITifile:DATA](#)
- [MMEMory:STORe:CITifile:FORMat](#)
- [MMEMory:STORe:TRACe:FORMat:CITifile](#)
- [MMEMory:STORe:TRACe:CONTent:CITifile](#)

### **Parameters**

<filename> (String) Name of the file in which data will be saved.

[See rules for specifying a filename.](#)

Choose from the following valid parameter combinations for ALL measurement classes:

Type of file to save	Parameters			
	<type> (String)	<scope> (String)	<format> (String)	<selector> (Numeric)
Click to learn about these file types:				
<a href="#">*.prn</a>	"PRN Trace Data"	"Trace"	"Displayed"	Measurement number
Example: MMEMory:STORe:DATA "myData.prn","PRN Trace Data","Trace","Displayed",2				
<a href="#">*.cti</a> (unformatted)	"Citifile Data"	"Trace" or "Auto"	"RI"	Measurement number

Example: MMEMory:STORe:DATA "myData.cti","Citifile Data Data","AUTO","RI",3

<a href="#">*.cti</a> (unformatted)	"Citifile Data Data"	"Displayed"	"RI"	-1
--	-------------------------	-------------	------	----

Example: MMEMory:STORe:DATA "myData.cti","Citifile Data Data","AUTO","RI",3

<a href="#">*.cti</a> (formatted)	"Citifile Formatted Data"	"Trace" or "Auto"	"RI" or "MA" or "DB"	Measurement number
-----------------------------------	------------------------------	-------------------	-------------------------	-----------------------

Example: MMEMory:STORe:DATA "myData.cti","Citifile Formatted Data","AUTO","MA",3

<a href="#">*.cti</a> (formatted)	"Citifile Formatted Data"	"Displayed"	"RI" or "MA" or "DB" or "Displayed"	-1
-----------------------------------	------------------------------	-------------	---	----

Example: MMEMory:STORe:DATA "myData.cti","Citifile Formatted Data","DISPLAYED","MA",-1

<a href="#">*.csv</a>	"CSV Formatted Data"	"Trace" or "Auto"	"RI" or "MA" or "DB" or "Displayed"	Measurement number
-----------------------	-------------------------	-------------------	---	-----------------------

Example: MMEMory:STORe:DATA "myData.csv","CSV Formatted Data","Trace","DB",3

<a href="#">*.csv</a>	"CSV Formatted Data"	"Displayed"	"RI" or "MA" or "DB"	-1
-----------------------	-------------------------	-------------	-------------------------	----

Example: MMEMory:STORe:DATA "myData.csv","CSV Formatted Data","displayed","RI",-1

<a href="#">*.mdf</a>	"MDIF Data"	"Trace" or "Auto"	"RI" or "Displayed"	Measurement number
-----------------------	-------------	-------------------	------------------------	-----------------------

Example: MMEMory:STORe:DATA "myData.mdf","MDIF Data","trace","displayed",1

<a href="#">*.mdf</a>	"MDIF Data"	"Displayed"	"RI" or "Displayed"	-1
-----------------------	-------------	-------------	------------------------	----

Example: MMEMory:STORe:DATA "myData.mdf","MDIF Data","displayed","displayed",-1

### Notes (for above file types)

Use [Calc:Par:MNUM?](#) to read the measurement number of the selected trace.

### Scope:

"Trace" - specified measurement number only.

"Displayed" - all displayed measurements.

"Auto" - for all Standard Meas Class (S-parameter) channels:

- When correction is OFF, saves the specified trace
- When correction is ON, saves all corrected parameters associated with the calibrated ports in the Cal Set.

"Auto" - for all other channels:

- When correction is OFF or ON, saves the specified trace

The following parameter combinations save \*.csv files in specific formats for GCA and Swept IMD classes:

	Parameters			
	<type> (String)	<scope> (String)	<format> (String)	<selector> (Numeric)
GCA channels ONLY:	"GCA Sweep Data"	"Auto"	"DB"	GCA channel number

[Learn more](#)

Example: MMEMory:STORe:DATA "myData.csv","GCA Sweep Data","Auto","db",1

Swept IMD channels ONLY:	"IMD Sweep Data"	"Auto"	"DB"	Swept IMD channel number
--------------------------	------------------	--------	------	--------------------------

[Learn more](#)

Example: MMEMory:STORe:DATA "myData.csv","IMD Sweep Data","Auto","db",1

**Query Syntax** Not applicable

**Default** Not applicable

## MMEMory:STORe:ENR CAL, <file>

(Write-only) Stores an **ENR** (Excess Noise Source) data ([Noise Figure App only](#))

### Parameters

<file> String - Name of any valid file that is not already in existence. See [Specifying Path Names](#)

### Examples

MMEM:STOR:ENR CAL, C:/data/calset/346C\_16500.enr

**Query Syntax** Not applicable

**Default** Not applicable

## MMEMory:STORe:TRACe:FORMat:CITifile <char> - Superseded

This command is replaced with [MMEMory:STORe:DATA](#).

(Read-Write) Specifies the format of subsequent citifile save statements.

### Parameters

- <char> Format in which the citifile will be saved with subsequent [MMEMory:STORe:CIT:FORMat](#) statements. Choose from:
- MA** - Linear Magnitude / degrees
  - DB** - Log Magnitude / degrees
  - RI** - Real / Imaginary
  - AUTO** - Format in which the trace is already displayed. If other than Log Mag, Linear Magnitude, or Real/Imag, then the format will be in Real/Imag.
  - DISP** - Displayed format.

**Examples** `MMEM:STOR:TRAC:FORM:CIT MA`

**Query Syntax** MMEMory:STORe:TRACe:FORMat:CITifile?

**Return Type** Character

**Default** Auto

---

## MMEMory:STORe:TRACe:CONTents:CITifile <char> - Superseded

This command is replaced with [MMEMory:STORe:DATA](#).

(Read-Write) Specifies the contents of subsequent citifile save statements. (See [Data Define Saves](#))

### Parameters

- <char> Choose from:
- SING** - Single trace
  - DISP** - All displayed traces
  - AUTO** - All displayed traces

**Examples** `MMEM:STOR:TRAC:CONT:CIT SING`

**Query Syntax** MMEMory:STORe:TRACe:CONTents?

**Return Type** Character

Default Auto

---

### MMEMory:STORe:TRACe:FORMat:SNP <char>

(Read-Write) Specifies the format of subsequent .s1p, .s2p, .s3p; s4p save statements. [Learn more](#).

To save SNP data, use [CALC:DATA:SNP:PORTs:SAVE](#)

#### Parameters

<char> Choose from:

**MA** - Linear Magnitude / degrees

**DB** - Log Magnitude / degrees

**RI** - Real / Imaginary

**AUTO** - data is output in currently selected trace format. If other than LogMag, LinMag, or Real/Imag, then output is in Real/Imag.

**Examples** `MMEM:STOR:TRAC:FORM:SNP MA`

**Query Syntax** MMEMory:STORe:TRACe:FORMat:SNP?

**Return Type** Character

Default Auto'

---

### MMEMory:TRANsfer <fileName>,<dataBlock>

(Read-Write) Transfers data between the PNA and an external controller. Other MMEM commands transfer data between the PNA application and the PNA hard drive. If <fileName> already exists, it will be overwritten. The file must be no larger than 20MB.

To read **trace data** from the PNA in block format, use [CALC:DATA](#).

#### Parameters

<fileName> String - File name. See [Specifying Path Names](#)

<dataBlock> [Block Data](#) - The contents of the file.

The data block is a block of binary data. Use the following syntax:

**#<num digits><byte count><data bytes><NL><END>**

where:



<num\_digits> specifies how many digits are contained in <byte\_count>

<byte\_count> specifies how many data bytes will follow in <data bytes>

**Example** [See example program](#)

**Query Syntax** MMEMemory:TRANsfer? <fileName>

Reads block data from the specified file location.

**Default** Not applicable

---

Last modified:

2-Mar-2010	Updated for csv and mdif formats (9.2)
30-Oct-2008	Fixed noise keywords for load and store.
30-Jul-2007	Added noise keywords to load and store.
9/12/06	MQ Store command has reference to PORTS Snp.

## Output Commands

---

Controls two output functions: RF power and Noise Source.

### OUTPut:

- | [MANual:NOISe\[:STATe\]](#)
- | [\[:STATe\]](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

## OUTPut:MANual:NOISe[:STATe] <bool>

(Read-Write) Sets and reads the noise source state ON or OFF. For [Noise Figure measurements](#).

### Parameters

<bool>    **ON (1)** - Noise source ON  
          **OFF (0)** - Noise source OFF

### Examples

```
OUTP:MAN:NOIS 0  
output:manual:noise:state 1
```

**Query Syntax**    OUTPut:MANual:NOISe[:STATe]?

**Return Type**    Boolean (1 = ON, 0 = OFF)

**Default**    OFF (0)

---

## OUTPut[:STATe] <ON | OFF>

(Read-Write) Turns RF power from the source ON or OFF.

[See note about source power state with instrument state save and recall.](#)

#### Parameters

<ON | OFF> **ON** (or 1) - turns RF power ON  
**OFF** (or 0) - turns RF power OFF

#### Examples

```
OUTP ON  
output:state off
```

**Query Syntax** OUTPut[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

#### Last Modified:

24-Feb-2009 Replaced True/False

22-Aug-2007 Added noise command

## Route Command

---

[Learn about Frequency Offset](#)

[SCPI Command Tree](#)

### ROUTe<cnum>:PATH:LOOP[:R1] <char>

(Read-Write) Throws internal switch to reference receiver when the specified channel is measured.

This feature is available on PNA models with [Option 081](#) (external reference switch) and ALL PNA-X models.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<char> Position of the switch. Choose from:

**INTernal** - bypass R1 Loop. Connects the port 1 source directly to the R1 receiver.

**EXTernal** - flow through R1 Loop. Allows direct access to the R1 receiver through the Reference 1 front-panel connectors.

#### Examples

```
ROUT:PATH:LOOP INT
route2:path:loop:r1 external
```

**Query Syntax** ROUTe<cnum>:PATH:LOOP:R1?

**Return Type** Character

**Default** INTernal

---

Last Modified:

30-Apr-2009    Added PNA-X

## Sense:AverAge Commands

---

Sets sweep-to-sweep averaging parameters. Averaging is a noise reduction technique that averages each data point over a user-specified number of sweeps. Averaging affects all of the measurements in the channel.

**SENSe:AVERage**

| **CLEar**

| **COUNt**

| **MODE**

| **[STATe]**

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example](#) using some of these commands.
- [Learn about Averaging](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### SENSe<num>:AVERage:CLEar

(Write-only) Clears and restarts averaging of the measurement data. Does NOT apply to point averaging.

#### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:AVER:CLE  
sense2:average:clear
```

**Query Syntax** Not applicable

**Default** Not applicable

---

### SENSe<num>:AVERage:COUNt <num>

**(Read-Write)** Sets the number of measurements to combine for an average. Must also set [SENS:AVER\[:STATe\] ON](#)

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Number of measurements to average. Choose any number between **1** and **65536** ( $2^{16}$ ).

**Examples**

```
SENS:AVER:COUN 999
sense2:average:count 73
```

**Query Syntax** SENSE<cnum>:AVERage:COUNT?

**Return Type** Numeric

**Default** 1

---

**SENSe<cnum>:AVERage:MODE <char>**

**(Read-Write)** Sets the type of averaging to perform: Point or Sweep.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Averaging Type. Choose from:
- POINT** - Averaging measurements are made on each data point before stepping to the next data point. (Not available on 'C' models).
- SWEEP** - Averaging measurements are made on subsequent sweeps until the required number of averaging sweeps are performed.

**Examples**

```
SENS:AVER:MODE POIN
sense2:average:mode sweep
```

**Query Syntax** SENSE<cnum>:AVERage:MODE?

**Return Type** Character

**Default** Sweep

---

**SENSe<cnum>:AVERage[:STATe] <ON | OFF>**

**(Read-Write)** Turns trace averaging ON or OFF.

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.  
<ON | OFF> **ON** (or 1) - turns averaging ON.  
**OFF** (or 0) - turns averaging OFF.

**Examples**

```
SENS:AVER ON  
sense2:average:state off
```

**Query Syntax** SENSE<cnum>:AVERage[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** Off

---

Last Modified:

June 8, 2009	Not available on 'C' models
13-Oct-2008	Added Point averaging

## Sense:Bandwidth Commands

---

**SENSe:BANDwidth:**  
**RESolution <num>**  
**TRACk <bool>**

### See Also

- [Example Programs](#)
- [Learn about IF Bandwidth](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### SENSe<cnum>:BANDwidth | BWIDth[:RESolution] <num>

(Read-Write) Sets the bandwidth of the digital IF filter to be used in the measurement. (Use either **Sense:Bandwidth** or **Sense:Bwidth**)

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> IF Bandwidth in Hz. The list of valid IF Bandwidths is different depending on the PNA model. [\(Click to see the lists.\)](#) If an invalid number is specified, the analyzer will round up to the closest valid number.
- This parameter supports MIN and MAX as arguments. [Learn more.](#)

#### Examples

```
SENS:BWID 1KHZ  
sense2:bandwidth:resolution 1000
```

**Query Syntax** SENSe<cnum>:BANDwidth | BWIDth[:RESolution]?

**Return Type** Numeric

**Default** See [Preset IFBW](#) for your PNA model.

---

### SENSe<cnum>:BANDwidth | BWIDth:TRACk <bool>



(**Read-Write**) Sets and returns the state of the [Reduce IF BW at Low Frequencies](#) feature.

(Use either **Sense:Bandwidth:Track** or **Sense:Bwidth:Track**).

#### Parameters

<cnun> Any existing channel number. If unspecified, value is set to 1

<bool> Choose from:

**ON** or **1** - Reduce IF BW at Low Frequencies is set ON

**OFF** or **0** - Reduce IF BW at Low Frequencies is set OFF

#### Examples

```
SENS:BWID:TRAC OFF  
sense2:bandwidth:track 1
```

**Query Syntax** SENSE<cnun>:BANDwidth | BWIDth:TRACk?

**Return Type** Boolean

**Default** ON

---

Last Modified:

15-Jan-2008 MIN and MAX added

# Sense:Class Command

SENSe:CLASs:  
NAME?

Click on a [blue](#) keyword to view the command details.

See Also

- [Learn about Measurement Class](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

## SENSe<num>:CLASs:NAME?

(Read-only) Returns the measurement class name of the specified channel. Use [Calc:Custom](#) commands to create measurements from classes other than standard S-Parameters. Use [Cal:Par:Define:Ext](#) to create standard measurements.

Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

Examples

```
SENS:CLAS:NAME?  
sense2:class:name?  
  
For a standard S-Parameter channel, returns...  
"Standard"
```

Default Not applicable

Last Modified:

12-May-2008    MX New topic

## Sense:Correction Commands

---

Performs and applies calibration and other error correction features.

To perform Guided Calibration, see [Sens:Corr Coll:GUIDed](#).

### **SENSe:CORRection**

#### **CCHeck**

| [\[ACQuire\]](#)

| [DONE](#)

| [PARAmeter](#)

#### **CKIT - More Commands**

#### **COLlect**

| [\[ACQuire\]](#)

| [APPLy](#)

| [CKIT - More Commands](#)

| [DISPlay:WINDow](#)

| [AOFF](#)

| [\[STATe\]](#)

| [GUIDed - More Commands](#)

| [ISOLation:](#)

| [AVER:INCRement](#)

| [ECAL\[:STATe\]](#)

| [NOISe](#)

| [ENR:ADAP:DEEMbed\[:STATe\]](#)

| [PSEN:ADAP:DEEMbed\[:STATe\]](#)

| [LO:PCAL\[:STATe\]](#)

| [METHod](#)

| [SAVE](#)

| [SESSion - More Commands](#)

| [SWEep:CHANnel](#)

| [AOFF](#)

| [\[STATe\]](#)

| [WINDow\[:STATe\]](#)

#### **CSET - More Commands**

#### **ENR:CALibration:TABLE**

| [DATA](#)

| [ID:DATA](#)

| [SERial:DATA](#)

#### **EXTension - More Commands**

#### **GCSetup**

| **POWer**  
| **SENSor:**  
| **CKIT**  
| **CONNector**  
**IMD - More Commands**  
**IMPedance:INPut**  
| **MAGNitude**  
**INTerpolate[:STATe]**  
**ISOLation[:STATe]**  
**PREference**  
| **CALibration**  
| **[FOM:]RANGe**  
| **CSET**  
| **SAVE**  
| **SAVUser**  
| **ECAL**  
| **ORientation**  
| **PMAP**  
| **SIMCal**  
| **TRIG:FREE**  
**RPOWer:OFFSet**  
| **[AMPLitude]**  
**RVELocity**  
| **COAX**  
**SFOward**  
| **[STATe]**  
**[STATe]**  
**TCOLd:USER:VALue**  
**TSTandards**  
| **[STATe]**  
**TYPE**  
| **CATalog?**

Click on a [blue](#) keyword to view the command details.

**Red** commands are superseded.

#### See Also

- [Example Programs](#)
- New [See Calibrating the PNA Using SCPI](#)
- [Learn about Measurement Calibration](#)
- [Synchronizing the PNA and Controller](#)

- [SCPI Command Tree](#)
- 

## **SENSe<cnum>:CORRection:CCHeck[:ACQuire] <mod>[,char]**

(Write-only) Reads the 'confidence data' associated with the specified ECal module and puts it into memory. The measurement is selected using [SENS:CORR:CCH:PAR](#). This command is compatible with \*OPC.

### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <mod> ECAL Module that contains the confidence data. Choose from:

**ECAL1**

..through..

**ECAL8**

- [char] Optional argument. Specifies which characterization within the ECal module that the confidence data will be read from.

**CHAR0** Factory characterization (data that was stored in the ECal module by Agilent). Default if not specified.

**CHAR1** User characterization #1

**CHAR2** User characterization #2

**CHAR3** User characterization #3

**CHAR4** User characterization #4

**CHAR5** User characterization #5

### **Examples**

```
SENS:CORR:CCHeck ECAL2
```

```
sense2:correction:ccheck:acquire ecal1,char1
```

**Query Syntax** Not applicable

**Default** Not applicable

---

## **SENSe<cnum>:CORRection:CCHeck:DONE**

(Write-only) Concludes the Confidence Check and sets the ECal module back into the idle state.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:CORR:CCH:DONE  
sense2:correction:ccheck:done
```

**Query Syntax** Not applicable

**Default** Not applicable

---

**SENSe<cnum>:CORRection:CCHeck:PARAmeter <Mname>**

(Read-Write) Specifies an existing measurement to be used for the Confidence Check.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<Mname> Name of the measurement you are selecting for the confidence check. The measurement must already exist.

**Examples**

```
SENS:CORR:CCH:PAR 'TEST'  
'selects the measurement "test" on channel 1 for the confidence  
check'  
  
sense2:correction:ccheck:parameter 'test'  
'selects the measurement "test" on channel 2 for the confidence  
check'
```

**Query Syntax** SENSe<cnum>:CORRection:CCHeck:PARAmeter?

Returns the name of the selected measurement on channel <cnum>.

**Return Type** String

**Default** Not applicable

---

**SENSe<cnum>:CORRection:COLLect[:ACQuire] <class>[,subclass][,sync]**

(Write-only) For UNGUIDED calibration, measures the specified standards from the selected calibration kit. The calibration kit is selected using the [Sense:Correction:Collect:CKIT](#) command.

For using two sets of standards, see [SENS:CORR:TST](#).

**Note:** Before using this command you must select two items:

1. Select a calibration method using [SENS:CORR:COLL:METH](#)
2. Select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<class> **Measures the standards associated with these class labels.** Choose from:

Label	Forward	Reverse
STAN1	S11A	S22A
STAN2	S11B	S22B
STAN3	S11C	S22C
STAN4	S21T	S12T
STAN5	Generic Isolation; not associated with calibration kit definition.	
ECAL1	ECAL modules	
through		
ECAL8		
RESPonse	Same as <a href="#">Normalize</a> selection in Unguided Cal. (subclass is ignored)	
POWER	Take a receiver power cal sweep and turn correction ON	
SLSET	Sets 'sliding load type', and increments the "number of slides" count. The total number of slides is critical to the correct calculation of the sliding load algorithm. See a <a href="#">sliding load cal example</a> .	
SLDONE	Computes the sliding load using a circle fit algorithm.	

[subclass] Optional argument. For mechanical calibration kits, choose from the following to specify the standard to be acquired from the [SENS:CORR:COLL:CKIT:ORDER](#) list. If not specified, subclass is set to **SST1**.

- SST1** First standard in the order list
- SST2** Second standard in the order list
- SST3** Third standard in the order list
- SST4** Fourth standard in the order list
- SST5** Fifth standard in the order list
- SST6** Sixth standard in the order list
- SST7** Seventh standard in the order list

If an ECAL module (1 through 8) is specified for <class>, choose one of the following for specifying which characterization within the ECal module will be used for the acquire. If not specified, the default is **CHAR0**.

- CHAR0** Factory characterization (data that was stored in the ECal module by Agilent)
- CHAR1** User characterization #1
- CHAR2** User characterization #2
- CHAR3** User characterization #3
- CHAR4** User characterization #4
- CHAR5** User characterization #5

[sync] Optional argument. Choose from:

**SYNChronous** - blocks SCPI commands during standard measurement (default behavior)

**ASYNchronous** - does NOT block SCPI commands during standard measurement.

[Learn more.](#)

## Examples

```
SENS:CORR:COLL STAN1

'If SENS:CORR:COLL:CKIT:ORDer2 5,3,7
was specified, the following command measures standard 3 (the
second in the order list)
sense1:correction:collect:acquire stan3,sst2

SENS:CORR:COLL ECAL4,ASYN; *OPC?

sense2:correction:collect:acquire ecal2,char1
```

**Query Syntax** Not applicable

**Default** Not applicable



## SENSe<num>:CORRection:COLLect:APPLy

(Write-only) Applies error terms to the measurement that is selected using [Calc:Par:Select](#).

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

**Note:** This command is only necessary if you need to modify error terms. If you do not need to modify error terms, [SENSe<num>:CORRection:COLLect:SAVE](#) calculates and then automatically applies error terms after you use [SENS:CORR:COLL:ACQuire](#) to measure cal standards.

### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

### Example

```
1. CALCulate2:PARameter:SElect S21_2 'select the measurement to
   apply terms to
2. SENSe2:CORRection:COLLect:METHod SPARSOLT 'set type of cal
   method.
3. CALCulate2:DATA? SCORR1 'download the error term of interest
4. 'Modify the error term here
5. CALCulate2:DATA SCORR1 'upload the error term of interest
6. SENSe2:CORRection:COLLect:APPLy 'applies the error terms to
   the measurement
```

Query Syntax Not applicable

Default Not applicable

---

## SENSe:CORRection:COLLect:DISPlay:WINDow:AOff

(Write-only) Clears the flags for windows to be shown during calibrations. To flag a window to be shown see [SENS:CORR:COLL:DISP:WIND](#).

### Examples

```
SENS:CORR:COLL:DISP:WIND:AOff
sense:correction:collect:display:window:aoff
```

[See an example using this command.](#)

Query Syntax Not Applicable

Default Not Applicable

---

## SENSe:CORRection:COLLect:DISPlay:WINDow<wNum>[:STATe] <bool>

**(Write-only)** Set the 'show' state of the window to be displayed during a calibration. [Learn more.](#)

When this command is sent, the specified window is 'flagged' to be shown during calibration. The flag is cleared when the window is closed. A Preset or Instrument State Recall also closes the window. If the same window number is reopened, this command must be sent again to show the window during a calibration. The flag is NOT saved with an instrument state.

Send this command for each additional window to show during a calibration.

#### Parameters

<wNum> Window number to show during a calibration. The calibration window will also be shown with this window.

The window must already be created.

Use [DISPlay:CATalog?](#) to read all existing window numbers.

<bool> Window state. Choose from:

**ON** (or 1) - Show the specified window during calibration.

**OFF** (or 0) - Do NOT show the specified window during calibration.

#### Examples

```
SENS:CORR:COLL:DISP:WIND1 1
```

```
sense:correction:collect:display:window2:state off
```

[See an example using this command.](#)

**Query Syntax** Not Applicable

**Default** OFF

---

**SENSe:CORRection:COLLect:ISOLation:AVERage:INCRement <num>**

**(Read-Write)** Specifies amount to increment (increase) the channel averaging factor during isolation measurement of the ECal module during an unguided ECal calibration.

**Note:** if the channel currently has averaging turned OFF and <num> is greater than 1, averaging will be turned ON only during the isolation measurements and with the averaging factor equal to <num>.

**Parameters**

<num> Incremental Averaging factor. The maximum averaging factor is 65536 ( $2^{16}$ ).

**Examples**

```
SENS:CORR:COLL:ISOL:AVER:INCR 16
```

```
sense:correction:collect:isolation:average:increment 0
```

**Query Syntax** SENSE:CORRection:COLLect:ISOLation:AVERage:INCRement?

**Return Type** Numeric

**Default** 8 - If this command is NOT sent, but [ECal isolation is measured](#), then averaging will be turned ON with factor set to 8 during the isolation measurement.

---

**SENSe<cnum>:CORRection:COLLect:ISOLation:ECAL[:STATe] <bool>**

**(Read-Write)** Specifies whether or not the isolation state of the ECal module will be measured as part of an unguided ECal calibration.

An unguided calibration is performed using the SENS:CORR:COLL:METH and SENS:CORR:COLL:ACQ commands.

**Note:** The inherent isolation of the PNA is better than that attained with this command. ONLY use this command when using an external test set, and ONLY using a 8509x ECal module.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<bool> **ON** (or 1) - isolation is measured during the unguided ECal calibration.

**OFF** (or 0) isolation is NOT measured during the unguided ECal calibration.

**Examples**

```
SENS1:CORR:COLL:ISOL:ECAL ON
```

```
sense2:correction:collect:isolation:ecal:state 0
```

**Query Syntax** SENSe:CORRection:COLLect:ISOLation:ECAL:STATe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

## SENSe<cnum>:CORRection:COLLect:METHod <char>

(Read-Write) For UNGUIDED calibration, sets the calibration method (also known as 'Calibration Type' on calibration dialog box.) To select a Cal Type from a Cal Set, use [CALC:CORR:TYPE](#).

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

Method	Description
NONE	No Cal method
REFL1OPEN	Response Open
REFL1SHORT or REFL1	Response Short
REFL3	Full 1 port
RESPonse	Same as <a href="#">Normalize</a> selection in Unguided Cal.
RPOWer	Receiver Power Cal - Used only with receiver measurements.
TRAN1	Response Thru - Requires a Thru standard.
TRAN2	Response Thru and Isolation - Requires a Thru standard.
SPARSOLT	Full SOLT 2 port
SPARTRL	TRL Cal ( <a href="#">not available on all PNAs</a> .)

### Examples

```
SENS:CORR:COLL:METH REFL1
sense2:correction:collect:method sparsolt
```

**Query Syntax** SENSe<cnum>:CORRection:COLLect:METHod?

**Return Type** Character

**Default** Not Applicable

---

## SENSe<ch>:CORRection:COLLect:NOISe:ENR:ADAPter:DEEMbed:[STATe] <bool>

**(Read-Write)** Set and read the state of ENR Adapter de-embedding. [Learn more.](#)

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> ENR Adapter de-embed state. Choose from:
- OFF** or **0** - Do not force de-embedding.
- ON** or **1** - Force de-embedding.

**Examples**

```
SENS:CORR:COLL:NOIS:ENR:ADAP:DEEM 0
```

```
sense2:correction:collect:noise:enr:adapter:deembed:state ON
```

**Query Syntax** SENSE:CORRection:COLLect:NOISe:ENR:ADAPter:DEEMbed:[STATe]?

**Return Type** Boolean

**Default** 0 - OFF

---

**SENSe<ch>:CORRection:COLLect:NOISe:PSENsor:ADAPter:DEEMbed:[STATe] <bool>**

**(Read-Write)** Set and read the state of power sensor adapter de-embedding. [Learn more.](#)

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> Power sensor adapter de-embed state. Choose from:
- OFF** or **0** - Do not force de-embedding.
- ON** or **1** - Force de-embedding.

**Examples**

```
SENS:CORR:COLL:NOIS:PSEN:ADAP:DEEM 0
```

```
sense2:correction:collect:noise:psensor:adapter:deembed:state ON
```

**Query Syntax** SENSE:CORRection:COLLect:NOISe:PSENsor:ADAPter:DEEMbed:[STATe]?

**Return Type** Boolean

**Default** 0 - OFF

---

**SENSe<ch>:CORRection:COLLect:NOISe:LO<n>:PCAL[:STATe] <bool>**

**(Read-Write)** Enables and disables LO power calibration for NFX.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> LO Stage (number). Choose 1 for NFX.
- <bool> LO Power Cal state. Choose from:
- OFF** or **0** - Disable LO Power Cal
- ON** or **1** - Enable LO Power Cal

**Examples**

```
SENS:CORR:COLL:NOIS:LO1:PCAL 0
```

```
sense2:correction:collect:noise:lo1:pcal:state ON
```

**Query Syntax** SENSE:CORRection:COLLect:NOISe:LO<n>:PCAL:STATe?

**Return Type** Boolean

**Default** 0 - OFF

---

## SENSe<cnum>:CORRection:COLLect:SAVE

**(Write-only)** For UNGUIDED calibrations ONLY. This command does the following:

- calculates the error terms using the selected :METHod
- applies the error terms to the selected measurement (turns error correction ON.)
- saves the calibration error-terms to the channels Cal Register or a User Cal Set.

The Cal Register or User Cal Set is determined by the setting of the SENS:CORR:PREFeRence:CSET:SAVE command.

Do NOT use this command during an ECAL. When performing an ECAL calibration using [SENS:CORR:COLL:ACQuire](#), this SAVE operation is performed automatically before the completion of a successful ACQuire.

Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:CORR:COLL:SAVE  
sense2:correction:collect:save
```

**Query Syntax**

Not applicable

**Default**

Not applicable

---

**SENSe:CORRection:COLLect:SWEep:CHANnel:AOff**

(Write-only) Clears ALL flags for channels to sweep during calibration. To flag a channel, see [SENS:CORR:COLL:SWE:CHAN](#).

**Examples**

```
SENS:CORR:COLL:SWE:CHAN:AOff  
  
sense:correction:collect:sweep:channel:aoff  
  
See an example using this command.
```

**Default**

Not applicable

---

**SENSe<cnum>:CORRection:COLLect:SWEep:CHANnel<cnum2>[:STATe] <bool>**

(Write-only) Specifies the channel to sweep during a Calibration.

When this command is sent, the <cnum2> channel is 'flagged' to be swept during calibration.

The flag is cleared when the channel is deleted, if the Measurement Class is changed, or if all measurements are deleted from the channel.

If the same channel number is recreated, this command must be sent again to sweep the channel during a calibration. The flag is NOT saved with an instrument state.

A Preset or Instrument State Recall deletes the channel.

**Parameters**

<cnum> The channel to be calibrated. If unspecified, value is set to 1.

<cnum2> The channel to sweep when waiting to measure a standard.

This channel must already exist with at least one measurement in the channel. If this channel is in continuous sweep mode, it must have the same attenuator settings and path configuration (PNA-X only).

<bool> Channel sweep state. Choose from:

**ON** (or 1) - Sweep the channel during calibration.

**OFF** (or 0) - Do NOT sweep the channel during calibration.

**Examples**

```
SENS:CORR:COLL:SWE:CHAN2 1
```

```
sense2:correction:collect:sweep:channel3:state off
```

[See an example using this command.](#)

**Query Syntax** Not Applicable

**Default** OFF

---

**SENSe:CORRection:ENR:CALibration:TABLE:DATA <freq, value, freq, value...>**

**(Read-Write)** Set and read the ENR calibration data. All of the frequency and ENR data must be sent at the same time. Use [MMEM:LOAD](#) and [MMEM:SAVE](#) to load and save ENR table data from disk. [Learn more about Noise Source ENR files.](#)

**Parameters**

<freq, value> (Numeric) ENR data. Frequency value in Hz followed by a ENR noise value in dB. Enter as many pairs as necessary.

**Examples**

```
SENS:CORR:ENR:CAL:TABL:DATA 1.0E9,14.37,2.5E9,15.28
```

```
sense:correction:enr:calibration:table:data
```

```
1.0E9,14.37,2.5E9,15.28
```

**Query Syntax** SENSe:CORRection:ENR:CALibration:TABLE:DATA?

**Return Type** Comma separated numeric values

**Default** Not Applicable

---

**SENSe:CORRection:ENR:CALibration:TABLE:ID:DATA <id>**



**(Read-Write)** Sets and returns ID of ENR table. While this is for informational purposes only, it can be used to record the model of the noise source. [Learn more about ENR files.](#)

**Parameters**

<id> (String) Identifier for the ENR table.

**Examples**

```
SENS:CORR:ENR:CAL:TABL:ID:DATA "346C"  
sense:correction:enr:calibration:table:id:data "ENR Table"
```

**Query Syntax** SENSE:CORRection:ENR:CALibration:TABLE:ID:DATA?

**Return Type** String

**Default** Not Applicable

---

**SENSe:CORRection:ENR:CALibration:TABLE:SERIal:DATA <sn>**

**(Read-Write)** Sets and returns the serial number of noise source. This is for informational purposes only to identify the specific noise source for which the data pertains. [Learn more about ENR files.](#)

**Parameters**

<sn> Serial number of the noise source for which the data applies, enclosed in quotes.

**Examples**

```
SENS:CORR:ENR:CAL:TABL:SER:DATA "ABCD1234"  
sense:correction:enr:calibration:table:serial:data "ABCD1234"
```

**Query Syntax** SENSE:CORRection:ENR:CALibration:TABLE:SERIal:DATA?

**Return Type** String

**Default** Not Applicable

---

**SENSe<ch>:CORRection:GCSetup:POWer <num>**

**(Read-Write)** Set and read the power level at which to perform the Source Power Cal portion of a Gain Compression (Opt 086) Calibration. [Learn more about this setting.](#)

**Parameters**

<num> Power level in dB. Choose a value from +30 to (-30).

**Examples**

```
SENS:CORR:GCS:POW 0  
sense:correction:gcsetup:power 5
```

**Query Syntax** SENSE:CORRection:GCSetup:POWer?

**Return Type** Numeric

**Default** 0

---

### SENSe<ch>:CORRection:GCSetup:SENSor:CKIT <string>

(Read-Write) Set and read the cal kit to be used for calibrating at the port 1 reference plane when the power sensor connector is different from the DUT port 1. [Learn more.](#)

#### Parameters

<string> Cal Kit. Use [SENS:CORR:COLL:GUID:CKIT:PORT1:CAT?](#) to return a list of valid cal kits.

#### Examples

```
SENS:CORR:GCS:SENS:CKIT "85052B"
```

**Query Syntax** SENSe:CORRection:GCSetup:SENSor:CKIT?

**Return Type** String

**Default** Not Applicable

---

### SENSe<ch>:CORRection:GCSetup:SENSor:CONNector<string>

(Read-Write) Set and read the power sensor connector type which is used to perform the Source Power Cal portion of a Gain Compression Calibration. [Learn more.](#)

#### Parameters

<string> Power sensor connector type. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to return a list of valid connector types.

Select "Ignored" to NOT compensate for the adapter.

#### Examples

```
SENS:CORR:GCS:SENS:CKIT "3.5 mm (50) male"
```

**Query Syntax** SENSe:CORRection:GCSetup:SENSor:CKIT?

**Return Type** String

**Default** Not Applicable

---

### SENSe:CORRection:IMPedance:INPut:MAGNitude <num>

**(Read-Write)** Sets and returns the system impedance value for the analyzer.

**Parameters**

<num> System Impedance value in ohms. Choose any number between 0 and 1000 ohms.

**Examples**

```
SENS:CORR:IMP:INP:MAGN 75
sense:correction:impedance:input:magnitude 50.5
```

**Query Syntax** SENSE:CORRection:IMPedance:INPut:MAGNitude?

**Return Type** Numeric

**Default** 50

---

**SENSe<cnum>:CORRection:INTerpolate[:STATe] <ON | OFF>**

**(Read-Write)** Turns correction interpolation ON or OFF.

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns interpolation ON.  
**OFF** (or 0) - turns interpolation OFF.

**Examples**

```
SENS:CORR:INT ON
sense2:correction:interpolate:state off
```

**Query Syntax** SENSe<cnum>:CORRection:INTerpolate[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**SENSe<cnum>:CORRection:ISOLation[:STATe] <ON | OFF> OBSOLETE**

This command no longer works beginning in the PNA 5.2 release. To perform isolation as part of an unguided calibration, you must explicitly measure the isolation standard using [SENS:CORR:COLL:ACQ Stan5](#). See Example program.

To measure isolation as part of an ECal, use [SENS:CORR:COLL:ISOL:ECAL](#).

**(Read-Write)** Turns isolation cal ON or OFF during Full 2-port calibration. If this command is not sent, the default state is to **disable** Isolation.

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns isolation ON.  
**OFF** (or 0) - turns isolation OFF.

#### Examples

```
SENS:CORR:ISOL ON
sense2:correction:isolation:state off
```

**Query Syntax** SENSE<num>:CORRection:ISOLation[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF - (Isolation disabled)

---

### SENSe:CORRection:PREFeRence:CALibration[:FOM]:RANGe <char>

**(Read-Write)** Specifies the FOM frequency range to use when performing calibration.

#### Parameters

<char> Choose from:

**PRIMary** - Used for calibrating at the mmWave frequencies when NOT using a test set. [Learn more.](#)

**AUTO** - All other calibration situations.

#### Examples

```
SENS:CORR:PREF:CAL:RANG PRIM
sense:correction:preference:calibration:fom:range auto
```

**Query Syntax** SENSe:CORRection:PREFeRence:CALibration[:FOM]:RANGe?

**Return Type** Character

**Default** AUTO

---

### SENSe:CORRection:PREFeRence:CSET:SAVE <char>

## Important Notes:

- This command replaces [SENS:CORR:PREF:CSET:SAVU](#)
- With 6.0 we implemented a change that defaults to saving completed calibrations to Cal Registers instead of User Cal Sets. To revert to the old behavior, send this command with the USER argument.

**(Read-Write)** Specifies the default manner in which calibrations that are performed using SCPI or COM are to be stored. Cal data is ALWAYS stored to the channel Cal Register regardless of this setting.

This setting survives instrument preset and reboot. It remains until changed by another execution of this command.

**Note:** Cal Set arguments used with commands such as [SENS:CORR:COLL:GUID:INIT](#), [SENS:CORR:COLL:GUID:SAVE](#) and [SENS:CORR:COLL:GUID:SAVE:CSET](#) will override any of these default preference settings.

Learn about [Cal Registers and User Cal Sets](#).

### Parameters

<char> **CALRegister** - Cal is saved ONLY to the channel Cal Register.

**USER** - Cal is saved to a new User Cal Set file when performing a SCPI calibration. The Cal Set name is automatically generated. To change the name, send [SENS:CORR:CSET:NAME](#) after the cal is complete. This reverts to pre-6.0 behavior.

**REUSE** - The cal is saved to the Cal Set that is currently selected on the specified channel, which could be the channel Cal Register. If the channel does not yet have a selected Cal Set, the cal will be saved to a new User Cal Set with an automatically-generated name.

### Examples

```
SENS:CORR:PREF:CSET:SAVE USER
sense:correction:preference:cset:save reuse
```

**Query Syntax** SENSE:CORRection:PREFerence:CSET:SAVE?

**Return Type** Character

**Default** CALRegister

---

**SENSe:CORRection:PREFerence:CSET:SAVUser <bool> Superseded**

This command is replace with [SENS:CORR:PREF:CSET:SAVE](#)

**NOTE:** With 6.0 we implemented a change that defaults to saving completed calibrations to Cal Registers instead of User Cal Sets. To revert to the old behavior, send this command as ON (1). For UI and COM use, this can be done from the [GPIB console](#).

**(Read-Write)** Specifies whether cal data is automatically saved to a User Cal Set file after performing a SCPI calibration. Cal data is always saved to a Cal Register regardless of this setting.

This setting survives instrument preset and reboot. It remains until changed by another execution of this command.

Learn about [Cal Registers and User Cal Sets](#).

#### Parameters

<bool> **ON** or **1** - Cal is automatically saved to a User Cal Set file when performing a SCPI calibration. The Cal Set name is automatically generated. To change the name, send [SENS:CORR:CSET:NAME](#) after the cal is complete. Reverts to pre-6.0 behavior.

**OFF** or **0** - Cal is NOT automatically saved to a User Cal Set. To save a calibration to a User Cal Set, use [SENS:CORR:COLL:GUID:INIT](#).

#### Examples

```
SENS:CORR:PREF:CSET:SAVU 1  
sense:correction:preference:cset:savuser 0
```

**Query Syntax** SENSE:CORRection:PREFeRence:CSET:SAVUser?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF (0)

---

**SENSe:CORRection:PREFeRence:ECAL:ORlentation[:STATe] <ON|OFF>**

**(Read-Write)** Specifies whether or not the PNA should perform orientation of the ECal module during calibration. Orientation is a technique by which the PNA automatically determines which ports of the module are connected to which ports of the PNA. Orientation begins to fail at very low power levels or if there is much attenuation in the path between the PNA and the ECal module. If orientation is turned OFF, the [SENS:CORR:PREF:ECAL:PMAP](#) command must be used to specify the port connections before performing a cal.

**Note:** For 3-port or 4-port measurements, when orientation is OFF, you are not allowed to specify how the ECAL module is connected. Instead, the PNA determines the orientation. Use [SENS:CORR:COLL:GUID:DESC?](#) to query the orientation. The PNA does not verify that you made the connection properly.

This setting remains until the PNA is restarted or this command is sent again.

**Parameters**

<bool> ECAL orientation state. Choose from:

**ON** or **1** - PNA performs orientation of the ECal module.

**OFF** or **0** - PNA does NOT performs orientation of the ECal module.

**Examples**

```
SENS:CORR:PREF:ECAL:ORI OFF
```

```
sense:correction:preference:ecal:orientation:state on
```

**Query Syntax** SENSE:CORRection:PREFeRence:ECAL:ORientation[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON (1)

---

**SENSe:CORRection:PREFeRence:ECAL:PMAP <module>,<string>**

**(Read-Write)** When ECal module orientation is turned OFF ([SENS:CORR:PREF:ECAL:ORI](#)), this command specifies the port mapping (which ports of the module are connected to which ports of the PNA) prior to performing ECal calibrations.

This setting remains until the PNA is restarted or this command is sent again.

**Parameters**

<module> Specifies which ECal module this port map is being applied to. Choose from:

**ECAL1**

.through.

**ECAL8**

<string> Format this parameter in the following manner:

Aw,Bx,Cy,Dz

where

- A, B, C, and D are literal ports on the ECAL module
- w,x,y, and z are substituted for PNA port numbers to which the ECAL module port is connected.

Ports of the module which are not used are omitted from the string.

For example, on a 4-port ECal module with

port A connected to PNA port 2

port B connected to PNA port 3

port C not connected

port D connected to PNA port 1

the string would be: A2,B3,D1

If either the receive port or source port (or load port for 2-port cal) of the CALC:PAR:SElected measurement is not in this string and orientation is OFF, an attempt to perform an ECal calibration will fail.

#### Examples

```
SENS:CORR:PREF:ECAL:PMAP ECAL2, 'A1,B2'  
sense:correction:preference:ecal:pmap ecal3, 'a2,b1,c3'
```

**Query Syntax** SENSE:CORRection:PREFeRence:ECAL:PMAP? <module>

**Return Type** String

**Default** Null string ()

---

**SENSe:CORRection:PREFeRence:SIMCal <bool>**



**(Read-Write)** Sets and returns a preference for the Unguided Cal behavior described below. This setting persists until it is changed.

This preference can also be set ON by executing the script on the PNA at C:/Program Files/Agilent/Network Analyzer/System/wincal32.reg.

#### Parameters

<bool> Boolean - Choose from:

**0 - OFF** - Reverts to new (preferred) behavior. An error is returned if standard data is not acquired before sending [SENS:CORR:COLL:SAVE](#).

**1 - ON** - (WinCal compatible) Prevents [SENS:CORR:COLL:SAVE](#) from failing when standard data has not, and will not, be acquired.

[Learn more about old and new behaviors.](#)

#### Examples

```
SENS:CORR:PREF:SIMC 0
```

```
sense:correction:preference:simcal 1
```

**Query Syntax** SENSE:CORRection:PREFeRence:SIMCal?

**Return Type** Boolean

**Default** 0

---

### SENSe:CORRection:PREFeRence:TRIG:FREE <char>, <bool>

**(Read-Write)** Sets and returns the preference for the trigger behavior during a calibration. This setting persists until it is changed.

**Note:** If [TRIGger:SOURce](#) = Manual, during a calibration the PNA ALWAYS switches to Internal for one trigger, then back to Manual, regardless of this preference command.

#### Parameters

<char> Character - Calibration type. Choose from:

**GUIDed** - preference setting pertains to a Guided calibration.

**UNGuided** - preference setting pertains to an Unguided calibration.

<bool> Boolean - Choose from:

**0 - OFF** - The trigger behavior during the specified calibration type DOES respect the setting of the [TRIGger:SOURce](#) command. For example, when Trigger source = External, the single trigger method will wait for the External

trigger signal and then allow only one sweep.

**1 - ON** - (Pre-6.0 behavior) The trigger behavior during the specified calibration type does NOT respect the setting of the [TRIGger:SOURce](#) command. For example, when Trigger source = External, during calibration the PNA switches to Internal sweep, responds to one trigger signal to measure the standard, then switches back to External.

**Examples**

```
SENS:CORR:PREF:TRIG:FREE GUID,1
```

```
sense:correction:preference:trig:free unguided,0
```

**Query Syntax**

SENSe:CORRection:PREFeRence:TRIG:FREE? <char>

**Return Type**

Boolean

**Default**

OFF for both calibration types.

---

**SENSe<cnum>:CORRection:RPOWer:OFFSet[:AMPLitude] <num>**

(Read-Write) Adjusts a receiver power cal to account for components or adapters that are added between the source port and receiver while performing this cal. For more information, see [Receiver Cal.](#)

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Offset Value in dB. Specify loss as a negative number; and gain as a positive number. Choose a number between -200 and 200.

**Examples**

```
SENS:CORR:RPOW:OFFS .5
```

```
sense2:correction:rpower:offset:amplitude -.5
```

**Query Syntax**

SENSe<cnum>:CORRection:RPOWer:OFFSet[:AMPLitude]?

**Return Type**

Numeric

**Default**

0

---

**SENSe<cnum>:CORRection:RVELocity:COAX <num>**

**(Read-Write)** Sets the velocity factor to be used with Electrical Delay and Port Extensions.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<num> Velocity factor. Choose a number between **0** and **10**

(.66 polyethylene dielectric; .7 teflon dielectric)

**Examples**

```
SENS:CORR:RVEL:COAX .66
sense2:correction:rvelocity:coax .70
```

**Query Syntax** SENSE<num>:CORRection:RVELocity:COAX?

**Return Type** Numeric

**Default** 1

---

**SENSe<num>:CORRection:SFORward[:STATe] <boolean>**

**(Read-Write)** Sets the direction a calibration will be performed when only one set of standards is used.

Use [SENSe:CORRection:TSTandards\[:STATe\]](#) **OFF** to specify that only one set of standards will be used.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<boolean> **ON (1)** - FORWARD direction of a 2-port calibration will be performed

**OFF (0)** - REVERSE direction of a 2-port calibration will be performed

**Examples**

```
SENS:CORR:SFOR 1
sense2:correction:sforward:state 0
```

See an [example](#) using this command

**Query Syntax** SENSE<num>:CORRection:SFORward[:STATe]?

**Return Type** Boolean

**Default** ON

---

**SENSe<num>:CORRection[:STATe] <ON | OFF>**

**(Read-Write)** Turns error correction ON and OFF for the specified channel.

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - correction is applied to the channel.
- OFF** (or 0) - correction is NOT applied to the channel.

**Examples**

```
SENS:CORR ON
sense2:correction:state off
```

**Query Syntax** SENSE<num>:CORRection[:STATe]?

To query the error correction state for a measurement, use [CALC:CORR:STATe?](#)

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**SENSe<num>:CORRection:TCOLd:USER:VALue <num>**

**(Read-Write)** Sets and returns the temperature of the noise source connector. Learn more about [Noise Figure Calibration](#).

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1.
- <num> Noise source temperature in Kelvin.

**Examples**

```
SENS:CORR:TCOL 295
sense2:correction:tcold 298
See an example using this command
```

**Query Syntax** SENSE<num>:CORRection:TCOLd:USER:VALue?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<num>:CORRection:TStandards[:STATe] <boolean>**

**(Read-Write)** Specifies the acquisition of calibration data using ONE or TWO sets of standards.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <boolean> **ON (1)** - TWO identical sets of standards will be used to simultaneously calibrate two ports (for both Forward and Reverse parameters).
- OFF (0)**- ONE set of standards will be used to perform a full 2-port calibration, one port at a time.

When specifying ON (use two sets), the [SENS:CORR:COLL:ACQuire](#) command uses the same standard index for each calibration class. To specify the calibration standard gender for each port, you must first ensure that the order of calibration class accurately reflects the configuration of your DUT.

For example, for a DUT with a male connector on port 1 and a female connector on port 2, order the devices within the S11 classes (A, B, and C) such that the MALE standards are first in the list. Then order the S22 classes specifying the FEMALE standards as the first in the list.

**Examples**

```
SENS:CORR:TST 1
sense2:correction:tstandard:state 0
```

See an [example](#) using this command

**Query Syntax** SENSE<cnum>:CORRection:TSTandards[:STATe]?

**Return Type** Boolean

**Default** ON

---

**SENSe:CORRection:TYPE:CATalog? <char>**

(Read-Write) Lists the Cal Types in the PNA by either GUID or registered name. [Learn more about applying Cal Type using SCPI.](#)

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

#### Parameters

<char> Specifies the type of list. Choose from:

**GUID** - the registered GUID of the Cal Type

**NAME** - the registered name of the Cal Type

#### Examples

```
SENS:CORR:TYPE:CAT? GUID
```

**Query Syntax** SENSE<cnum>:CORRection:TYPE:CATalog? <char>

**Return Type** Comma-separated string

**Default** Not Applicable

---

Last modified:

26-Oct-2009 Added Noise commands and Normalization method. (9.1)

16-Sep-2009 Added Cal Pref FOM

13-Feb-2009 Added Acq, sync

23-Jan-2009 Added 'if meas deleted' to Swe:Chan

6-Mar-2008 Added Noise TCOLd

19-Sep-2007 Added missing <cnum> arguments

July 30, 2007 Added ENR commands

April 14, 2007 Addd ECal isolation commands

Oct 30, 2006 Modified SavUser command

## Sense:Correction:CKIT Commands

---

Manages the list of cal kits that are installed in the PNA.

### SENSe:CORR:CKIT

[CLEar](#)

[COUNT?](#)

**ECAL**

| [CHARacterize More commands](#)

| [CLISt?](#)

| **DMEMory**

| [CLEar](#)

| [IMPort](#)

| [EXPort](#)

| **INFormation**

| **KNAME**

| [INFormation](#)

| [LIST?](#)

| [ORient?](#)

| **PATH**

| [COUNT?](#)

| [DATA?](#)

[EXPort](#)

[IMPort](#)

[INITialize](#)

[LOAD](#)

- Click on a [blue](#) keyword to view the command details.
- Red is a superseded command

- New [See Calibrating the PNA Using SCPI](#)
- [Learn about Modifying Cal Kits](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## SENSe:CORRection:CKIT:CLEar[:IMMediate] [ckit]

(Write-only) Deletes installed cal kits.

### Parameters

[ckit] Optional String. Cal Kit to delete. If not specified, all PNA Cal kits are deleted, including custom kits.

### Examples

```
SENS:CORR:CKIT:CLE
```

```
sense:correction:ckit:clear:immediate "85052B"
```

Query Syntax Not Applicable

Default Not Applicable

---

## SENSe:CORRection:CKIT:COUNt?

(Read-only) Returns the number of installed cal kits.

### Examples

```
SENS:CORR:CKIT:COUNt?
```

Query Syntax SENS:CORR:CKIT:COUNt?

Return Type Numeric

Default Not Applicable

---

## SENSe:CORRection:CKIT:ECAL<mod>:CLISt?



**(Read-only)** Returns a list of characterizations stored in the specified ECal module.

**Parameters**

<mod> ECal module from which to read user characterization numbers. If unspecified, value is set to 1.

**Examples**

Module 1 contains User Characterizations 1 and 3.

```
SENSe:CORRection:CKIT:ECAL:CLIST?
```

'Returns the following (0 always indicates the factory characterization):

0,1,3

**Return Type** Numeric list, separated by commas.

**Default** Not Applicable

---

**SENSe:CORRection:CKIT:ECAL:DMEMory:CLEar <kitName>**

**(Write-only)** Deletes user characterizations from PNA disk memory.

**Parameters**

<kitName> Optional String argument. ECal Model, User Characterization name + " ECal", and serial number of the ECal module, separated by spaces. See examples below.

If unspecified, ALL User Characterizations that are stored in PNA disk memory are deleted.

**Examples**

'These examples all use "MyUserChar" as the User characterization name.

'The "My User Char" characterization is deleted from disk memory.

```
SENS:CORR:CKIT:ECAL:DMEM:CLE "N4433A MyUserChar ECal 00001"
```

'All User characterizations are deleted from disk memory.

```
SENS:CORR:CKIT:ECAL:DMEM:CLE
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe:CORRection:CKIT:ECAL:DMEMory:IMPort <file>**

(Write-only) After the PNA disk memory is [Exported](#) to a file, use this command to Import the file into PNA disk memory, which allows the User Characterization to be used with the PNA and ECal module.

#### Parameters

<file> String. Full path and file name of file that was exported.

#### Examples

```
SENS:CORR:CKIT:ECAL:DMEM:IMP C:/Program Files/Agilent/Network Analyzer/ECal User Characterizations/myDiskUserChar.euc"
```

Query Syntax Not Applicable

Default Not Applicable

---

### SENSe:CORRection:CKIT:ECAL:EXPort <kit>[,<file>][,<NewName>]]

(Write-only) Saves an existing ECal characterization to a file. Use this command to archive the user characterization or to move the characterization to a different PNA for use with the specified ECal module. After exporting the user characterization, use [SENS:CORR:CKIT:ECAL:DMEM:IMPort](#) to make the user characterization available for use.

#### Parameters

<kit> String. Not case sensitive. ECal Model, User char name + " ECal", and serial number of the ECal module used for the characterization, separated by spaces. See examples below.

If the model and serial number of the module is not found, an error is returned.

<file> Optional String argument. Path and filename of the user characterization. If not specified, the file is saved using characterization name + ".euc". If the path is not specified, it is stored in "C:/Program Files/Agilent/Network Analyzer/ECal User Characterizations/". The extension ".euc" is appended if one is not specified.

<NewName> Optional String argument. This allows you to change the name for the User Characterization. When specified, the new name is saved in the file with the characterization. If unspecified, the existing user characterization name is saved.

#### Examples

```
'These examples all use "MyUserChar" as the User characterization name.
```

```
'All parameters specified
```

```
SENS:CORR:CKIT:ECAL:EXP "N4433A MyUserChar ECal 00001", "myUserChar.euc", "NewUserChar"
```

```
'First two parameters are specified
sense:correction:ckit:ecal:export "N4691B MyUserChar ECal 00500"
myUserChar.euc"

'Only first parameter is specified
SENS:CORR:CKIT:ECAL:EXP "N4433A MyUserChar ECal 00001"

'Only <file> is NOT specified
SENS:CORR:CKIT:ECAL:EXP "N4433A MyUserChar ECal 00028", "",
"NewUserChar"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

## **SENSe:CORRection:CKIT:ECAL<mod>:INFormation? [<char>] Superseded**

**(Read-only)** This command is replaced with [SENS:CORR:CKIT:ECAL:KNAM:INF?](#)

Reads the user-characterization information from the specified ECal module. This command returns the same values as [SENS:CORR:COLL:CKIT:INF?](#)

### **Parameters**

<mod> ECal module to read characterizations from. Choose from:

1 through 8. If unspecified, value is set to 1.

<char> Optional argument. Specifies which characterization within the ECal module to read information from. If not specified, value is set to CHAR0.

Choose from:

- CHAR0 Factory characterization (data that was stored in the ECal module by Agilent)
- CHAR1 User characterization #1
- CHAR2 User characterization #2
- CHAR3 User characterization #3
- CHAR4 User characterization #4
- CHAR5 User characterization #5

**Examples** `SENS:CORR:CKIT:ECAL2:INFormation? char5`

**'Example return string:**

```
ModelNumber: 85092-60007, SerialNumber: 01386, ConnectorType:
N5FN5F, PortAConnector: Type N (50) female, PortBConnector: Type
N (50) female, MinFreq: 30000, MaxFreq: 9100000000,
NumberOfPoints: 250, Calibrated: July 4 2002
```

**Return Type** Character

**Default** Not Applicable

---

## **SENSe:CORRection:CKIT:ECAL:KNAME:INFormation? <kitName>**

(Read-only) This command replaces [SENS:CORR:COLL:CKIT:INF?](#)

Reads the user-characterization information from the specified ECal module or PNA disk memory.

[Learn more about User Characterization in PNA Disk Memory.](#)

### **Parameters**

<kitName> String. ECal kit name and user characterization to read information from.

Optionally include the serial number in the <kit name>. Include the serial number when two or more ECal modules with same model number are attached to the PNA, or the PNA may not be referencing the intended ECal module.

### **Examples**

'For a factory characterization in module memory:

```
SENS:CORR:CKIT:ECAL:KNAM:INF? 'N4433A ECal 00028'
```

'For user characterization in module memory:

```
SENS:CORR:CKIT:ECAL:KNAM:INF? 'N4433A User 1 ECal 00028'
```

'For user characterization in disk memory:

```
SENS:CORR:CKIT:ECAL:KNAM:INF? 'N4433A foo ECal 00028' "
```

'Example return string:

```
ModelNumber: 85092-60007, SerialNumber: 01386, ConnectorType:
N5FN5F, PortAConnector: Type N (50) female, PortBConnector: Type
N (50) female, MinFreq: 30000, MaxFreq: 9100000000,
NumberOfPoints: 250, Calibrated: July 4 2002
```

**Return Type** Character

**Default** Not Applicable

---

## **SENSe:CORRection:CKIT:ECAL:LIST?**

**(Read-only)** Returns a list of index numbers to be used for referring to the ECal modules that are currently attached to the PNA.

**Examples**

```
SENS:CORR:CKIT:ECAL:LIST?  
  
'If 2 modules are attached to the PNA  
'then the returned list will be:  
  
1, 2
```

**Return Type** Numeric list, separated by commas.

**Default** Not Applicable

---

**SENSe<ch>:CORRection:CKIT:ECAL<n>:ORient? <pnaPort>[,<charN>]**

**(Read-only)** Returns the ECal port that is connected to the specified PNA port. A calibration does not have to be in process.

<ch> Channel number that contains the frequency range to be calibrated.

<n> ECal module number. If unspecified (only one ECal module is connected to the USB), <n> is set to 1. If two or more modules are connected, use [SENS:CORR:CKIT:ECAL:LIST?](#) to determine how many, and [SENS:CORR:CKIT:ECAL:INF?](#) to verify their identities.

<pnaPort> PNA port number.

<charN> User Characterization number to be used in determining the orientation. If unspecified, factory data (0) is used.

**Examples**

```
SENS1:CORR:CKIT:ECAL1:ORI? 2,0
```

**Return Type** The returned ECal port number is a 1-based number: 1 = Port A, 2 = Port B, 3 = Port C, 4 = Port D

**Default** Not Applicable

---

**SENSe:CORRection:CKIT:ECAL<n>:PATH:COUNT? <path>**

**(Read-only)** Returns the number of unique states that exist for the specified path name on the selected ECal module.

This command performs exactly the same function as [CONT:ECAL:MOD:PATH:COUNT?](#)

Use the [CONT:ECAL:MOD:PATH:STAT](#) command to set the module into one of those states.

Use [SENS:CORR:CKIT:ECAL:PATH:DATA?](#) to read the data for a state.

### Parameters

<n> USB number of the ECal module. If unspecified (only one ECal module is connected to the USB), <n> is set to 1. If two or more modules are connected, use [SENS:CORR:CKIT:ECAL:LIST?](#) to determine how many, and [SENS:CORR:CKIT:ECAL:INF?](#) to verify their identities.

<path> Name of the path for which to read number of states. Choose from:

Reflection paths

- **A**
- **B**
- **C** (4-port modules)
- **D** (4-port modules)

Transmission paths

- **AB**
- **AC** (4-port modules)
- **AD** (4-port modules)
- **BC** (4-port modules)
- **BD** (4-port modules)
- **CD** (4-port modules)

### Examples

```
CONT:ECAL:MOD:PATH:COUNT?  
control:ecal:module2:path:count?
```

**Return Type** Integer

**Default** Not Applicable

## **SENSe<ch>:CORRection:CKIT:ECAL[num]:PATH:DATA? <path>, <stateNum>|,<char>|**

**(Read-only)** Returns the data for a state from the memory of the selected ECal module. The returned data is interpolated if necessary to have the same stimulus values as the specified channel <ch>.

- For a reflection path state, the data is reflection S-parameter data. The number of values equals the number of stimulus points on the channel multiplied by 2 (because they are complex numbers).
- For a transmission path state, the data is all 4 S-parameters of the state. The number of values returned is 4 times that of a reflection state.

The data is returned in the same format as [CALC:DATA:SNP?](#)

**Note:** This command returns SNP data without header information, and in columns, not in rows as .SnP files. This means that the data returned from this command sends all frequency data, then all Sx1 magnitude or real data, then all Sx1 phase or imaginary data, and so forth.

### **Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1.
- [num] Optional argument. USB number of the ECal module. If unspecified (only one ECal module is connected to the USB), <num> is set to 1. If two or more modules are connected, use [SENS:CORR:CKIT:ECAL:LIST?](#) to determine how many, and [SENS:CORR:CKIT:ECAL:INF?](#) to verify their identities.
- <path> Name of the path for which to read number of states. Choose from:

#### Reflection paths

- **A**
- **B**
- **C** (4-port modules)
- **D** (4-port modules)

#### Transmission paths

- **AB**
- **AC** (4-port modules)
- **AD** (4-port modules)
- **BC** (4-port modules)
- **BD** (4-port modules)

- **CD** (4-port modules)

<stateNum> Number of the state to set. Refer to the following table to associate the <stateNum> with a state in your ECal module.

In addition, [CONT:ECAL:MOD:PATH:COUNt?](#) returns the number of states in the specified ECal module.

<stateNum>	N4432A and N4433A States	N4431A States	N469x States**	8509x States
<b>One-Port Reflection States</b>				
1	Open	Open	Impedance 1	Open
2	Short	Short	Impedance 2	Short
3	Impedance 1	Impedance 1	Impedance 3	Impedance 1
4	Impedance 2	Impedance 2	Impedance 4	Impedance 2
5			Impedance 5	
6			Impedance 6	
7			Impedance 7	
<b>Two-Port Transmission States</b>				
1	Thru	Thru	Thru	Thru
2	Confidence	Confidence	Confidence	Confidence

\*\* The following modules have only FOUR Impedance states (1, 2, 3, 4):  
N4690B ,N4691B ,N4692A ,N4696B.

<char> Optional argument. Specifies which characterization within the ECal module to read information from. If not specified, value is set to CHAR0.

Choose from:

- **CHAR0** Factory characterization (data that was stored in the ECal module by Agilent)
- **CHAR1** User characterization #1



- **CHAR2** User characterization #2
- **CHAR3** User characterization #3
- **CHAR4** User characterization #4
- **CHAR5** User characterization #5

**Examples** `SENS:CORR:CKIT:ECAL1:PATH:DATA? A, 1`

**Return Type** S1P or S2P

**Default** Not Applicable

## **SENSe:CORRection:CKIT:EXPort <kit>[,<file>]**

**(Write-only)** Saves an existing cal kit definitions to a file. Use this command to archive or move a user-defined or modified cal kit to a different PNA. After exporting the cal kit, use [SENS:CORR:CKIT:IMPort](#) to make the cal kit available for use on the PNA. This command provides the same behavior as the Installed Kits - Save As button on the [Edit PNA Cal Kits](#) dialog.

### **Parameters**

- <kit>** String. Not case sensitive. Name of the cal kit to export, as seen in the Cal Kits field of the [Select DUT Connectors and Cal Kits](#) dialog of a SMART Cal.
- <file>** Optional String argument. Path and filename to where the Cal Kit file is to be saved. If not specified, the file is saved using <kit> + ".ckt". If the path is not specified, it is stored in "C:/Program Files/Agilent/Network Analyzer/PNACalKits/User".

**Examples**

```
'File unspecified
SENS:CORR:CKIT:EXP "MyCalKit"

'Both parameters are specified
sense:correction:ckit:export "MyCalKit", "C:/myBackupCalKit.ckt"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

## **SENSe:CORRection:CKIT:IMPort <string>**

**(Write-only)** Imports the specified cal kit (.ckt file) and appends the imported kit to the end of the list of kits.

**Parameters**

<string> Path and cal kit name.

**Examples**

```
SENSe:CORRection:CKIT:IMPort "C:/Program Files/Agilent/Network Analyzer/Documents/85033D.ckt"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe:CORRection:CKIT:INITialize[:IMMediate] [ckit]**

**(Write-only)** Restores default factory installed cal kits.

**Note:** This command can also delete all existing User-defined Cal Kits. However, if saved using Save As, these kits can be restored in the same manner as after a PNA firmware upgrade. [Learn more about saving modified Cal Kits.](#)

**Parameters**

[ckit] Optional String. Cal Kit to restore. If not specified, all PNA factory Cal kits are restored.

**Examples**

```
SENS:CORR:CKIT:INITialize  
sense:correction:ckit:initialize:immediate "85052B"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe:CORRection:CKIT:LOAD <string>**

**(Write-only)** Loads the specified collection of cal kits from a .wks file. You can make your own collection of cal kits from the [Advanced Modify Cal Kit](#) menu.

**Parameters**

<string> Path and file name of the cal kit collection.

**Examples**

```
sense:correction:ckit:load "C:/Program Files/Agilent/Network Analyzer/PnaCalKits/factory/wMyCalKits.wks"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

Last modified:

19-May-2009	Added [ckit] argument to clear and init
24-Mar-2009	Edited Path SNP
6-Mar-2009	Added ECal orient?
31-Oct-2008	Added Characterizations (8.33)
16-Jun-2008	Added CKIT:INIT note
10/16/06	Modified Ecal:Data to include <ch>

## Sense:Correction:Collect:Ckit Commands

---

Use to change the definitions of calibration kit standards.

**SENSe:CORRection:COLLect:CKIT:**

| **CATalog?**

| **CONNector**

| **ADD**

| **CATalog?**

| **DELeTe**

| **FNAME**

| **SNAME**

| **DESCription**

| **INFormation?**

| **NAME**

| **OLAB**

| **OLISt?**

| **ORDer**

| **PORT[:SElect]**

| **RESet**

| **SElect**

| **STANdard**

| **CO, C1, C2, C3**

| **CHARacter**

| **DELay**

| **FMAX**

| **FMIN**

| **IMPedance**

| **LO, L1, L2, L3**

<b>LABel</b>
<b>LOSS</b>
<b>REMove</b>
<b>SDEscription</b>
<b>[SE]lect</b>
<b>TYPE</b>
<b>TZReal</b>
<b>TZImag</b>

Click on a [blue](#) keyword to view the command details.

**Red** keywords are superseded commands.

Most of these commands act on the currently selected standard from the currently selected calibration kit.

- To select a Calibration kit, use [SENS:CORR:COLL:CKIT:SEL](#).
- To select a Calibration standard, use [SENS:CORR:COLL:CKIT:STAN:SEL](#)
- See an **example** program that [CREATES a New Cal Kit](#)
- See an **example** program that [MODIFIES an Existing Cal Kit](#)
- [Learn about Modifying Cal Kits](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Note:** You should provide data for every definition field - for every standard in your calibration kit. If a field is not set, the default value may not be what you expect.

For more information, read [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(Application Note 1287-11\)](#)

---

**SENSe:CORRection:COLLect:CKIT:CATalog?**

**(Read-only)** Returns the names of the first 50 mechanical cal kits in your PNA that can be used for unguided calibrations.

**Examples** `SENS:CORR:COLL:CKIT:CAT?`

**Return Type** A comma-separated string

**Default** Not Applicable

---

### **SENSe:CORRection:COLLect:CKIT:CONNector:ADD <family>,<start>,<stop>,<z0>,<gender>,<media>,<cutoff>**

**(Write only)** Creates a new connector. The connector is automatically added to the list of available connectors for the currently selected cal kit. If a connector includes both male and female connectors, each connector must be added separately.

#### **Parameters**

<family> (String) Name of connector family. Limited to 50 characters.

<start> Start frequency

<stop> Stop frequency

<z0> Characteristic Impedance of the connector in ohms.

<gender> Connector gender. Choose from:  
MALE  
FEMALE  
NONE

<media> Media of the connector. Choose from:

**COAX** - coaxial

**WAVE** - waveguide

<cutoff> Cutoff frequency of the connector (waveguide only).

**Examples** `SENS:CORR:COLL:CKIT:CONN:ADD "PSC 1.8 mm",0 HZ,999.9  
GHZ,50,FEMALE,COAX,0.0  
SENS:CORR:COLL:CKIT:CONN:ADD "PSC 1.8 mm",0 HZ,999.9  
GHZ,50,MALE,COAX,0.0`

**Query Syntax** Not applicable

**Default** Not Applicable

---

### **SENSe:CORRection:COLLect:CKIT:CONNector:CATalog?**

**(Read-only)** Returns a comma-separated list of all connectors defined within the currently selected cal kit. The returned string includes the connector family name followed by the connector gender, if any. Kits may include a primary connector family name and additional connector family names.

Connector family names are case sensitive. A connector family named "PSC 2.4" is different from a connector family named "psc 2.4".

Learn more about [Connector Family Name](#)

**Examples**

```
SENS:CORR:COLL:CKIT:CONN:CAT?
```

'Returned string

```
"Type-N (50) male, Type-N (50) female"
```

**Default**

Not Applicable

---

## **SENSe:CORRection:COLLect:CKIT:CONNector:DELeTe**

**(Write-only)** Deletes the primary connector family name from the selected kit. The PNA allows multiple connector families for each kit. If a kit includes multiple connector families, only the first listed (primary) connector family name is deleted.

Once the connector family is deleted, the connector may not be assigned to any new or existing standard within the kit.

The previously defined standards retain their association to the deleted connector name. To reassign standards to a new connector family name, use [SENSe:CORRection:COLLect:CKIT:CONNector:SNAME](#).

**Examples**

```
SENS:CORR:COLL:CKIT:CONN:DEL
```

**Query Syntax**

Not Applicable

**Default**

Not Applicable

---

## **SENSe:CORRection:COLLect:CKIT:CONNector:FNAME <name>**

**(Read-Write)** Replaces the primary connector family name from the selected kit with a new connector family name. The connector family name is replaced in all standards in the kit that share that name. The PNA allows multiple connector families for each kit. If a kit includes multiple connector families, only the first listed (primary) connector family name is replaced. Use the query form of this command to return the name of the primary connector family.

#### Parameters

<name> New connector family name. Limited to 50 characters.

#### Examples

```
SENS:CORR:COLL:CKIT:CONN:FNAME 'MYPSC35'
```

```
Sense:correction:collect:ckit:connector:name 'My Type N'
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:CONNector:FNAME?

**Return Type** String

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:CKIT:CONNector:SNAME <family>,<gender>,<port>

**(Read-Write)** Assigns a family name to the currently selected standard from the currently selected kit. Specify each port of a 2-port standard individually. Use the query form of this command to read the connector family name assigned to the current standard. The name is not assigned unless the connector family name is previously defined within the selected kit.

#### Parameters

<family> String. Connector family name.

<gender> Connector gender. Choose from:  
MALE  
FEMALE  
NONE

<port> Number of the connector port to be assigned the connector family name. 2-port standards such as a thru line must be assigned separately. It is not relevant which connector is port 1 or port 2.

1 Specifies a 1-port standard or the first port of a 2-port standard.

2 Specifies the second port of a 2-port standard.

#### Examples

```
SENS:CORR:COLL:CKIT:CONN:SNAME "Type-N (50)",MALE,1
```

**Query Syntax** SENSe:CORRection:COLLect:CKIT:CONNector:SNAME?

**Return Type** String



Default Not Applicable

---

### SENSe:CORRection:COLLect:CKIT:DESCription <string>

(Read-Write) Modifies the cal kit description field of the selected kit. This description appears in the [Edit PNA Cal Kit dialog box](#).

#### Parameters

<string> Description of the cal kit. Limited to 50 characters.

#### Examples

```
SENS:CORR:COLL:CKIT:DESC "My New CalKit"
```

#### Query Syntax

SENSe:CORRection:COLLect:CKIT:DESCription?

#### Return Type

String

#### Default

Not Applicable

---

### SENSe:CORRection:COLLect:CKIT:INFormation? <module>[,char]

(Read Only) Reads characterization information from an ECal module.

#### Parameters

<module> Specifies which ECal module to read from. Choose from:

**ECAL1**

.through.

**ECAL8**

[char] Optional argument.

Specifies which characterization within the ECal module to read information from. If this argument is not used, the default is **CHAR0**. **CHAR1** through **CHAR5** are for user characterizations that may have been written to the module by the User Characterization feature on the PNA. Choose from:

**CHAR0** Factory characterization (data that was stored in the ECal module by Agilent)

**CHAR1** User characterization #1

**CHAR2** User characterization #2

**CHAR3** User characterization #3

**CHAR4** User characterization #4

**CHAR5** User characterization #5

**Examples**

```
SENS:CORR:COLL:CKIT:INF? ECAL4  
sense:correction:collect:ckit:information? ecal2,char1
```

Example return string:

```
ModelNumber: 85092-60007, SerialNumber: 01386, ConnectorType:  
N5FN5F, PortAConnector: Type N (50) female, PortBConnector: Type  
N (50) female, MinFreq: 30000, MaxFreq: 9100000000,  
NumberOfPoints: 250, Calibrated: July 4 2002
```

**Return Type** Character

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:NAME <name>**

(Read-Write) Sets a name for the selected calibration kit.

**Parameters**

<name> Calibration Kit name. Any string name, can include numerics, period, and spaces; any length (although the dialog box display is limited to about 30 characters).

**Examples**

```
SENS:CORR:COLL:CKIT:NAME 'MYAPC35'  
sense:correction:collect:ckit:name 'mytypen'
```

**Query Syntax** SENSe:CORRection:COLLect:CKIT:NAME?

**Return Type** String

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:OLABel<class> <name>**

(Read-Write) Sets the label for the calibration class designated by <class>. The label is used in the prompts for connecting the calibration standards associated with that <class>.

**Parameters**

<class> Number of the calibration class. Choose a number between: 1 and 18. The <class> numbers are associated with the following calibration Classes:

	<b>Class</b>	<b>Description</b>
1	S11A	Reflection standard
2	S11B	Reflection standard
3	S11C	Reflection standard
4	S21T	Thru/Delay standard
5	S22A	Reflection standard
6	S22B	Reflection standard
7	S22C	Reflection standard
8	S12T	Thru/Delay standard

#### **3-port analyzers only**

9	S33A	Reflection standard
10	S33B	Reflection standard
11	S33C	Reflection standard
12	S32T	Thru/Delay standard
13	S23T	Thru/Delay standard
14	S31T	Thru/Delay standard
15	S13T	Thru/Delay standard

#### **TRL Calibrations**

16	TRL "T"	Thru standard
17	TRL "R"	Reflect standard
18	TRL "L"	Line standard

<name> Label for the calibration class. Must be enclosed in quotes. Any string between 1 and 12 characters long. Cannot begin with a numeric.

#### **Examples**

```
SENS:CORR:COLL:CKIT:OLAB3 'LOADS'
sense:correction:collect:ckit:olabel14 'Thru'
```

**Return Type** String

**Default** Not Applicable

## **SENSe:CORRection:COLLect:CKIT:OLIST[class]?**

**(Read-only)** Returns seven values of standards that are assigned to the specified class.

#### **Parameters**

<class> Number of the calibration class to be queried. The <class> numbers are associated with the following calibration Classes:

	<b>Class</b>	<b>Description</b>
1	S11A	Reflection standard
2	S11B	Reflection standard
3	S11C	Reflection standard
4	S21T	Thru/Delay standard
5	S22A	Reflection standard
6	S22B	Reflection standard
7	S22C	Reflection standard
8	S12T	Thru/Delay standard

### **3-port analyzers ONLY**

**4-port analyzers use S11 and S22 classes ([see example program](#))**

9	S33A	Reflection standard
10	S33B	Reflection standard
11	S33C	Reflection standard
12	S32T	Thru/Delay standard
13	S23T	Thru/Delay standard
14	S31T	Thru/Delay standard
15	S13T	Thru/Delay standard

### **TRL Calibrations**

16	TRL "T"	Thru standard
17	TRL "R"	Reflect standard
18	TRL "L"	Thru standard

#### **Examples**

**SENS:CORR:COLL:CKIT:OLIST8?**

Always returns 7 standard numbers. Unassigned standards return 0

#### **Return Type**

Numeric; returns the <class> number of the selected standard.

#### **Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:ORDer<class> <std> [,<std>] [,<std>] [,<std>] [,<std>] [,<std>] [,<std>]**

**(Read-Write)** Sets a standard number to a calibration class. Does **NOT** set or dictate the order for measuring the standards. For more information, see Assigning Standards to a Calibration Class

### Parameters

<class> Number of the calibration class that is assigned to <standard>. Choose a number between: **1** and **18**. The <class> numbers are associated with the following calibration Classes:

	<b>Class</b>	<b>Description</b>
1	S11A	Reflection standard
2	S11B	Reflection standard
3	S11C	Reflection standard
4	S21T	Thru/Delay standard
5	S22A	Reflection standard
6	S22B	Reflection standard
7	S22C	Reflection standard
8	S12T	Thru/Delay standard

### **3-port analyzers ONLY**

**4-port analyzers use S11 and S22 classes** ([see example program](#))

9	S33A	Reflection standard
10	S33B	Reflection standard
11	S33C	Reflection standard
12	S32T	Thru/Delay standard
13	S23T	Thru/Delay standard
14	S31T	Thru/Delay standard
15	S13T	Thru/Delay standard

### **TRL Calibration**

16	TRL "T"	Thru standard
17	TRL "R"	Reflect standard
18	TRL "L"	Line standard

<std> Standard number to be assigned to the class; Choose a standard between 1 and 8. One standard is mandatory; up to six additional standards are optional.

**Examples**

```
Assigns standard 3 to S11A class:  
SENS:CORR:COLL:CKIT:ORD1 3  
Assigns standard 2 and 5 to S21T class class:  
sense:correction:collect:ckit:order4 2,5
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:ORDer<class>?

'Returns only the first standard assigned to the specified class. To query the remaining standards, use [SENSe:CORRection:COLLect:CKIT:OLIST\[1-15\]?](#)

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe<cnum>:CORRection:COLLect:CKIT:PORT<n>[:SELection] <string>**

(Read-Write) Sets and returns the name of the Cal Kit to use for unguided cal.

**Parameters**

<cnum> Currently not used. The unguided cal kit selection is for all ports on all channels.

<n> Currently not used. The unguided cal kit selection is for all ports on all channels.

<string> Cal Kit name enclosed in quotes. Use [SENS:CORR:COLL:CKIT:CAT?](#) to read a list of all available Cal Kits in the PNA.

**Examples**

```
SENS:CORR:COLL:CKIT:PORT "85052B"  
sense2:correction:collect:ckit:port:select "85052D"
```

**Query Syntax**

SENSe<cnum>:CORRection:COLLect:CKIT:PORT<n>:SELECT?

**Return Type**

String

**Default**

Last kit selected

---

**SENSe:CORRection:COLLect:CKIT:RESet <num> - Superseded**

This command is replaced by [Sens:Corr:Ckit:Init](#).

**(Write-only)** Resets the selected calibration kit to factory default definition values.

#### Parameters

<num> The number of the calibration kit to be reset. Choose any integer between: **1** and **8**

#### Examples

```
SENS:CORR:COLL:CKIT:RESet 1  
sense:correction:collect:ckit:reset 4
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<cnum>:CORRection:COLLect:CKIT[:SElect] <num>

**(Read-Write)** Selects (makes active) a calibration kit for **performing** an **UNGUIDED** calibration or for **modifying** standards. All subsequent "CKIT" commands that are sent apply to this selected calibration kit. Select a calibration standard using [SENS:CORR:COLL:CKIT:STAN <num>](#). Kits 1 to approximately kit 37 are factory installed Cal Kits.

Use the [Edit PNA Cal Kits](#) dialog to see the name that is associated with each Kit ID <num>.

Another method is to iterate through the cal kits, select the kit using this command, then query the name using [SENS:CORR:COLL:CKIT:NAME](#).

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<num> The number of the calibration kit. Choose from:

Use [SENSe:CORRection:COLLect:CKIT:RESet](#) to restore Cal Kits to default values.

#### Name

- 1 Cal Kit 1
- 2 Cal Kit 2
- 3 Cal Kit 3
- "
- "
- 48 Cal Kit 48
- 49 Cal Kit 49
- 50 Cal Kit 50

99 ECal module

**Examples**

```
SENS:CORR:COLL:CKIT 2  
sense2:correction:collect:ckit:select 7
```

**Query Syntax**

SENSe<cnum>:CORRection:COLLect:CKIT?

**Return Type**

Numeric

**Default**

Last kit selected

---

**SENSe:CORRection:COLLect:CKIT:STANdard:C0 <num>**

(Read-Write) Sets the C0 value (the first capacitance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for C0 in femtofarads (1E-15)

**Examples**

The following commands set C0=15 femtofarads:

```
SENS:CORR:COLL:CKIT:STAN:C0 15  
sense:correction:collect:ckit:standard:c0 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:C0?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:C1 <num>**

(Read-Write) Sets the C1 value (the second capacitance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for C1.

**Examples**

The following two commands set C1=15:

```
SENS:CORR:COLL:CKIT:STAN:C1 15  
sense:correction:collect:ckit:standard:c1 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:C1?

**Return Type**

Numeric

**Default**

Not Applicable

---



## SENSe:CORRection:COLLect:CKIT:STANdard:C2 <num>

(Read-Write) Sets the C2 value (the third capacitance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

### Parameters

<num> Value for C2.

### Examples

The following two commands set C2:

```
SENS:CORR:COLL:CKIT:STAN:C2 15
sense:correction:collect:ckit:standard:c2 15
```

**Query Syntax** SENSe:CORRection:COLLect:CKIT:STANdard:C2?

**Return Type** Numeric

Default Not Applicable

---

## SENSe:CORRection:COLLect:CKIT:STANdard:C3 <num>

(Read-Write) Sets the C3 value (the fourth capacitance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

### Parameters

<num> Value for C3.

### Examples

The following two commands set C3

```
SENS:CORR:COLL:CKIT:STAN:C3 15
sense:correction:collect:ckit:standard:c3 15
```

**Query Syntax** SENSe:CORRection:COLLect:CKIT:STANdard:C3?

**Return Type** Numeric

Default Not Applicable

---

## SENSe:CORRection:COLLect:CKIT:STANdard:CHARacter <char>

**(Read-Write)** Sets the media type of the selected calibration standard.

**Parameters**

<char> Media type of the standard. Choose from:  
**Coax** - Coaxial Cable  
**Wave** - Waveguide

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:CHAR COAX
sense:correction:collect:ckit:standard:character wave
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:CHARacter?

**Return Type** Numeric

**Default** Coax

---

**SENSe:CORRection:COLLect:CKIT:STANdard:DELay <num>**

**(Read-Write)** Sets the electrical delay value for the selected standard.

**Parameters**

<num> Electrical delay in picoseconds

---

**Examples**

The following two commands set delay to 50 picoseconds

```
SENS:CORR:COLL:CKIT:STAN:DEL 50e-12
sense2:correction:collect:ckit:standard:delay 50ps
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:DELay?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:FMAX <num>**

**(Read-Write)** Sets the maximum frequency for the selected standard.

**Parameters**

<num> Maximum frequency in Hertz.

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:FMAX 9e9
sense:correction:collect:ckit:standard:fmax 9Ghz
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:FMAX?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:FMIN <num>**

**(Read-Write)** Sets the minimum frequency for the selected standard.

**Parameters**

<num> Minimum frequency in Hertz.

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:FMIN 1e3
sense:correction:collect:ckit:standard:fmin 1khz
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:FMIN?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:IMPedance <num>**

**(Read-Write)** Sets the characteristic impedance for the selected standard.

**Parameters**

<num> Impedance in Ohms

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:IMP 75
sense:correction:collect:ckit:standard:impedance 50.3
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:IMPedance?

**Return Type**

Numeric

**Default**

50

---

**SENSe:CORRection:COLLect:CKIT:STANdard:L0 <num>**

**(Read-Write)** Sets the L0 value (the first inductance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for L0 in femtohenries (1E-15)

**Examples**

The following two commands set L0=15 femtohenries:

```
SENS:CORR:COLL:CKIT:STAN:L0 15
sense:correction:collect:ckit:standard:10 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:L0?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:L1 <num>**

**(Read-Write)** Sets the L1 value (the second inductance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for L1.

**Examples**

The following two commands set L1=15:

```
SENS:CORR:COLL:CKIT:STAN:L1 15
sense:correction:collect:ckit:standard:11 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:L1?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:L2 <num>**

**(Read-Write)** Sets the L2 value (the third inductance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for L2.

**Examples**

The following two commands set L2=15:

```
SENS:CORR:COLL:CKIT:STAN:L2 15  
sense:correction:collect:ckit:standard:12 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:L2?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:L3 <num>**

**(Read-Write)** Sets the L3 value (the fourth inductance value) for the selected standard. For a detailed discussion of this value, search for App Note 8510-5B at [www.Agilent.com](http://www.Agilent.com).

**Parameters**

<num> Value for L3.

**Examples**

The following two commands set L3=15:

```
SENS:CORR:COLL:CKIT:STAN:L3 15  
sense:correction:collect:ckit:standard:13 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:L3?

**Return Type**

Numeric

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:LABel <name>**

**(Read-Write)** Sets the label for the selected standard. The label is used to prompt the user to connect the specified standard.

**Parameters**

<name> Label for the standard; Must be enclosed in quotes. Any string between **1** and **12** characters long. Cannot begin with a numeric.

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:LAB 'OPEN'  
sense:correction:collect:ckit:standard:label 'Short2'
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:LABel?

**Return Type** String

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:LOSS <num>**

**(Read-Write)** Sets the insertion loss for the selected standard.

**Parameters**

<num> Insertion loss in Gohms / sec. (GigaOhms per second of electrical delay)

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:LOSS 3.5e9  
sense:correction:collect:ckit:standard:loss 3
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:LOSS?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:REMove**

**(Write only)** Deletes the selected standard from the selected cal kit.

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:REMove
```

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:SDEscription <string>**

**(Read-Write)** Modifies the description of the selected standard of the selected kit. This description appears in the [edit kit dialog box](#).

**Parameters**

<string> Description of the standard.

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:SDS "My New Standard"
```

**Query Syntax**

```
SENSe:CORRection:COLLect:CKIT:STANdard:SDScription?
```

**Return Type**

String

**Default**

Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard[:SELEct] <num>**

**(Read-Write)** Selects the calibration standard. All subsequent "CKIT" commands to modify a standard will apply to the selected standard. Select a calibration kit using [SENS:CORR:COLL:CKIT:SEL](#)

**Parameters**

<num> Number of the standard. Choose any number between:  
**1** and **30**

**Examples**

```
SENS:CORR:COLL:CKIT:STAN 3  
sense:correction:collect:ckit:standard:select 8
```

**Query Syntax**

```
SENSe:CORRection:COLLect:CKIT:STANdard[:SELEct]?
```

**Return Type**

Numeric

**Default**

1

---

**SENSe:CORRection:COLLect:CKIT:STANdard:TYPE <char>**

**(Read-Write)** Sets the type for the selected standard.

**Parameters**

<char> Choose from:  
**OPEN**  
**SHORT**  
**LOAD**  
**SLOAD** (sliding load)  
  
**THRU** (through)  
  
**ARBI**(arbitrary)

**Examples**

```
SENS:CORR:COLL:CKIT:STAN:TYPE LOAD  
sense:correction:collect:ckit:standard:type short
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:TYPE?

**Return Type** Character

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:TZReal <num>**

**(Read-Write)** Sets the TZReal component value of the Terminal Impedance for the selected standard.

**Note:** Only applicable when the Standard Type is set to **ARBI**

**Parameters**

<num> Value for TZReal in Ohms

**Examples**

The following commands set TZReal=15 Ohms:

```
SENS:CORR:COLL:CKIT:STAN:TZReal 15  
sense:correction:collect:ckit:standard:TZReal 15
```

**Query Syntax** SENSE:CORRection:COLLect:CKIT:STANdard:TZReal?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:CKIT:STANdard:TZImag <num>**



**(Read-Write)** Sets the TZImag component value of the Terminal Impedance for the selected standard.

**Note:** Only applicable when the Standard Type is set to **ARBI**

**Parameters**

<num> Value for TZImag in Ohms

**Examples**

The following two commands set TZImag=15 Ohms:

```
SENS:CORR:COLL:CKIT:STAN:TZImag 15  
sense:correction:collect:ckit:standard:TZImag 15
```

**Query Syntax**

SENSe:CORRection:COLLect:CKIT:STANdard:TZImag?

**Return Type**

Numeric

**Default**

Not Applicable

---

Last Modified:

17-Mar-2010	Added Ckit:CAT and CKIT:Select by string (9.2)
10-Nov-2008	Clarified Select command
30-Oct-2008	Fixed SDES query per CN
14-Apr-2008	Added link to app note
17-Sep-2007	Fixed 'select' command

## Sense:Correction:Cset Commands

---

Performs actions on calibration sets.

### **SENSe:CORRection:CSET**

**[ACTivate](#)**

**[CATalog?](#)**

**[COPY](#)**

**[CREate](#)**

**[DATA](#)**

**[DELeTe](#)**

**[DESCription](#)**

**[ETERm](#)**

**[FLATten](#)**

**[GUID](#)**

**[NAME](#)**

**[\[SELeCt\]](#)**

**[SAVE](#)**

**[STANdard](#)**

**[STIMulus?](#)**

**[TSET](#)**

| **[ALLPorts?](#)**

| **[TYPE?](#)**

**[TYPE](#)**

| **[CATalog?](#)**

Click on a [blue](#) keyword to view the command details.

## See Also

- [Creating Cal Sets](#)
  - [Example Programs](#)
  - [Learn about Cal Sets](#)
  - [Synchronizing the PNA and Controller](#)
- 

### SENSe<cnum>:CORRection:CSET:ACTivate <string>, <bool>

This command replaces [SENS:CORR:CSET:GUID](#)

**(Read-Write)** Selects and applies a Cal Set to the specified channel.

Use [SENS:CORR:CSET:CAT?](#) to list the Cal Sets.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <string> Cal Set to make active. Specify the Cal Set by **GUID** or **Name**. Use [SENS:CORR:CSET:CAT?](#) to list the available Cal Sets in either format.
- <bool> Should the Cal Set stimulus values be applied to the channel. Choose from:
- ON (1)** Apply the Cal Set stimulus values to the channel.
- OFF (0)** Do NOT apply the Cal Set stimulus values. If the Cal Set stimulus values do not match the channel stimulus values, then the following will occur:
- If interpolation is ON, then interpolation will be attempted. This may fail if the channel frequency is outside the range of the Cal Set.
  - If interpolation is OFF, the selection will be abandoned and an error is returned:

#### Examples

```
SENS:CORR:CSET:ACT "My2Port",1  
  
sense:correction:cset:activate? name  
'returns  
"My2Port"
```

**Query Syntax** SENSe<cnum>:CORRection:CSET:ACTivate? [GUID|NAME]

Returns the name of the Cal Set that is applied to the specified channel. Choose from **GUID** or **NAME** to specify which string is returned. If unspecified, the GUID of the Cal Set is returned. If no Cal Set is applied to the specified channel, then "No Calset Selected" is returned.

**Return Type** String  
**Default** Not Applicable

---

### SENSe:CORRection:CSET:CATalog? [char]

(Read-only) Returns a list of Cal Sets.

#### Parameters

<char> Optional argument. The list is returned in one of the following formats. Both return comma-separated string lists.

**GUID** Cal Sets are listed by GUID (Default if unspecified).

**NAME** Cal Sets are listed by Name

#### Examples

```
SENS:CORR:CSET:CAT?
```

```
'Returns:
```

```
{FD6F863E-9719-11d5-8D6C-00108334AE96},{1B03B2CE-971A-11d5-8D6C-00108334AE96}
```

```
sense2:correction:cset:catalog? name
```

**Default** Not Applicable

---

### SENSe<cnun>:CORRection:CSET:COpy <string>

(Write-only) Creates a new Cal Set and copies the current Cal Set data into it. Use this command to manipulate data on a Cal Set without corrupting the original cal data.

#### Parameters

<cnun> Channel number using the Cal Set to be copied. If unspecified, value is set to 1

<string> Name of the new Cal Set.

#### Examples

```
SENS2:CORR:CSET:COpy 'My2Port'
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<cnun>:CORRection:CSET:CREate [name]

**(Write-only)** Creates an empty Cal Set and attaches it to the specified channel. This command is ONLY necessary before remotely filling the Cal Set with error term data. (For Advanced Users).

A Cal Set is automatically created, applied to the channel, and saved at the completion of a guided cal according to the preference setting [SENS:CORR:PREF:CSET:SAVE](#).

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- [name] Optional argument. Name of the Cal Set. Spaces or punctuation are NOT allowed. If unspecified, a unique name is chosen in the form "Calset\_N" where N is a unique number.

#### Examples

```
SENS:CORR:CSET:CRE 'My2Port'
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### **SENSe<num>:CORRection:CSET:DATA <eterm, portA, portB,>[<param>] <block>**

**(Read-Write)** Read or Write a specific error term from/to the Cal Set currently attached to the specified channel. (For Advanced Users). The command can be used only for the error terms listed. See [SENS:CORR:CSET:ETERM](#) to get and put error term data using a string argument for all error terms..

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <eterm, portA, portB> **Error Term, Port pair of the specified error term.**
- Although not all error terms use two port numbers, two are required by the PNA in all cases. Each port number must be between 1 and the number of ports on the PNA.
- EDIR** - directivity
- portA: the port at which directivity is measured.
- portB: Not used, but must be a valid PNA port number.
- ESRM** - source match
- portA: the port at which source match is measured.
- portB: Not used, but must be a valid PNA port number.

### **ERFT** - reflection tracking

portA: the port at which reflection tracking is measured.

portB: Not used, but must be a valid PNA port number.

### **ELDM** - load match

portA: the port at which load match is measured.

portB: the source port.

Load match is measured with a cable connected between the measured port (portA) and the source port (portB).

The cal system requires that the complete matrix of loadmatch arrays be filled.

In most cases you can measure loadmatch once at a port, driven by any other port. Then use that data for all variations of the receive port. (The exception is the 3-port PNA models, which requires the loadmatch-measured port to be driven by every other port.)

For example: Measure the loadmatch at port2 while driving port1. Then upload this same data to the following arrays:

ELDM,2,1,<data>

ELDM,2,3,<data>

ELDM,2,4,<data>

### **ETRT** - transmission tracking

portA: the receive port

portB: the source port for this measurement

### **EXTLK** - crosstalk

portA: the receive port

portB: the source port for this measurement

### **ERSPT** - response tracking.

portA: Not used, but must be a valid PNA port number.

portB: Not used, but must be a valid PNA port number.

**ERSPI** - response isolation.

portA: Not used, but must be a valid PNA port number.

portB: Not used, but must be a valid PNA port number.

<param> **<string>** - required ONLY when Eterm is response tracking (**ERSPT**) or response isolation (**ERSPI**). Specify the S-parameter, ratio, or unratioed measurement for which the Eterm applies.

[Logical receiver notation](#) is allowed. Ratioed measurements do not require source port to be specified.

A full 4-port calibration requires the following terms be uploaded:

		PORT B			
		1	2	3	4
P O R T A	1	EDIR,1,1	ELDM,1,2	ELDM,1,3	ELDM,1,4
		ERFT,1,1	ETRT,1,2	ETRT,1,3	ETRT,1,4
		ESRM,1,1	EXTLK,1,2	EXTLK,1,3	EXTLK,1,4
	2	ELDM,2,1	EDIR,2,2	ELDM,2,3	ELDM,2,4
		ETRT,2,1	ERFT,2,2	ETRT,2,3	ETRT,2,4
		EXTLK,2,1	ESRM,2,2	EXTLK,2,3	EXTLK,2,4
	3	ELDM,3,1	ELDM,3,2	EDIR,3,3	ELDM,3,4
		ETRT,3,1	ETRT,3,2	ERFT,3,3	ETRT,3,4
		EXTLK,3,1	EXTLK,3,2	ESRM,3,3	EXTLK,3,4
	4	ELDM,4,1	ELDM,4,2	ELDM,4,3	EDIR,4,4
		ETRT,4,1	ETRT,4,2	ETRT,4,3	ERFT,4,4
		EXTLK,4,1	EXTLK,4,2	EXTLK,4,3	ESRM,4,4

Reflection terms

Transmission terms

<block> (Block). Error term data. A Real / Imaginary data pair for each data point.

Format is set using [FORM:DATA](#) command.

For REAL binary formats, refer to [Getting Data from the Analyzer using SCPI](#)

**Examples**

```
'Set the directivity term with a cal set using 5 points  
SENS1:CORR:CSET:DATA EDIR, 1, 1, +6.12569600000E-002,-  
7.27163800000E-003,-3.63812000000E-003,+1.33521800000E-002,-  
4.36775100000E-003,+1.87792400000E-002,-4.09239100000E-  
003,+4.24291200000E-002,-2.03784900000E-002,+3.21425100000E-002"
```

**Query Syntax** SENSE<cnum>:CORRection:CSET:DATA? <eterm,portA,portB>

**Return Type** Block data

**Default** Not Applicable

---

**SENSe:CORRection:CSET:DELeTe <string>**

**(Write-only)** Deletes a Cal Set from the set of available Cal Sets. This method immediately updates the Cal Set file on the hard drive. If the Cal Set is currently being used by a channel or does not exist, this request will be denied and an error is returned.

**Parameters**

<string> Cal Set to be deleted. Specify the Cal Set by **GUID** or **Name**. Use [SENS:CORR:CSET:CAT?](#) to list the available Cal Sets in either format.

**Examples**

```
SENS:CORR:CSET:DEL '{2B893E7A-971A-11d5-8D6C-00108334AE96}'  
sense2:correction:cset:delete 'MyCalSet'
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe<cnum>:CORRection:CSET:DESCription <string>**

**(Read-Write)** Sets or returns the descriptive string assigned to the selected Cal Set. Change this string so that you can easily identify each Cal Set. Apply and select the Cal Set using [SENS:CORR:CSET:ACT](#).

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<string> The descriptive string associated with the currently-selected Cal Set

**Examples**

```
SENS:CORR:CSET:DESC 'MyCalSet'  
sense2:correction:cset:description 'thisCalSet'
```

**Query Syntax** SENSE<cnum>:CORRection:CSET:DESCription?

**Return Type** String

**Default** Not Applicable

---



## SENSe<cnum>:CORRection:CSET:ETERm <string>,<r, i [r,i]...>

(Read-Write) Sets or returns error term data for all PNA measurements.

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <string> (String) Error term to read or write. The error term is specified using the EXACT case-sensitive string displayed in the [Cal Set Viewer](#) utility. See [SENS:CORR:CSET:DATA](#) for a description of port numbers.

The following Noise figure error terms are listed for convenience:

- **RcvNoiseCorr\_m\_n** Noise correlation matrix of the noise receiver (a 2x2 complex matrix). The row and column indices m and n range from 1 to 2.
- **RcvT\_m\_n** T-matrix of the noise receiver (a 2x2 complex matrix). The row and column indices m and n range from 1 to 2.
- **GammaTuner\_n** Reflection coefficient for impedance state n of the embedded noise tuner (Ecal module) in the port 1 source path. For the Agilent 4691 family of Ecal modules, n can range from 1 to 7.

<r,i> (Block) Error term data. A Real / Imaginary data pair for each data point.

Format is set using [FORM:DATA](#) command.

For REAL binary formats, refer to [Getting Data from the Analyzer using SCPI](#)

### Examples

```
SENS:CORR:CSET:ETERM "Directivity(1,1)", 0.237,-1.422, 0.513,
0.895 ' set directivity(source error term for 2 points
SENS:CORR:CSET:ETERM? "Directivity(1,1)" ' read
```

**Query Syntax** SENSe<cnum>:CORRection:CSET:ETERm? <string>

**Return Type** Block data

**Default** Not Applicable

---

## SENSe<cnum>:CORRection:CSET:FLATten <string>

**(Write-only)** When a Cal Set that was produced by a calibration has been interpolated or otherwise modified (for example, by [Fixturing operations](#)) this command saves the modified Cal Set to the PNA hard drive so that it can be reused. There is no User Interface equivalent for this command.

## Background

When a Cal Set is selected for use by a channel, the channel reads the Cal Set from disk (master Cal Set). If the channel aligns perfectly with the Cal Set, the master Cal Set is used directly. In this case, the “active” Cal Set is the master Cal Set.

When processing occurs on the error terms due to interpolation or modification due to the use of fixturing, the channel will generate a temporary “memory-resident” Cal Set. In this case, the “active” Cal Set is the memory-resident Cal Set. This FLATten command allows you to save the “active” Cal Set to disk.

Depending on the measurement conditions, this flattening of the Cal Set can improve performance, especially if the Cal Set is applied often (using multiple recall states) or used by many channels.

Flattening a version of the Cal Set for each channel can avoid the interpolation or the fixturing processing that would otherwise occur when the Cal Set is selected or the instrument state is recalled.

You will have to manage the application of such a Cal Set as the PNA itself will have no way to determine what processing had been done once the flatten command is used. For example, if fixture de-embedding occurred prior to the flatten command, that Cal Set should then be applied WITHOUT fixturing on, because fixturing is already embedded in that Cal Set. It is your responsibility to apply the Cal Set properly.

### Parameters

- <num> Channel number on which the modified Cal Set resides. If unspecified, value is set to 1
- <string> Name of the new Cal Set. Spaces or punctuation NOT allowed.

### Examples

```
SENS:CORR:CSET:FLAT "MyCalSet"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe<num>:CORRection:CSET:GUID <string>** **Superseded**

This command is replaced by [SENS:CORR:CSET:ACTivate](#).

**(Read-Write)** Selects the Cal Set identified by the string parameter (GUID) and applies it to the specified channel.

- A Cal Set cannot be selected for a channel which is not ON.
- If the stimulus settings of the selected Cal Set differ from those of the selected channel, the instrument will automatically change the channel's settings to match the Cal Set.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <string> GUID of the desired Cal Set. The curly brackets and hyphens must be included.

#### Examples

```
SENS:CORR:CSET:GUID '{2B893E7A-971A-11d5-8D6C-00108334AE96}'  
sense2:correction:cset:guid '{2B893E7A-971A-11d5-8D6C-00108334AE96}'
```

**Query Syntax** SENSE<num>:CORRection:CSET:GUID?

Returns the GUID of the currently-selected Cal Set for the specified channel.

**Return Type** String

**Default** Not Applicable

---

### SENSe<num>:CORRection:CSET:NAME <string>

**(Read-Write)** Sets or queries the name of the Cal Set currently applied to the specified channel.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <string> Name of the Cal Set. Spaces or punctuation NOT allowed.

#### Examples

```
SENS:CORR:CSET:NAME 'MyCalSet'  
sense2:correction:cset:name 'thisCalSet'
```

**Query Syntax** SENSe<num>:CORRection:CSET:NAME?

**Return Type** String

**Default** Not Applicable

---

SENSe<num>:CORRection:CSET[:SElect] <char> **Superseded**

This command is replaced by [MMEM:LOAD](#)

**(Read-Write)** Recalls a \*.cst file from memory. The file name is "CSETx.cst" where x is the user number assigned to <char>. Learn more about [.cst files](#)

For more information on the file naming syntax, see the [MMEMory](#) subsystem.

**Note:** This command does NOT select a Cal Set for a channel. To select a Cal Set, use [SENS:CORR:CSET:ACTivate](#)

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

**DEF** - Presets the analyzer

**USER01** - Restores User01 calibration data

**USER02** - Restores User02 calibration data

through...

**USER10** - Restores User10 calibration data

#### Examples

```
SENS:CORR:CSET DEF  
sense2:correction:cset:select user02
```

**Query Syntax** SENSE<num>:CORRection:CSET[:SElect]?

**Return Type** Character

**Default** DEF

---

**SENSe<num>:CORRection:CSET:SAVE <char>**

This command is NOT necessary after completion of a calibration. A Cal Set is automatically created, applied to the channel, and saved at the completion of a guided cal according to the preference setting [SENS:CORR:PREF:CSET:SAVE](#).

(Read Write)

Saves the channel's Cal Set to the PNA hard drive. For example, use this command after writing data to a Cal Set using [SENS:CORR:CSET:DATA](#) (For Advanced Users).

The file name is saved as "CSETx.cst" where x is the user number assigned to <char>, and .cst specifies a Cal Set and instrument state. This is not the same syntax as a file saved through the default choices from the front panel, which is "at00x.cst". For more information on the file naming syntax, see the [MMEMory](#) subsystem. Learn more about [Instrument/Cal States](#).

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <char> Choose from:  
USER01  
USER02...  
USER10

#### Examples

```
SENS:CORR:CSET:SAVE USER03  
sense2:correction:cset:save user09
```

#### Query Syntax

SENSe<cnum>:CORRection:CSET:SAVE?  
Queries the last correction set saved.

#### Return Type

Character

#### Default

Not applicable

---

**SENSe<cnum>:CORRection:CSET:STANdard <string>,<data>**

**(Read-Write)** Sets or returns standard data. Standard data is available for Unguided Cals ONLY.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <string> (String) Cal standard to read or write. The standard is specified using the EXACT case-sensitive string displayed in the [Cal Set Viewer](#) utility. See [SENS:CORR:CSET:DATA](#) for a description of port numbers.
- <data> (Block). Acquisition data. A Real / Imaginary data pair for each data point.
- Format is set using [FORM:DATA](#) command.

For REAL binary formats, refer to [Getting Data from the Analyzer using SCPI](#)

**Examples**

```
SENS:CORR:CSET:STAN 'S11C(1,1), 0.237,-1.422, 0.513, 0.895 '
```

Set acquisition data for two points.

```
SENS:CORR:CSET:STAN? "S11C(1,1)" 'Read data
```

**Query Syntax** SENSE<num>:CORRection:CSET:STANdard? (string)

**Return Type** Block data

**Default** Not Applicable

---

**SENSe<ch>:CORRection:CSET:STIMulus? [num]**

**(Read-only)** Returns the source or response stimulus values for the Cal Set that is currently used by channel <ch>. Values are returned in the format specified by [FORM:DATA](#) (Block or ASCII).

**Parameters**

- <ch> Channel number to query Cal Set stimulus values. If unspecified, value is set to 1
- [num] Optional argument. Range of frequencies to return. These values would be different when FOM (Opt 080) is enabled.
- 0** - returns source frequencies. Default setting if not specified.
- 1** - returns response frequencies.
- 2** - returns primary frequencies.

**Examples**

```
SENS:CORR:CSET:STIM?
```

```
sense:correction:cset:stimulus 1
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe:CORRection:CSET:TSET:ALLPorts? <cset>

(Read-only) Reads the port mapping used for the specified Cal Set. The returned values are the physical ports. The POSITION of the returned values corresponds to the logical ports.

For example, with an N44xx test set, if the returned string is "PNA 1,TS 2,PNA 2, TS 4" this means:

- PNA 1 is assigned to logical port 1
- TS 2 is assigned to logical port 2
- PNA 2 is assigned to logical port 3
- TS 4 is assigned to logical port 4

#### Parameters

<cset> **(String)** Name or GUID of the Cal Set. Use [SENS:CORR:CSET:CAT?](#) to read the list of available Cal Set names or GUIDs.

#### Examples

```
SENS:CORR:CSET:TSET:ALLP? "MyCalSet "  
sens:correction:cset:tset:allports? "{2B893E7A-971A-11d5-8D6C-  
00108334AE96}"
```

**Return Type** String

Default Not Applicable

---

### SENSe:CORRection:CSET:TSET:TYPE? <cset>

(Read-only) Reads the test set type (model) used for the specified Cal Set.

#### Parameters

<cset> **(String)** Name or GUID of the Cal Set. Use [SENS:CORR:CSET:CAT?](#) to read the list of available Cal Set names or GUIDs.

#### Examples

```
SENS:CORR:CSET:TSET:TYPE? "MyCalSet "  
  
'returns "N44xx"  
  
sens:correction:cset:tset:type? "{2B893E7A-971A-11d5-8D6C-  
00108334AE96}"
```

**Return Type** String

Default Not Applicable

---

## SENSe<ch>:CORRection:CSET:TYPE:CATalog? [format]

(Read-only) Query the Cal Types available in the selected Cal Set. The output is a comma separated list of Guids or a Cal Type names. [Learn more about applying Cal Types using SCPI.](#)

Use [CALC:CORR:TYPE](#) to apply a Cal Type.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1  
[format] (Optional) Format of the output of cal types. choose from:

**NAME** - (default) returns a list of cal type string names.

**GUID** - returns a list of cal type GUIDs

### Examples

```
SENS:CORR:CSET:TYPE:CAT? NAME  
SENS2:CORRection:CSET:TYPE:CAT?
```

**Return Type** String

**Default** Not Applicable

---

Last modified:

19-Nov-2009	Added Flatten command
28-Sep-2009	Added <block> desc to cset:Stan command
23-Mar-2009	Added <block> desc to cset:data command
12-Nov-2008	Added Stimulus? (8.33)
6-Mar-2008	Added CSET Delete by Name (8.0)
5-Mar-2008	Added Noise ETerm and Std commands (8.0)
9/12/06	Added TSET commands for multiport.



## Sense:Correction:Extension Commands

---

Performs and applies Port Extensions.

### **SENSe:CORRection:EXTension:**

#### **AUTO**

- | **CONFig**
- | **DCOFset**
- | **LOSS**
- | **MEAS**
- | **PORT**
- | **RESet**
- | **STARt**
- | **STOP**

#### **PORT**

- | **DISTance**
- | **FREQ**
- | **INCLude**
  - | **[STATe]**
- | **LDC**
- | **LOSS**
- | **MEDium**
- | **SYSVelocity**
- | **SYSMedia**
- | **[TIME]**
- | **UNIT**

| **VELFactor**

| **WGCutoff**

**RECeiver**

| **[TIME]**

**[STATe]**

Click on a [blue](#) keyword to view the command details.

#### See Also

- [Example Programs](#)
- [Learn about Port Extensions](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## **SENSe<cnum>:CORRection:EXTension:AUTO:CONFig <char>**

**(Read-Write)** Sets the frequencies used to calculate Automatic Port Extension. [Learn more about calculating Automatic Port Extension.](#)

### **Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<char> Frequencies to be used:

**CSPN** Use current frequency span.

**AMKR** - Use active marker frequency.

**USPN** - Use custom user span. Use [SENS:CORR:EXT:AUTO:STAR](#) and [SENS:CORR:EXT:AUTO:STOP](#) to specify start and stop frequency.

### **Examples**

```
SENS:CORR:EXT:AUTO:CONF CSPN
sense2:correction:extension:auto:config amkr
```

**Query Syntax** SENSe<cnum>:CORRection:EXTension:AUTO:CONFig ?

**Return Type** Character

**Default** CSPN

---

### SENSe<num>:CORRection:EXTension:AUTO:DCOFFset <bool>

(Read-Write) Specifies whether or not to include DC Offset as part of automatic port extension. [Learn more about Automatic DC Offset](#). Only allowed when [SENS:CORR:EXT:AUTO:LOSS](#) is set to ON.

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<bool> ON (or 1) - Includes DC Offset correction.

OFF (or 0) - Does NOT include DC Offset correction.

#### Examples

```
SENS:CORR:EXT:AUTO:DCOF 1
sense2:correction:extension:auto:dcoffset off
```

**Query Syntax** SENSe<num>:CORRection:EXTension:AUTO:DCOFFset?

**Return Type** Boolean

**Default** OFF (0)

---

### SENSe<num>:CORRection:EXTension:AUTO:LOSS <bool>

(Read-Write) Specifies whether or not to include loss correction as part of automatic port extension. [Learn more about Loss Compensation](#) in port extension.

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<bool> ON (or 1) - Includes Loss correction.

OFF (or 0) - Does NOT include Loss correction.

#### Examples

```
SENS:CORR:EXT:AUTO:LOSS 1
sense2:correction:extension:auto:loss off
```

**Query Syntax** SENSe<num>:CORRection:EXTension:AUTO:LOSS?

**Return Type** Boolean

**Default** OFF (0)

---

### SENSe<num>:CORRection:EXTension:AUTO:MEASure <char>

(Write-only) Measures either an OPEN or SHORT standard. When this command is sent, the PNA acquires the measurement with which to set automatic port extensions. [Learn more about which standard to measure.](#)

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<char> Standard to be measured. Choose from:

**OPEN** Measure OPEN standard

**SHORT** Measure SHORT standard

**Examples**

```
SENS:CORR:EXT:AUTO:MEAS OPEN  
sense2:correction:extension:auto:measure short
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## **SENSe<num>:CORRection:EXTension:AUTO:PORT<n> <bool>**

(Read-Write) Enables and disables automatic port extensions on the specified port.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<n> PNA Port number to enable or disable for automatic port extensions.

<bool> ON (or 1) - Enable

OFF (or 0) - Disable

**Examples**

```
SENS:CORR:EXT:AUTO:PORT2 0  
sense2:correction:extension:auto:port4 on
```

**Query Syntax** SENSe<num>:CORRection:EXTension:AUTO:PORT<n>?

**Return Type** Boolean

**Default** All ports ON (enabled)

---

## **SENSe<num>:CORRection:EXTension:AUTO:RESet**

**(Write-only)** Clears old port extension delay and loss data in preparation for acquiring new data. Send this command prior to sending a new series of [SENS:CORR:EXT:AUTO:MEAS](#). If acquiring both OPEN and SHORT standards, do not send this command between those acquisitions.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:CORR:EXT:AUTO:RES  
sense2:correction:extension:auto:reset
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe<num>:CORRection:EXTension:AUTO:STARt <value>**

**(Read-Write)** Set the start frequency for custom user span. [Learn more about User Span.](#)

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<value> User span start value. Must be within the frequency range of the active channel and less than the value set by SENS:CORR:EXT:AUTO:STOP.

**Examples**

```
SENS:CORR:EXT:AUTO:STAR 1E9  
sense2:correction:extension:auto:start 200e6
```

**Query Syntax** SENSe<num>:CORRection:EXTension:AUTO:STARt <value>?

**Return Type** Numeric

**Default** Start frequency of the current active channel.

---

**SENSe<num>:CORRection:EXTension:AUTO:STOP <value>**

**(Read-Write)** Set the stop frequency for custom user span. [Learn more about User Span.](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <value> User span stop value. Must be within the frequency range of the active channel and greater than the value set by SENS:CORR:EXT:AUTO:START

**Examples**

```
SENS:CORR:EXT:AUTO:STOP 1E9  
sense2:correction:extension:auto:stop 200e6
```

**Query Syntax** SENSE<num>:CORRection:EXTension:AUTO:STOP <value>?

**Return Type** Numeric

**Default** Stop frequency of the current active channel.

---

**SENSe<num>:CORRection:EXTension:PORT<pnum>:DISTance <value>**

**(Read-Write)** Sets and returns the port extension delay in physical length (distance).

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1.
- <pnum> Port Number that will receive the delay setting. If unspecified, value is set to 1.
- <value> Physical length of fixture of added transmission line. First specify units with [SENS:CORR:EXT:PORT:UNIT](#).

**Examples**

```
SENS:CORR:EXT:PORT1:DIST 12  
sense2:correction:extension:port2:distance .003
```

**Query Syntax** SENSe<num>:CORRection:EXTension:PORT<pnum>:DISTance?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:CORRection:EXTension:PORT<pnum>:FREQuency<n> <value>**

**(Read-Write)** Sets and returns the frequency "Use" number (1|2) for the specified port number.

[Learn about Loss Compensation values.](#)

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number that will receive the freq/loss settings. If unspecified, value is set to 1.
- <n> Frequency "Use" number. Choose from 1 or 2. If unspecified, value is set to 1.
- <value> Frequency value. Choose a frequency within the frequency span of the PNA.

#### Examples

```
SENS:CORR:EXT:PORT1:FREQ1 10E9  
sense2:correction:extension:port2:freq2:2E10
```

**Query Syntax** SENSE<cnum>:CORRection:EXTension:PORT<pnum>:FREQuency<n>?

**Return Type** Numeric

**Default** 1 GHz

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>:INCLude<n>:STATe <bool>

**(Read-Write)** Sets and returns the ON/OFF state for the use of the [Loss](#) and [Freq](#) values for the specified port number.

[Learn about Loss Compensation values.](#)

**Note:** This command affects ALL measurements on the specified channel.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number that will receive the Freq/Loss settings. If unspecified, value is set to 1.
- <n> Freq and Loss pair. Choose from 1 or 2. If unspecified, value is set to 1.
- <value> State of Freq and Loss values for port extension.

**0 or OFF** Specified Freq and Loss values are OFF

**1 or ON** Specified Freq and Loss values are ON

#### Examples

```
SENS:CORR:EXT:PORT:INCL 0  
sense2:correction:extension:port2:include:state on
```

**Query Syntax** SENSe<cnum>:CORRection:EXTension:PORT<pnum>:INCLude:STATe?

**Return Type** Boolean

**Default** OFF

---

### **SENSe<cnum>:CORRection:EXTension:PORT<pnum>:LDC <value>**

**(Read-Write)** Sets and returns the Port Loss at DC value for the specified port number.

[Learn about Loss Compensation values.](#)

**Note:** This command affects ALL measurements on the specified channel.

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port number to receive Loss value. If unspecified, value is set to 1.
- <value> Loss in dB. Choose a value between -90 and 90

#### **Examples**

```
SENS:CORR:EXT:PORT:LDC 1.5  
sense2:correction:extension:port2:ldc .1
```

**Query Syntax** SENSe<cnum>:CORRection:EXTension:PORT<pnum>:LDC?

**Return Type** Numeric

**Default** 0

---

### **SENSe<cnum>:CORRection:EXTension:PORT<pnum>:LOSS<n> <value>**

**(Read-Write)** Sets and returns the Loss value for the specified port number.

[Learn about Loss Compensation values.](#)

**Note:** This command affects ALL measurements on the specified channel.

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number that will receive the Freq/Loss settings. If unspecified, value is set to 1.
- <n> Loss "Use" number. Choose from 1 or 2. If unspecified, value is set to 1.
- <value> Loss in dB. Choose a value between -90 and 90

#### **Examples**

```
SENS:CORR:EXT:PORT:LOSS1 1  
sense2:correction:extension:port2:loss2 .1
```



<b>Query Syntax</b>	SENSe<cnum>:CORRection:EXTension:PORT<pnum>:LOSS<n>?
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	0

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>:MEDium <char>

**(Read-Write)** Sets and returns the media type of the added fixture or transmission line.

See also [SENS:CORR:EXT:PORT:SYSMedia](#)

**Note:** This command affects ALL measurements on the specified channel.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number for which media type is being set. If unspecified, value is set to 1.
- <char> Medium type. Choose from:
  - COAX
  - WAVEguide

#### Examples

```
SENS:CORR:EXT:PORT:MED COAX
sense2:correction:extension:port2:medium waveguide
```

<b>Query Syntax</b>	SENSe<cnum>:CORRection:EXTension:PORT<pnum>:MEDium?
<b>Return Type</b>	Character
<b><u>Default</u></b>	COAX

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>:SYSMedia <bool>

**(Read-Write)** Sets and returns the state of coupling with the system Media type. [Learn more.](#)

**Note:** This command potentially affects ALL measurements on the PNA.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number for which system Velocity Factor coupling is being set. If unspecified, value is set to 1.
- <bool> Coupling state. Choose from:
- **ON** (or 1) - Media type is coupled with the system setting.
  - **OFF** (or 0) - Media type is NOT coupled with the system setting.

#### Examples

```
SENS:CORR:EXT:PORT:SYSM 1  
  
sense2:correction:extension:port2:sysmedia off
```

**Query Syntax** SENSE<num>:CORRection:EXTension:PORT<pnum>:SYSMedia?

**Return Type** Boolean

**Default** 1 or ON (Coupled)

---

### SENSe<num>:CORRection:EXTension:PORT<pnum>:SYSVelocity <bool>

**(Read-Write)** Sets and returns the state of coupling with the system Velocity Factor value. [Learn more.](#)

**Note:** This command potentially affects ALL measurements on the PNA.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number for which system Velocity Factor coupling is being set. If unspecified, value is set to 1.
- <bool> Coupling state. Choose from:
- **ON** (or 1) - Velocity Factor is coupled with the system setting.
  - **OFF** (or 0) - Velocity Factor is NOT coupled with the system setting.

#### Examples

```
SENS:CORR:EXT:PORT:SYSV 1  
  
sense2:correction:extension:port2:sysvelocity off
```

<b>Query Syntax</b>	SENSe<cnum>:CORRection:EXTension:PORT<pnum>:SYSVelocity?
<b>Return Type</b>	Boolean
<b><u>Default</u></b>	1 or ON (Coupled)

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>[:TIME] <num>

(Read-Write) Sets the extension delay value in time at the specified port. Must also set [SENS:CORR:EXT ON](#).

**Note:** This command affects ALL measurements on the specified channel.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pnum> Port Number that will receive the extension. If unspecified, value is set to 1.
- <num> The port extension in seconds; may include suffix. Choose a number between: -1E18 and 1E18

#### Examples

```
SENS:CORR:EXT:PORT 2MS
sense2:correction:extension:port2 .00025
```

<b>Query Syntax</b>	SENSe<cnum>:CORRection:EXTension:PORT<pnum>[:TIME]?
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	0

---

### SENSe<cnum>:CORRection:EXTension:PORT:UNIT <char>

(Read-Write) Sets and returns the units for specifying port extension delay in physical length (distance).

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <char> Units for delay in distance. Choose from:
  - METer
  - FEET
  - INCH

#### Examples

```
SENS:CORR:EXT:PORT:UNIT MET
sense2:correction:extension:port:unit inch
```

---

**Query Syntax**    SENSE<cnum>:CORRection:EXTension:PORT:UNIT?

**Return Type**    Character

**Default**    METer

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>:VELFactor <value>

**(Read-Write)** Sets and returns the velocity factor of the fixture or added transmission line.

#### Parameters

<cnum>    Any existing channel number. If unspecified, value is set to 1

<pnum>    Port Number for which velocity factor is being set. If unspecified, value is set to 1.

<value>    Velocity Factor.

Set [SENS:CORR:EXT:PORT:SYSV](#) to use the system velocity factor.

#### Examples

```
SENS:CORR:EXT:PORT:VELF .6
```

```
sense2:correction:extension:port2:velfactor 1
```

**Query Syntax**    SENSE<cnum>:CORRection:EXTension:PORT<pnum>:VELFactor?

**Return Type**    Numeric

**Default**    System Velocity Factor

---

### SENSe<cnum>:CORRection:EXTension:PORT<pnum>:WGCutoff <value>

**(Read-Write)** Sets and returns the cutoff (minimum) frequency of the added waveguide fixture or transmission line.

#### Parameters

<cnum>    Any existing channel number. If unspecified, value is set to 1

<pnum>    Port Number for which media type is being set. If unspecified, value is set to 1.

<value>    Cutoff frequency in Hz.

This value is ignored when [SENS:CORR:EXT:PORT:MED](#) is set to **COAX** for the same port.

#### Examples

```
SENS:CORR:EXT:PORT:WGC 1e8
```

```
sense2:correction:extension:port2:wgcutoff 100Mhz
```

**Query Syntax**    SENSE<cnum>:CORRection:EXTension:PORT<pnum>:WGCutoff?

**Return Type** Numeric

**Default** System Media Cutoff Frequency

---

## **SENSe<cnum>:CORRection:EXTension:RECeiver<Rnum>[:TIME] <num> OBSOLETE**

(Read-Write) This command has NO replacement and no longer works.

Sets the extension value at the specified receiver. Must also set [SENS:CORR:EXT ON](#).

**Note:** Before using this command you must select a measurement using [CALC:PAR:SEL](#). You can select one measurement for each channel.

### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <Rnum> Number of the receiver that will receive the extension. If unspecified, value is set to 1  
Choose from:
- 1 for Receiver A
  - 2 for Receiver B
- <num> The electrical length in seconds; may include suffix. Choose a number between:  
**-10 and 10**

### **Examples**

```
SENS:CORR:EXT:REC 2MS  
sense2:correction:extension:receiver2:time .00025
```

**Query Syntax** SENSe<cnum>:CORRection:EXTension:RECeiver<Rnum>[:TIME]?

**Return Type** Numeric

**Default** 0

---

## **SENSe<cnum>:CORRection:EXTension[:STATe] <ON | OFF>**

**(Read-Write)** Turns port extensions ON or OFF.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns port extensions ON.

**OFF** (or 0) - turns port extensions is OFF.

**Examples**

```
SENS:CORR:EXT ON  
sense2:correction:extension:state off
```

**Query Syntax** SENSE<num>:CORRection:EXTension[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

Last Modified:

28-Sep-2009 Fixed Port:Frequency command syntax  
6-Feb-2009 Added new commands (A.08.50)

## Sense:Correction:Collect:Guided Commands

---

Performs and applies a SmartCal (Guided) calibration and other error correction features.

### SENSe:CORRection:COLLect:GUIDed:

**ABORt**

**ACQuire**

**ADAPter**

| **COUNt?**

| **CREate?**

| **DELay**

| **DESCription**

| **PATHs**

**CKIT**

| **CATalog?**

| **PORT**

| **CATalog?**

| **[SElect]**

**CONNector**

| **CATalog?**

| **PORT**

| **[SElect]**

**DESCription**

**DMATch**

| **APPLy**

| **[IMMediate]**

| **PORTs?**

| **[INITiate]**

**ETERms:LOAD**

| **[CSET]**

**INITiate**

**ISOLation**

| **AVERage**

| **INCRement**

| **PATHs**

**METHod**

**PACQuire**

**PATH**

| **CMETHod**

| **TMETHod**

**PSENsor**

| [CKIT](#)  
| [CONNector](#)  
| [POWer](#)  
| [LEVel](#)  
[SAVE](#)  
| [CSET](#)  
[STEPS?](#)  
[THRU](#)

Click on a [blue](#) keyword to view the command details.

[Red](#) keywords are superseded commands.

#### See Also

[Examples](#) using these commands.

New [Calibrating the PNA Using SCPI](#)

[Learn about Measurement Calibration](#)

[Synchronizing the PNA and Controller](#)

---

## SENSe:CORRection:COLLect:GUIDed:ABORt

**(Write-only)** Aborts the acquiring of a guided calibration that has been [INITialized](#) but has not yet been concluded using the [SAVE](#) command. If at least one Cal standard has already been measured, and the [Calibration Window](#) is being displayed, this command also closes the Calibration Window and re-tiles the other measurement windows.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

#### Examples

```
SENS:CORR:COLL:GUID:ABOR  
sense2:correction:collect:guided:abort
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe:<cnum>CORRection:COLLect:GUIDed:ACQuire <std>[,sync]**



**(Write-only)** Initiates the measurement of the specified calibration standard. Executing this command with an unnecessary standard has no affect.

The measured data is stored and used for subsequent calculations of error correction coefficients. All standards must be measured before a calibration can be completed. Any measurement can be repeated until the [SENS:CORR:COLL:GUID:SAVE](#) command is executed.

Query the user prompt description using [SENS:CORR:COLL:GUID:DESC?](#)

Query the required calibration steps using [SENS:CORR:COLL:GUID:STEP?](#)

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<std> Choose from:STAN1, STAN2, STAN3, through STAN40

[sync] Optional argument. Choose from:

**SYNChronous** - blocks SCPI commands during standard measurement (default behavior).

**ASYNchronous** - does NOT block SCPI commands during standard measurement.

[Learn more.](#)

#### Examples

```
SENS:CORR:COLL:GUID:ACQ STAN1
sense2:correction:collect:guided:acquire stan1
```

Query Syntax Not Applicable

Default Not Applicable

---

**SENSe<cnum>:CORRection:COLLect:GUIDed:ADAPter:CREate? <conn1>, <conn2>**

**(Read-only)** Specifies the use of a THRU adapter to be used during the Guided Cal Unknown THRU and Adapter Removal Cal. Returns an adapter index <n> which is used to refer to the adapter in several related commands. [See Cal Thru Methods](#). While the choice of which end of the adapter is <conn1> and <conn2> is arbitrary, it is necessary to remember which will be used on each test port.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <conn1> Adapter port 1 connector type. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to return a list of valid connector types.
- <conn2> Adapter port 2 connector type.

**Examples** [See example using this command.](#)

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<num>:CORRection:COLLect:GUIDed:ADAPter:COUNt?**

**(Read-Only)** Returns the number of THRU adapters that have been created for this calibration using [SENS:CORR:COLL:GUID:ADAP:CREate?](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1

**Examples** [See example using this command.](#)

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<num>:CORRection:COLLect:GUIDed:ADAPter<n>:DELay <coax>, [w phase, wdelay]**

**(Write-only)** Specifies the adapter delay. and optionally waveguide delay and optional phase offset (degrees) of adapter <n>.

**Parameters**

- <cnun> Any existing channel number. If unspecified, value is set to 1
- <n> Adapter index number that was returned from [SENS:CORR:COLL:GUID:ADAP:CREate?](#)
- <coax> Delay value of coax adapter <n> in seconds. If the adapter has no coax connector, enter 0.
- <wphase> Waveguide phase offset in degrees. If the adapter has no waveguide connector, do not enter a value.
- <wdelay> Waveguide delay in seconds. If the adapter has no waveguide connector, do not enter a value.

**Examples** [See example using this command.](#)

**Default** Not Applicable

---

**SENSe<cnun>:CORRection:COLLect:GUIDed:ADAPter<n>:DESCription <string>**

**(Write-only)** Specifies the adapter description for use as the guided cal connection prompts.

**Parameters**

- <cnun> Any existing channel number. If unspecified, value is set to 1
- <n> Adapter index number that was returned from [SENS:CORR:COLL:GUID:ADAP:CREate?](#)
- <string> Adapter description.

**Examples** [See example using this command.](#)

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SENSe<cnun>:CORRection:COLLect:GUIDed:ADAPter<n>PATHs <port pairs>**

**(Write-only)** Specifies the port pairs for which the adapter will be used for a THRU connection.

For example, for a 3-port cal on channel 1 using ports 1,2,and 3), to use adapter 1 between the ports (1 to 2) and (1 to 3) the following command is used: SENS1:CORR:COLL:GUID:ADAP1:PATH 1,2,1,3.

The adapter must have the same DUT connectors as the ports that are already specified for these ports.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <n> Adapter index number that was returned from [SENS:CORR:COLL:GUID:ADAP:CREate?](#)
- <port pair> Ports for which the adapter will be used. The orientation is not critical, as the PNA will align the connector types as necessary. The minimum number of Thru connections required is the number of ports to calibrated -1.

**Examples** [See example using this command.](#)

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:CKIT:CATalog? <connector>

**(Read-only)** This command replaces [SENS:CORR:COLL:GUID:CKIT:PORT:CAT?](#)

Returns a comma-separated list of valid kits that use the specified connector type. This includes mechanical cal kits, applicable characterizations found within ECal modules currently connected to the PNA, **and all user characterizations stored in PNA disk memory**. For ECal modules, the returned list includes the serial numbers. [See ECal User Characterization commands.](#)

Use items in the list to select the kit to be used with the [SENS:CORR:COLL:GUID:CKIT:PORT](#) command.

#### Parameters

- <conn> String. Connector type. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to return a list of valid connector types.

**Examples** `SENSe:CORR:COLL:GUID:CKIT:CAT? "Type N (50) male"`

**Return Type** String

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:CKIT:PORT<pnum>:CATalog? **Superseded**

(Read-only) This command is replaced by [SENSe:CORR:COLL:GUID:CKIT:CAT?](#).

Returns a comma-separated list of valid kits for the specified PNA port. In addition to mechanical calibration kits, this will include applicable characterizations found within ECal modules currently connected to the PNA.

Use items in the list to select the kit to be used with the [SENS:CORR:COLL:GUID:CKIT:PORT](#) command.

**Note:** Beginning with PNA Rev 9.1, the serial number is returned for ALL ECal modules that are connected with the connector type of the specified port. Previously, the returned list would include the serial numbers to distinguish the ECal modules only when two or more identical ECal models were connected to the PNA.

### Parameters

<pnum> Any existing port number. If unspecified, value is set to 1

### Examples

```
SENS:CORR:COLL:GUID:CKIT:PORT1:CAT?
'When "Type N (50) male" is specified for connector type,
returns:

"85054D, 85032F"

'When two identical ECal modules are connected for the connector
type,
'the return string includes serial numbers

"85092-60010 ECal 10675, 85092-60010 ECal 00758"
```

**Return Type** String

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:GUIDed:CKIT:PORT<pnum>[:SElect] <kit>**

**(Read-Write)** Specifies the calibration kit for each port to be used during a guided calibration. An unused port does NOT need to have a specified Cal Kit.

**Note:** Sliding loads are not fully supported from Sens:Corr:Coll:Guided... The **Measure** button must be manually pressed.

1. Specify the connector type for the port with [SENS:CORR:COLL:GUID:CONN:PORT](#).
2. Query the valid available kits for the connector on each port with [SENS:CORR:COLL:GUID:CKIT:PORT:CAT?](#)
3. Specify the kit using this command.
4. Perform a query of this command. If the <kit> parameter was incorrectly entered, an error will be returned.

#### Parameters

- <pnum> Any existing port number. If unspecified, value is set to 1
- <kit> Calibration kit to be used for the specified port.

When using an ECal module, include the characterization name in the <kit> string. Use [SENSe:CORR:COLL:GUID:CKIT:CAT?](#) to read the list of characterizations available in the module and in PNA disk memory.

If two or more identical ECal modules are connected to the PNA, the serial number must be included to distinguish the ECal modules.

#### Examples

```
SENS:CORR:COLL:GUID:CKIT:PORT1 '85055A'
```

```
'The following includes a serial number when two or more ECal  
mods are connected  
sense:correction:collect:guided:ckit:port2:select '85092-60010  
ECal 10685'
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:CKIT:PORT<pnum>[:SElect]?

**Return Type** String - If the <kit> parameter was incorrectly entered while writing, an error will be returned.

**Default** Not Applicable

---

[SENSe:CORRection:COLLect:GUIDed:CONNector:CATalog?](#)

**(Read only)** Returns a list of valid connectors based on the connector descriptions of the available cal kits. Use an item from the returned list to specify a connector for

[SENS:CORR:COLL:GUID:CONN:PORT](#)

**Examples**

```
SENS:CORR:COLL:GUID:CONN:CAT?
```

**Returns:**

```
Type N (50) female, Type N (50) male, APC 7 (50), 3.5 mm (50) male, 3.5 mm (50) female, User Connector A
```

**Return Type** Comma separated string values

**Default** Not Applicable

---

## **SENSe:CORRection:COLLect:GUIDed:CONNector:PORT<pnum>[:SElect] <conn>**

**(Read-Write)** Specifies a connector type for every port during the Guided Calibration procedure. Valid connector names are stored within calibration kits. Some cal kits may include both male and female connectors. Therefore, specifying connector gender may be required.

The PNA remembers previous Guided Cal settings. Therefore, for completeness, unused ports should be defined as "Not used". See [Guided Cal examples](#).

- A single port with a valid <conn> name indicates a 1-Port calibration will be performed.
- Two ports with valid <conn> names indicate either a 2-Port SOLT or [TRL](#) calibration will be performed depending on the standards definition found within the cal kit and the capability of the PNA.
- Three ports with valid <conn> names indicate a 3-Port calibration will be performed, and so forth.

### **Follow these steps to ensure port connectors are specified correctly:**

1. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to query available connectors before specifying the port connector.
2. Set a connector type for each port using this command.
3. Perform a query of this command. If the connector type was incorrectly entered, an error will be returned.
4. Specify the cal kit to use for each port with [SENS:CORR:COLL:GUID:CKIT:PORT](#)

### **Parameters**

<pnum> Any existing port number. If unspecified, value is set to 1

<conn> String - DUT connector type to connect with PNA port <pnum>

**Examples**

```
'Specifying a 2-port cal (1 & 2) on a 4-port PNA
```

```
SENS:CORR:COLL:GUID:CONN:PORT1 'Type N (50) female'
SENS:CORR:COLL:GUID:CONN:PORT2 'Type N (50) male'
SENS:CORR:COLL:GUID:CONN:PORT3 'Not Used'
SENS:CORR:COLL:GUID:CONN:PORT4 'Not Used'
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:CONNector:PORT<pnum>[:SElect]?

**Return Type** String

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:DESCription? <step>

(Read-only) Returns the connection description for the specified calibration step.

#### Parameters

<step> A number from 1 to the number of steps required to complete the calibration (Use [SENS:CORR:COLL:GUID:STEP?](#) to query the number of steps )

#### Examples

```
SENS:CORR:COLL:GUID:DESC? 10
```

```
'Returns:
Connect APC 7 Open to port3
```

**Return Type** String

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:DMATch:APPLy[:IMMediate] [<CalSetGUID>]

(Write-only) Specifies a Cal Set as a source of delta match correction.

If CalSetGUID is not specified, then the Global Delta Match Cal Set is assumed. An error is returned if the **specified** Cal Set does not meet the following Delta Match criteria. The Global Delta Match Cal can ALWAYS be applied.

- Must have been performed using ECal or as a guided mechanical cal (not Unguided).
- Must have the same start freq, stop freq, and number of points as the channel being calibrated.
- Must calibrate the ports that are required by the TRL or Unknown Thru cal as indicated by [SENS:CORR:COLL:GUID:DMATch:APPLy:PORTs?](#).

[Learn more about Delta match calibration.](#)

See example of a complete [Global Delta Match calibration.](#)

See example where [Delta Match is applied to a calibration.](#)



### Parameters

<CalSetGUID> Optional. GUID of the Cal Set to use. If unspecified, the Global Delta Match Cal Set is used.

### Examples

```
SENS:CORR:COLL:GUID:DMAT:APPL
sense:correction:collect:guided:dmatch:apply:immediate
"{2B893E7A-971A-11d5-8D6C-00108334AE96}"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:DMATch:APPLy:PORTs?

(Read-only) Returns the port numbers for which delta match correction is required. 0 (zero) is returned if the Cal does NOT require Delta Match correction for one of the following reasons:

- The Cal does NOT involve Unknown THRU or TRL. You specify this using [SENS:CORR:COLL:GUID:METH <UNKN | TRL>](#).
- The Cal DOES involve Unknown THRU or TRL, but the delta match data can be calculated by the Unknown Thru or TRL Cal. [Learn how this is possible](#). However, you can force the Cal to use the Delta Match data from a Cal Set.

[Learn more about Delta match calibration.](#)

See example of a complete [Delta Match calibration](#).

### Parameters

### Examples

```
SENS:CORR:COLL:GUID:APPL:PORT?

'Returns:
1, 2, 3
```

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:DMATch[:INITiate] <conn>,<cKit>

(Write-only) Initiates a global delta match calibration.

[Learn more about Global Delta Match calibration.](#)

See example of a complete [Delta Match calibration.](#)

#### Parameters

- <conn> **String**. Connector type for port 1. All other ports are set automatically.
- <cKit> **String** Cal Kit for all ports. If incorrectly entered while writing, an error is returned.

#### Examples

```
SENS:CORR:COLL:GUID:DMAT APC 3.5 female,"85052B"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:ETERms:LOAD[:CSET] <cset>,<calPort> [,csPort]

(Write-only) Loads 1-port error terms from a Cal Set into the current Guided Cal sequence. When the Cal steps are recomputed, connection steps are removed due to the loading of the error terms.

[See example of how to use this command.](#)

#### Parameters

- <cset> **String** Name of User Cal Set in which the error terms reside.
- <pnum> **Integer** Port number of the current cal to receive error terms.
- [csPort] **Integer** Optional argument. Port number associated with the error terms in the Cal Set. If unspecified, the same port number as <calPort> is used.

#### Examples

[See example](#)

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<cnum>:CORRection:COLLect:GUIDed:INITiate [cal set name or GUID [,bool]]

**(Write-only)** Initiates a guided calibration.

- The PNA determines the measurements needed to perform the calibration using the settings specified from the [SENS:CORR:COLL:GUID:CONN:PORT](#) and [SENS:CORR:COLL:GUID:CKIT:PORT](#) commands.
- After this command is executed, subsequent commands can be used to query the number of measurement steps, issue the acquisition commands, query the connection description strings, and subsequently complete a guided calibration.

### Parameters

<num> Channel to be calibrated. Must be the active channel. Use [Calc:Par:Sel](#) to activate any measurement on the channel.

<cal set name or GUID> **(String)** Optional argument.

If specified, choose an **existing** Cal Set, either by name or by GUID.

If NOT specified, behavior depends on the [SENS:CORR:PREFeRence:CSET:SAVE](#) setting.

- By Cal Set name: include quotes.
- By Calset GUID in the form: "{GUID}"; including quotes and curly brackets.
- Query all Cal Set GUIDs with [SENS:CORR:CSET:CAT?](#)

An error is reported if the Cal Set is not found.

The Cal Set is either supplemented or overwritten depending on the method, connectors, and ports selected. [Learn more about Cal Sets.](#)

<bool> Optional argument.

**OFF (0)** If Cal Set stimulus settings differ from the existing channel, do not change channel stimulus settings. The Cal Set is save to the current setting of the [SENS:CORR:PREF:CSET:SAVE](#) command. This is the default setting if not specified.

**ON (1)** If cal set stimulus settings differ from the existing channel, change the channel stimulus settings to match the Cal Set settings.

### Examples

```
SENS:CORR:COLL:GUID:INIT "{2B893E7A-971A-11d5-8D6C-00108334AE96}",1
sense2:correction:collect:guided:initiate 'myCalSet'
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:ISOLation:AVERage:INCRement <num>

**(Read-Write)** Specifies amount to increment (increase) the channels averaging factor during measurement of isolation standards in a guided calibration.

**Note:** If the channel has averaging turned OFF and the value of <num> is greater than 1, averaging will be turned ON only during the isolation measurements and with the averaging factor equal to <num>.

### Parameters

<num> Amount to increment the averaging factor for the isolation measurement. The maximum averaging factor for the channel is 65536 ( $2^{16}$ ).

### Examples

```
'Measure isolation on all paths for the cal  
SENS:CORR:COLL:GUID:ISOL ALL  
  
'Remove the port pairs 1-to-2 and 1-to-3 from the list of paths  
on which to measure isolation  
  
sense:correction:collect:guided:isolation:paths REMOVE,1,2,1,3
```

**Query Syntax** SENSe:CORRection:COLLect:GUIDed:ISOLation:AVERage:INCRement?

**Return Type** Numeric

**Default** 8 - If this command is NOT sent, but [isolation is measured](#), then averaging will be turned ON with factor set to 8 during the isolation measurements.

---

## SENSe:CORRection:COLLect:GUIDed:ISOLation[:PATHs] <char>[,<p1a, p1b, p2a, p2b,&&gt;]

**(Read-Write)** Specifies the paths (port pairs) to make isolation measurements on during a guided calibration.

### Parameters

<char> **ALL** Measure isolation on all pairings of the ports that are to be calibrated.

**NONE** Do not measure isolation on any pairing of the ports to be calibrated. (Default behavior).

**ADD** Add one or more specific pairings of ports to the list of port pairings for which isolation will be measured.

**REMOve** Remove one or more specific pairings of ports from the list of port pairings for which isolation will be measured. If many paths are to be measured, it may be easier to first send ALL, then REMOve and specify the

paths to remove.

<p1a, p2a...> For use when <char> is **ADD** or **REMove**.

Specify Port numbers in pairs:

- For 3-port cals, specify up to 3 pairs.
- For 4-port cals, specify up to 6 pairs.

p1a, p1b (Path1 - port A and port B)

p2a, p2b (Path2 - port A and port B)

p3a, p3b (Path3 - port A and port B)

#### Examples

```
'Measure isolation on all paths for the cal  
SENS:CORR:COLL:GUID:ISOL ALL  
  
'Remove the port pairs 1-to-2 and 1-to-3 from the list of paths  
on which to measure isolation  
  
sense:correction:collect:guided:isolation:paths REMove,1,2,1,3
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:ISOLation:PATHs?

**Note:** if isolation is not be measured on any of the paths, the query returns 0

**Return Type** Numeric

**Default** 0 - Isolation not measured on any paths.

---

**SENSe:CORRection:COLLect:GUIDed:METHod <char> Superseded**

This command is replaced with [SENS:CORR:COLL:GUID:PATH:CMETHod](#) and [SENS:CORR:COLL:GUID:PATH:TMETHod](#).

(Read-Write) Selects from one of several algorithms available for performing the THRU portion of a guided calibration. [Learn more about THRU methods](#).

#### Parameters

<char> **DEFAULT** - Informs guided calibrations to use the default algorithm when computing the number of needed standards acquisition steps. (default selection if omitted.)

**ADAP** - Use the adapter removal algorithm

**FLUSH** - Use with insertable devices.

**UNKN** - Use the Unknown THRU algorithm with calibrations for non-insertable devices.

**DEFined** - Use the THRU definition that you stored in the cal kit file, or ECal module.

**TRL** - Select TRL Cal Type for guided cals. Valid for "TRL ready" Cal Kits with properly assigned TRL cal classes.

**SOLT** - Select SOLT Cal Type for guided cals. Valid for any kit with properly assigned SOLT cal classes.

#### Examples

```
SENS:CORR:COLL:GUID:METH ADAP
sense:correction:collect:guided:method unkn
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:METHod?

**Return Type** Character

**Default** DEFAULT

---

**SENSe:CORRection:COLLect:GUIDed:PACQuire <std>**

(Read-Write) Show the [Cal Window](#), and optionally one or more other specific windows before acquiring a Cal standard. This command will cause the Cal Window to display the specific measurements that are to be made for that particular Cal standard.

**Parameters**

<std> Choose from:STAN1, STAN2, STAN3, through STAN40.

**Examples**

```
SENS:CORR:COLL:GUID:PACquire STAN2  
sense:correction:collect:guided:pacquire STAN5
```

[See an example that uses this command.](#)

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:PACQuire?

**Return Type** Character

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:PATH:CMETHod

<port1>,<port2>,<caltype1[,caltype2]>

**Note:** This command replaces [SENS:CORR:COLL:GUID:METH.](#)

(Read-Write) Specifies the calibration method for each port pair.

**Note:** Before sending this command, first do the following:

1. Set the connector types: [SENS:CORR:COLL:GUID:CONN:PORTn](#)
2. Set cal kit: [SENS:CORR:COLL:GUID:CKIT:PORT](#)
3. Set or query the thru path pairs: [SENS:CORR:COLL:GUID:THRU:PORT](#)
4. Set the Thru method: [SENS:CORR:COLL:GUID:PATH:TMETHod](#)
5. **Send this command** (SENS:CORR:COLL:GUID:PATH:CMET)

**After** sending this command, send the query form to be sure that the command was accepted. If not, then the chosen Cal method is not compatible with the specified Thru method. For example, if the specified Thru method is Unknown Thru, an attempt to set Enhanced Response Cal should be rejected.

[Learn more about Thru Methods.](#)

**Parameters**

<port1> First port of the pair to be calibrated.

<port2> Second port of the pair to be calibrated.

<caltype1[caltype2]> (String) Cal types for 1st and 2nd ports of the pair, enclosed in a single pair of quotes. NOT case-sensitive.

**caltype1** Cal type for the pair if caltype2 is not specified. Otherwise, Cal type for port 1. Choose from:

- TRL
- SOLT
- QSOLTN
- EnhRespN

For the last two arguments, replace **N** with the port to be used as the source port, which **MUST** be one of the port pair.

**caltype2** Optional argument. Use only when performing an adapter removal cal on the pair. This argument specifies the Cal type on the second port. Caltype1 then specifies the Cal type of the first port.

Choose from the same arguments as caltype1.

#### Examples

```
SENS:CORR:COLL:GUID:PATH:CMETHOD 2,3,"QSOLT2"
```

```
sense:correction:collect:guided:path:cmethod 2,3,"solt,trl"
```

#### Query Syntax

SENSe:CORRection:COLLect:GUIDed:PATH:CMETHOD? <port1>,<port2>

If only one caltype is returned then its NOT adapter removal.

#### Return Type

String

#### Default

Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:PATH:TMETHOD <port1>,<port2>,<thruType1[,thruType2]>

**Note:** This command replaces [SENS:CORR:COLL:GUID:METH.](#)

(Read-Write) Specifies the calibration **THRU** method for each port pair.

**Note:** Before sending this command, first do the following:

1. Set the connector types: [SENS:CORR:COLL:GUID:CONN:PORTn](#)
2. Set cal kit: [SENS:CORR:COLL:GUID:CKIT:PORT](#)



3. Set or query the thru path pairs: [SENS:CORR:COLL:GUID:THRU:PORT](#)
4. **Send this command** (SENS:CORR:COLL:GUID:PATH:TMETHod)

**After** sending this command, set and query the Cal Type:

[SENS:CORR:COLL:GUID:PATH:CMETHod](#)

[Learn more about Thru methods.](#)

### Parameters

- <port1> First port of the port pair to be calibrated.
- <port2> Second port of the port pair to be calibrated.
- <thruType1[,thruType2]> (String) Thru methods for 1st and 2nd ports of the pair, enclosed in a single pair of quotes. NOT case-sensitive.

**thruType1** Calibration thru method for the pair if thruType2 is not specified. Otherwise, thru method for port 1.

Choose from:

- **Defined Thru** A thru type for which there is a stored definition in the Cal Kit.
- **Zero Thru** Zero length thru, also known as flush-thru.
- **Undefined Thru** A thru type for which there is NOT a stored definition in the Cal Kit. Also known as Unknown Thru. Valid ONLY for SOLT cal type.
- **Undefined Thru using a Defined Thru** Using an ECal module, measure the internal thru using the "Undefined Thru" method.

**thruType2** Optional argument. Use only when performing an adapter removal cal on the pair as determined by [SENS:CORR:COLL:GUID:PATH:CMETHod](#). The only valid arguments for ThruType1&2 is "Defined Thru, Defined Thru".

### Examples

```
SENS:CORR:COLL:GUID:PATH:TMETHod 2,3,Zero Thru
sense:correction:collect:guided:path:tmethod 2,3,"Defined Thru,Defined Thru"
```

### Query Syntax

SENSe:CORRection:COLLect:GUIDed:PATH:TMETHod  
<port1>,<port2>?

Always returns two parts:

If the second part of the string is empty, adapter removal is NOT being performed.

If the string is "Defined Thru, Defined Thru", adapter removal IS being performed.

**Return Type** String

**Default** Not Applicable

---

### **SENSe:CORRection:COLLect:GUIDed:PSEnSor<pnum>:CKIT <kit>, [src]**

**(Read-Write)** Specifies the calibration kit to be used when an adapter is necessary to connect the power sensor to the port <pnum> during a guided calibration.

1. Specify the connector type for the adapter with [SENS:CORR:COLL:GUID:PSEN1:CONN](#)
2. Query the valid available kits for the connector with [SENS:CORR:COLL:GUID:CKIT:PORT:CAT?](#)
3. Specify the kit using this command.
4. Perform a query of this command. If the <kit> parameter was incorrectly entered, an error will be returned.

#### **Parameters**

<pnum> Any existing port number. If unspecified, value is set to 1. For NFX cals, port number must be 1.

<kit> Calibration kit to be used for the specified port.

When using an ECal module, include the characterization name in the <kit> string. Use [SENSe:CORR:COLL:GUID:CKIT:CAT?](#) to read the list of characterizations available in the module and in PNA disk memory.

If two or more identical ECal modules are connected to the PNA, the serial number must be included to distinguish the ECal modules.

[src] String - (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2".

Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

See [Remotely Specifying a Source Port](#)

#### Examples

```
SENS:CORR:COLL:GUID:PSEN1:CKIT '85055A'
```

'The following includes a serial number when two or more ECal mods are connected

```
sense:correction:collect:guided:psensor2:ckit '85092-60010 ECal 10685'
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:PSENSor<pnum>:CKIT?

**Return Type** String

**Default** Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:PSENSor<pnum>:CONNector <conn>, [src]

(Read-Write) Specifies the connector type for a power sensor when a power calibration is part of the Guided Calibration process. This can be done during [NFX](#) calibrations. Valid connector names are stored within calibration kits. Some cal kits may include both male and female connectors. Therefore, specifying connector gender may be required.

#### Follow these steps to ensure port connectors are specified correctly:

1. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to query available connectors before specifying the port connector.
2. Set a connector type for each port using this command.
3. Perform a query of this command. If the connector type was incorrectly entered, an error will be returned.
4. Specify the cal kit to use for each port with [SENS:CORR:COLL:GUID:PSENSor:CKIT](#)

#### Parameters

- <pnum> Any existing port number. If unspecified, value is set to 1. For NFX cals, port number must be 1.
- <conn> String - Power sensor connector type to connect to the specified PNA port <pnum>.
- [src] String - (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2".

Otherwise, the <port> argument performs the same function. If both arguments

are specified, [src] takes priority.

See [Remotely Specifying a Source Port](#)

**Examples**

```
SENS:CORR:COLL:GUID:PSEN1:CONN "3.5 mm (50) male"  
sense:correction:collect:guided:psensor1:connector "3.5 mm (50)  
female"
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:PSENsor<pnum>:CONNector?

**Return Type** String

**Default** Not Applicable

---

**SENSe:CORRection:COLLect:GUIDed:PSENsor<pnum>:POWer:LEVel <value>, [src]**

(Read-Write) Specifies the power level at which to perform a power calibration during a guided calibration.

This can be done during [NFX](#) calibrations.

**Parameters**

<pnum> Any existing port number. If unspecified, value is set to 1. For NFX cals, port number must be 1.

<value> Power level in dBm at which to perform the power calibration. [Learn more.](#)

[src] String - (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2".

Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

See [Remotely Specifying a Source Port](#)

**Examples**

```
SENS:CORR:COLL:GUID:PSEN1:POW:LEV 0  
sense:correction:collect:guided:psensor:power:level -5
```

**Query Syntax** SENSE:CORRection:COLLect:GUIDed:PSENsor<pnum>:POWer:LEVel?

**Return Type** Numeric

**Default** 0

---

## SENSe:CORRection:COLLect:GUIDed:SAVE [bool]

(Write-only) Completes the guided cal by computing the error correction terms, turning Correction ON, and saving the calibration to a Cal Set.

If all of the required standards have not been measured, the calibration will not complete properly.

### Parameters

[bool] Optional argument. If unspecified, the default behavior is the current PNA preference setting of [SENSe:CORRection:PREFeRence:CSET:SAVE](#)

**OFF (0)** Save cal data ONLY to a Cal Register.

**ON (1)** Save cal data to a Cal Register and a User Cal Set. The filename is automatically generated.

### Examples

```
SENS:CORR:COLL:GUID:SAVE  
sense2:correction:collect:guided:save 0
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SENSe:CORRection:COLLect:GUIDed:SAVE:CSET <cal set name or guid>

(Write-only) Completes the guided cal by computing the error correction terms, turning Correction ON, and saving the calibration to the specified Cal Set. This command performs the same function as [SENSe:CORRection:COLLect:GUIDed:SAVE](#), but this command allows the name or GUID of the Cal Set to be specified.

- Use this command instead of the optional name or GUID argument in [SENS:CORR:COLL:GUID:INIT](#).
- Use [SENS:CORRection:CSET](#) commands to get names or GUIDs of existing Cal Sets.
- The cal data is also saved to the channel Cal Register.
- If all of the required standards have not been measured, the calibration will not complete properly.

### Parameters

<cal set name or guid> **String** - Name or GUID of an existing Cal Set to be overwritten. If specifying a GUID, curly brackets must be included.

### Examples

```
SENS:CORR:COLL:GUID:SAVE:CSET "{2B893E7A-971A-11d5-8D6C-  
00108334AE96}"  
  
sense:correction:collect:guided:save:cset "MyCalSet"
```

**Query Syntax** Not Applicable

Default Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:STEPS?

(Read-only) Returns the number of measurement steps required to complete the current guided calibration. This command is sent after the [SENS:CORR:COLL:GUID:INIT](#), [SENS:CORR:COLL:GUID:CONN:PORT](#) and [SENS:CORR:COLL:GUID:CKIT:PORT](#) commands.

**Examples** `SENS:CORR:COLL:GUID:STEP?  
sense2:correction:collect:guided:steps?`

**Return Type** Numeric

Default Not Applicable

---

### SENSe:CORRection:COLLect:GUIDed:THRU:PORTs <t1a, t1b, t2a, t2b, t3a, t3b...>

(Read-Write) For calibrating more than 2-ports ONLY. Specifies the port pairs for the thru connections of the calibration.

Use the query to learn the port pairs that the PNA suggests.

Use the set command to override these thru pairs.

#### Parameters

<t1a,...> Specify Port numbers in pairs:

- For 3-port cals, specify two or three pairs.
- For 4-port cals, specify from three up to six pairs.

t1a, t1b (Thru1 - port A and port B)

t2a, t2b (Thru2 - port A and port B)

t3a, t3b (Thru3 - port A and port B)

...and so forth up to six pairs.

**Examples** `SENS:CORR:COLL:GUID:THRU:PORT 1,2,1,3,1,4 '4-port measurement  
sense:correction:collect:guided:thru:ports 1,2,2,3 '3-port  
measurement`

**Query Syntax** SENSe:CORRection:COLLect:GUIDed:THRU:PORTs?

**Return Type** Numeric

Default Port pairings that were used in the previous cal.

---

Last modified:

16-Nov-2009	Added Note to GUID:CKIT:PORT<num>:CAT?
24-Aug-2009	Added ckitCat (9.0)
17-Jul-2009	Fixed order in which CMET and TMET are set.
27-Mar-2009	Fixed Isolation:Incr query
24-Feb-2009	Replaced True/False
13-Feb-2009	Added sync to ACQuire
26-Sep-2008	Added link to apply Delta Match example
10-Mar-2008	Added Abort command (A.07.50.27)
1-Nov-2007	Added PAcquire command
14-Apr-2007	Added Cal Set by name and isolation commands
8-Mar-2007	Added CMethod and TMethod
23-Oct-2006	Fixed wording in Conn:Port:Sel

## Sense:Correction:IMD Commands

---

Controls an IMD and IMDx calibration. These commands supplement the [Guided Cal commands](#).

### SENSe:CORRection:IMD

| CALibration

| FREQuencies

| METHod

| LO

| PCAL

| MPProduct

| POWer

| SENSor

| CKIT

| CONNector

| SORDer:INCLude

Click on a [blue](#) keyword to view the command details.

### Other IMD (Opt 087) commands

- [CALC:CUSTom:DEFine](#) - creates an IMD measurement.
- [Swept IMD](#)
- [IMSpectrum](#)

### See Also

- **Example** - [Create and Cal an IMD measurement](#)
  - [Learn about IMD Calibration](#)
  - [Learn about Measurement Class](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
-



## SENSe<num>:CORRection:IMD:CALibration:FREQuencies <char>

(Read-Write) Sets and returns the frequencies at which an IMD source power cal is performed.

### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

<char> Choose from:

**CENTer** - Perform source power calibration at only the center frequencies, midway between the main tones.

**ALL** - Perform source power calibration at all main tone frequencies.

### Examples

```
SENS:CORR:IMD:CAL:FREQ ALL
```

```
sense2:correction:imd:calibration:frequencies center
```

**Query Syntax** SENSe<num>:CORRection:IMD:CALibration:FREQuencies?

**Return Type** Character

**Default** CENTer

---

## SENSe<num>:CORRection:IMD:CALibration:METHod <char>

(Read-Write) Sets and returns the method by which the match-correction portion of an IMD calibration is performed. [Learn more.](#)

### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

<char> Choose from:

**MATCh** - Performs a full 2-port cal for full match-correction.

**RESPonse** - Performs only a response (normalization) cal instead of a full 2-port cal.

### Examples

```
SENS:CORR:IMD:CAL:METH MATC
```

```
sense2:correction:imd:calibration:method response
```

**Query Syntax** SENSe<num>:CORRection:IMD:CALibration:METHod?

**Return Type** Character

**Default** MATCh

---

## SENSe<num>:CORRection:IMD:LO<n>:PCAL[STATe] <bool>

**(Read-Write)** Sets and returns whether or not the LO power cal step is included in the cal steps when an IMDX cal is performed. [Learn more.](#)

**Parameters**

<num> Any existing channel number; if unspecified, value is set to 1.

<n> LO Stage. Choose 1.

<char> LO Power Cal state. Choose from:

**0** or **OFF** - Skips over the LO Power Cal when calibrating.

**1** or **ON** - Includes a step for LO Power Cal when calibrating

**Examples**

```
SENS:CORR:IMD:LO1:PCAL 0
```

```
sense2:correction:imd:lo1:pcal:state on
```

**Query Syntax** SENSE<num>:CORRection:IMD:LO<n>:PCAL[STATe]?

**Return Type** Boolean

**Default** PNA Rev. 9.1 and above: **0** or **OFF**

Before Rev 9.1: **1** or **ON**

---

**SENSe<num>:CORRection:IMD:MPProduct <num>**

**(Read-Write)** Sets and returns the maximum intermod product frequencies to be calibrated. All lower product frequencies are also calibrated.

**Parameters**

<num> Any existing channel number; if unspecified, value is set to 1.

<char> Maximum IM products to calibrate. Choose from:

**2** - second order products

**3** - third order products

**5** - fifth order products

**7** - seventh order products

**9** - ninth order products

**Examples**

```
SENS:CORR:IMD:MPR 5
```

```
sense2:correction:imd:mproduct 9
```

---

<b>Query Syntax</b>	SENSe<cnum>::CORRection:IMD:MPRoduct?
<b>Return Type</b>	Numeric
<b>Default</b>	3

---

### SENSe<cnum>:CORRection:IMD:POWer <num>

**(Read-Write)** Sets and returns the power level at which to perform the source power calibration using a power sensor.

#### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <char> Power level. Choose a value between the min and max power level of the PNA.

#### Examples

```
SENS:CORR:IMD:POW -5
sense2:correction:imd:power 5
```

---

<b>Query Syntax</b>	SENSe<cnum>::CORRection:IMD:POWer?
<b>Return Type</b>	Numeric
<b>Default</b>	0

---

### SENSe<cnum>:CORRection:IMD:SENSor:CKIT <string>

**(Read-Write)** Sets and returns the cal kit to be used for calibrating at the port 1 reference plane when the power sensor connector is different from the DUT port 1. This effectively removes the effects of an adapter that is used to connect the power sensor.

#### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <string> Cal Kit enclosed in quotes. First set the DUT connector for port 1 and the connector of the power sensor. Then use [SENS:CORR:COLL:GUID:CKIT:PORT1:CAT?](#) to return a list of valid cal kits.

#### Examples

```
SENS:CORR:IMD:SENS:CKIT "85052B"
```

---

<b>Query Syntax</b>	SENSe<cnum>::CORRection:IMD:SENSor:CKIT?
<b>Return Type</b>	String
<b>Default</b>	Depends on the specified connectors

---

## SENSe<cnum>:CORRection:IMD:SENSor:CONNector <string>

(Read-Write) Sets and returns the power sensor connector type which is used to perform the Source Power Cal portion of an IMD calibration.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <string> Power sensor connector type. Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to return a list of valid connector types.
- Select "Ignored" to NOT compensate for the adapter.

### Examples

```
SENS:CORR:IMD:SENS:CONN "APC 3.5 male"
sense2:correction:imd:sensor:connector "Ignored"
```

**Query Syntax** SENSe<cnum>::CORRection:IMD:SENSor:CONNector?

**Return Type** String

**Default** "Ignored"

---

## SENSe<cnum>:CORRection:IMD:SORDer:INCLude <bool>

(Read-Write) Sets and returns whether to include the second order products in the calibration. These frequencies of these products can be far from the main tones.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <bool> Choose from:
- ON (or 1)** - Include 2nd order products
- OFF (or 0)** - Do NOT include 2nd order products

### Examples

```
SENS:CORR:IMD:SORD:INCL ON
```

**Query Syntax** SENSe<cnum>::CORRection:IMD:SORDer:INCLude?

**Return Type** Boolean

**Default** OFF or 0

---

Last Modified:

27-Mar-2009 Edited for IMDx  
18-Sep-2008 MX New topic



## Sense:Correction:Collect:Session Commands

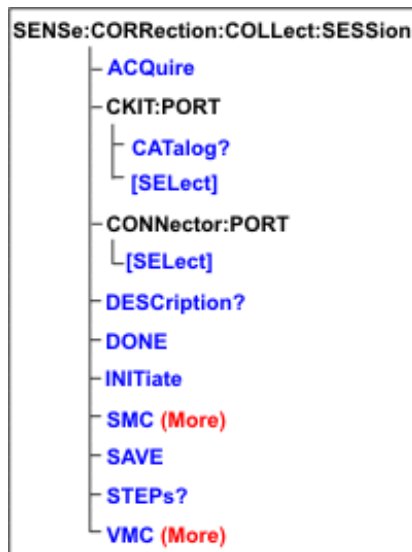
---

The commands in this topic are common to perform both SMC and VMC calibrations.

A **calibration session** is a term used to describe an instance of a SMC or VMC calibration. The session number is chosen in the [SENSe:CORR:COLL:SESS:INITiate](#) command. All other commands refer to that session number. For more commands, see [SESS:SMC](#) and [SESS:VMC](#).

Commands to read ([STEP?](#)) and describe ([DESC?](#)) each step are provided to facilitate a remote user interface.

To perform a **NON** - SMC or VMC calibration, use either [Guided](#) or [Mechanical](#) commands.



Click on a [blue](#) keyword to view the command details.

### See Also

- SCPI [SMC](#) and [VMC](#) calibration examples.
- Learn about [SMC and VMC calibrations](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:ACQuire <step>[,sync]**

(Write only) Acquire a calibration measurement.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1.
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <step> Step number to acquire. Use [SENS:CORR:COLL:SESS:STEPS?](#) to find the number of steps required for the calibration.
- [sync] Optional argument. Choose from:  
**SYNChronous** - blocks SCPI commands during standard measurement (default behavior).  
**ASYNchronous** - does NOT block SCPI commands during standard measurement.  
[Learn more.](#)

**Examples** `SENSe2:CORR:COLL:SESS6:ACQ 5,ASYN;*OPC?`

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:CKIT:PORT<p>:CATalog?

(Read only) Returns a list of cal kits that are compatible with the connector on port <p>. The port connector type is set with [SENS:CORR:COLL:SESS:CONN:PORT](#)

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1.
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <p> PNA port number connector to query for compatible cal kits.  
For VMC, output port of the calibration mixer, specify 3 unless already used for the output of the mixer. Otherwise, specify 4.  
[See the user interface \(UI\) equivalent of this command.](#)

**Examples** `SENS2:CORR:COLL:SESS6:CKIT:PORT2:CAT?`

**Return Type** Comma separated string values

**Default** Not Applicable

---

## SENSe<ch>:CORRection:COLLect:SESSion<n>:CKIT:PORT<p>[:SElect] <calkit>

(Read-Write) Set or return the Cal Kit for the specified port. Use [SENS:CORR:COLL:SESS:CKIT:PORT:CAT?](#) to list compatible Cal Kits.

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <p> PNA port number connector for which to specify a cal kit.  
For VMC, output port of the calibration mixer, specify 3 unless already used for the output of the mixer. Otherwise, specify 4.  
[See the user interface \(UI\) equivalent of this command.](#)
- <calkit> Cal Kit Name

### Examples

```
SENS:CORR:COLL:SESS:CKIT:PORT:SEL 85091A
SENS2:CORR:COLL:SESS6:CKIT:PORT2:SEL?
```

**Query Syntax** SENS<ch>:CORR:COLL:SESS<n>:CKIT:PORT<p>[:SEL] ?

**Return Type** String

**Default** Not Applicable

---

## SENSe<ch>:CORRection:COLLect:SESSion<n>:CONN:PORT<p>[:SEL] <conn>

(Read-Write) Set the connector type and sex for the specified port number. Catalog valid connector types using [SENS:CORR:COLL:GUID:CONN:CAT?](#)

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <p> PNA port number connector for which to specify a connector type.
- <conn> Name of the connector type

### Examples

```
SENS2:CORR:COLL:SESS6:CONN:PORT1:SEL "N Type"
```

**Query Syntax** SENS<ch>:CORR:COLL:SESS<n>:CONN:PORT<p>[:SEL] ?



**Return Type** String

**Default** Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:DESC? <step>

(Read-only) Returns the connection prompt for the step. List the number of steps in the calibration using [SENS:CORR:COLL:SESS:STEPS?](#).

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<step> Step number

**Examples** SENS1 : CORR : COLL : SESS6 : DESC?3

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:DONE

(Write only) Ends the calibration session. First use [SAVE?](#) to calculate error terms and save the CalSet.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

**Examples** SENS1 : CORR : COLL : SESS6 : SAVE?

SENS1 : CORR : COLL : SESS6 : DONE

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:INITiate <string>

(Write only) Initiates an SMC or VMC calibration session. Use the session number for subsequent SMC or VMC commands.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number. Choose from 1 to the [max number of channels](#).. If the session number already exists it will be terminated and a new session initiated.
- <string> Name of the calibration. Choose from:  
"VMC" or "VectorMixerCal.VMCType"  
"SMC" or "ScalarMixerCal.SMCType"

**Examples** `SENS1:CORR:COLL:SESS6:INITiate "VectorMixerCal.VMCType"`

**Query Syntax** Not Applicable

Default Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:SAVE?

(Read only) Finish the SMC or VMC calibration, compute error terms, populate and save the Cal Set, and return the GUID of the Cal Set.

**Note:** The destination (Cal Register or User Cal Set) is determined by the setting of the [SENS:CORR:PREFerence:CSET:SAVE](#) command.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)

**Examples** `SENS1:CORR:COLL:SESS6:SAVE?`

**Return Type** String specifying the GUID of the Cal Set produced by this session.

Default Not Applicable

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:STEPS?

(Read-only) Returns the number of steps required by the Calibration.

To ensure this query always completes successfully, first send the write command: SENS:CORR:COLL:SESS:STEP, then send the query.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

#### Examples

```
SENS1 : CORR : COLL : SESS6 : STEP s ?
```

**Return Type** Numeric

**Default** Not Applicable

---

#### Last Modified:

4-Jun-2009	Edits to DONE for clarification.
9-Apr-2009	Fixed commands for mixer port numbers.
13-Feb-2009	Added sync to ACQUIRE
5-Aug-2008	Edited sessions descriptions
13-Aug-2007	New image and detail for <port> argument.

## Sense:Correction:Collect:Session:SMC Commands

---

Performs scalar (SMC) calibration on a frequency converting device.

**SENSe:CORRectionCOLLect:SESSion:SMC**

| **ECAL**

| [CHARacterza](#)

| **FSIMulator**

| **NETWork**

| [FILename](#)

| [MODE](#)

| [IMPort](#)

| **PHASe**

| [DELay](#)

| [METHod](#)

| [MIXer](#)

| **PWRCal**

| [SEParate](#)

| [SRCPort](#)

| **TWOPort**

| **ECAL**

| [STATe](#)

| [PORTmap](#)

| [METHod](#)

| [OMITisolat](#)

| [OPTion](#)

Click on a [blue](#) keyword to view the command details.

Red keywords are obsolete.

## See Also

- [Example Programs](#)
- [Learn about FCA Calibrations](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**NOTE:** To configure a power meter and sensor see [SOURCE:POWer:](#) commands.

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:ECAL:CHARacteriza <mod> ,<char>**

(Read-Write) Specifies the ECal module and characterization to be used for the SMC calibration.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<mod> **1** - Electronic Calibration Module

<char> Specifies which characterization within the ECal module from which to read the confidence data.

**0** Factory characterization (data that was stored in the ECal module by Agilent). Default if not specified.

**1** User characterization #1

**2** User characterization #2

**3** User characterization #3

**4** User characterization #4

**5** User characterization #5

### Examples

```
SENS:CORR:COLL:SESS:SMC:ECAL:CHAR 1,2
```

**Query Syntax** SENS:CORR:COLL:SESS:SMC:ECAL:CHAR?

**Return Type** Numeric

**Default** 1,0

## **SENSe<ch>:CORRection:COLLect:SESSion<n> :SMC:FSIMulator:NETWork<p>:FILename<string>**

(Read-Write) Specifies the S2P filename to embed or de-embed on the input or output of your mixer measurement. [Learn more.](#)

### **Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.

[Learn about Cal sessions.](#)

- <p> Apply network to input or output of mixer. Choose from:

- 1 - Input of mixer
- 2 - Output of mixer

- <string> Filename of the S2P used for embedding or de-embedding. Use the full path name, file name, and .S2P suffix, enclosed in quotes.

### **Examples**

```
SENS:CORR:COLL:SESS:SMC:FSIM:NETW1:FIL "C:/Program  
Files/Agilent/Network Analyzer/Documents/WaveguideAdapt.S2P"
```

**Query Syntax** SENS<ch>:CORRection:COLLect:SESSion<n>  
:SMC:FSIMulator:NETWork<x>:FILename?

**Return Type** String

**Default** Not Applicable

---

## **SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:FSIMulator:NETWork<p>:MODE <char>**

**(Read-Write)** Allows you to embed (add) or de-embed (remove) circuit network effects on the input and output of your mixer measurement. [Learn more.](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<p> Apply network to input or output of mixer. Choose from:

1 - Input of mixer

2 - Output of mixer

<char> Choose from:

**NONE** - Do nothing with effects of S2P file.

**EMBed** - Add effects of S2P file from the measurement results.

**DEEMbed** - Remove effects of S2P file from the measurement results.

#### Examples

```
SENS:CORR:COLL:SESS:SMC:FSIM:NETW1:MODE EMB
```

**Query Syntax** SENS<ch>:CORRection:COLLect:SESSion<n>  
:SMC:FSIMulator:NETWork<x>:MODE?

**Return Type** Character

**Default** NONE

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:IMPort <calName>, <dataset>**

**(Write-only)** Imports existing Source Power Cal data into the SMC calibration.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<calName> (String) Name of existing Cal Set from which power meter data is imported.

<dataset> (String) Name of the data set. Use "POWER\_STEP"

**Examples**

```
SENS2:CORR:COLL:SESS1:SMC:IMP "MyPowerCal", "POWER_STEP"
```

[See example program](#)

**Query Syntax** Not Applicable

**Default** NONE

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PHASe:DELay <num>**

**(Read-Write)** Set and return the known delay through the calibration mixer. [Learn more.](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> Known delay through the calibration mixer in seconds.

**Examples**

```
SENS:CORR:COLL:SESS:SMC:PHAS:DEL 12e-9
```

```
sense2:correction:collect:session2:smc:phase:delay 12e-10
```

**Query Syntax** SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PHASe:DELay?

**Return Type** Numeric

**Default** 0 seconds

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PHASe:METHod <char>**



**(Read-Write)** Set and return the method of setting the delay through the calibration mixer. [Learn more.](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> **Choose from:**

**FIXed** - use a known delay value set with  
[SENS:CORR:COLL:SESS:SMC:PHAS:DEL](#)

**MIXer** - use the S2P file set with  
[SENS:CORR:COLL:SESS:SMC:PHAS:MIX](#)

#### Examples

```
SENS:CORR:COLL:SESS:SMC:PHAS:METH FIX
```

```
sense2:correction:collect:session2:smc:phase:method mixer
```

**Query Syntax** SENSE<ch>:CORRection:COLLect:SESSion<n>:SMC:PHASe:METHod?

**Return Type** Character

**Default** FIXed

---

### SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PHASe:MIXer <string>

**(Write-only)** Set the filename of the S2P file used to characterize the calibration mixer. [Learn more.](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<string> Calibration mixer filename. Use the following rules to specify path names:

- The default folder is "C:/Program Files/Agilent/Network Analyzer/Documents"
- You can change the active directory using [MMEMory:CDIRectory](#).
- Specify only the file name if using the active directory.
- You can also use an absolute path name to specify the folder and file.

**Examples**

```
SENS:CORR:COLL:SESS:SMC:PHAS:MIX "MyCalMixer.s2p"  
  
sense2:correction:collect:session2:smc:phase:mixer  
"MyCalMixer.s2p"
```

**Query Syntax** Not Applicable**Default** Not Applicable

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PWRCal:SEParate <bool>**

(Read-Write) Specifies whether to use a Thru standard or to use two power sensor connections during the power cal of an SMC calibration.

This command must be sent immediately after the [INITiate](#) command, but before you set all the other calibration commands.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> **OFF or 0** - Perform Cal with Thru standard.

**ON or 1** - Do NOT use a Thru, but instead perform separate power cals on Input and Output reference planes.

**Examples**

```
SENS:CORR:COLL:SESS:INIT "SMC"  
SENS:CORR:COLL:SESS:SMC:PWRC:SEP 1
```

**Query Syntax** SENS:CORR:COLL:SESS:SMC:PWRCal:SEParate?**Return Type** Boolean**Default** 0 or OFF

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:PWRCal:SRCPort <string> **Obsolete****

**(Read-Write)** Specifies which port to calibrate.

**Note:** Beginning with Rev 6.0, this command is no longer necessary. [Learn more.](#) Because of improved calibration techniques, **Both** is always selected although a power meter measurement is performed only on port 1.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> **'1'** Source port 1 (SMC forward direction)

**'2'** Source port 2 (SMC reverse direction)

**'BOTH'** (both forward and reverse directions)

#### Examples

```
SENS:CORR:COLL:SESS:SMC:PWRC:SRCP 'both'  
SENSe2:CORRection:COLLection:SESSion6:SMC:PWRCal:SRCPort '2'
```

**Query Syntax** SENS:CORR:COLL:SESS:SMC:PWRC:SRCP?

**Return Type** String

**Default** 1

---

### SENSe<ch>:CORRection:COLLection:SESSion<n>:SMC:TWOPort:ECAL:ORientation[:STATe] <bool>

**(Read-Write)** Sets ECAL Auto-Orientation ON or OFF. If setting auto-orientation OFF, you must manually specify the orientation of the ECAL module with [SENS:CORR:COLL:SESS:SMC:TWOP:ECAL:PORTmap](#).

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> **OFF or 0** = Orientation OFF

**ON or 1** = Orientation ON

#### Examples

```
SENS:CORR:COLL:SESS:SMC:TWOP:ECAL:ORI 1
```

**Query Syntax** SENS:CORR:COLL:SESS:SMC:TWOP:ECAL:ORI?

**Return Type** Boolean

**Default** 1

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:TWOPort:ECAL:PORTmap <mod>, <string>**

**(Read-Write)** Specifies the manual orientation (which ports of the module are connected to which ports of the PNA) when [auto-orientation](#) is OFF.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<mod> **1** - Electronic Calibration Module

<string> Format in the following manner:

Aw,Bx,Cy,Dz

where

- A, B, C, and D are literal ports on the ECAL module
- w,x,y, and z are substituted for PNA port numbers to which the ECAL module port is connected.

Ports of the module which are not used are omitted from the string.

**For example**, on a 4-port ECal module with

- port A connected to PNA port 2
- port B connected to PNA port 3
- port C not connected
- port D connected to PNA port 1

the string would be: A2,B3,D1

If either the receive port or source port (or load port for 2-port cal) of the CALC:PAR:SElected measurement is not in this string and orientation is OFF, an attempt to perform an ECal calibration will fail.

**Examples**

```
SENS:CORR:COLL:SESS:SMC:TWOP:ECAL:PORTmap 1, 'A1,B2'
```

**Query Syntax**

SENS:CORR:COLL:SESS:SMC:TWOP:ECAL:PORTmap?

**Return Type** String

**Default** "A1,B2"

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:TWOPort:METHod <string>**

(Read-Write) Specifies the guided ECal method for performing the thru portion of the calibration.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<string> ECAL Method: Choose from:

'DEFAULT' - Default

'ADAP' - Adapter removal

'FLUSH' - Flush Through

'UNKN' - Unknown Thru

**Examples**

```
SENS:CORR:COLL:SESS:SMC:TWOPort:METH 'default'
```

**Query Syntax**

SENS:CORR:COLL:SESS:SMC:TWOPort:METH?

**Return Type** String

**Default** DEFAULT

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:TWOPort:OMITisolat <bool>**

(Read-Write) Select to omit or perform the isolation portion of the ECAL.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> **ON or 1** - Omit isolation

**OFF or 0** - Perform isolation

**Examples**

```
SENS:CORR:COLL:SESS:SMC:TWOPort:OMIT 1
```

**Query Syntax** SENS:CORR:COLL:SMC:TWOPort:OMIT?

**Return Type** Boolean

**Default** 1

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:SMC:TWOPort:OPTion <string>**

(Read-Write) Sets the SMC calibration to ECAL or MECHanical

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> Choose from:

'ECAL' Electronic Calibration Module

'MECH' Mechanical Calibration Kit

**Examples**

```
SENS:CORR:COLL:SESS:SMC:TWOPort:OPTion 'ECAL'
```

**Query Syntax** SENS:CORR:COLL:SESS:SMC:TWOPort:OPTion?

**Return Type** String

**Default** ECAL

Last Modified:

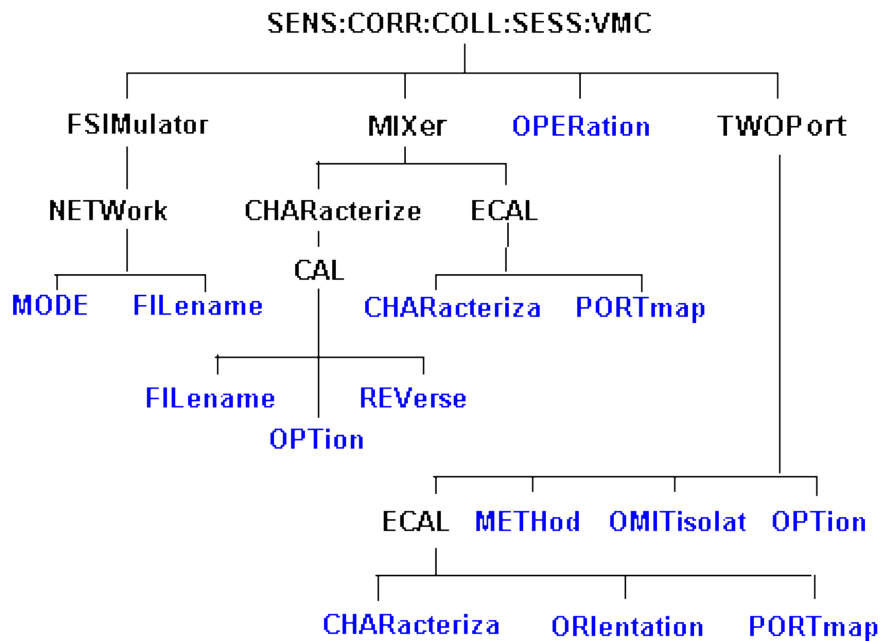
22-Dec-2009	Added Import
24-Feb-2009	Replaced True/False
5-Aug-2008	Edited sessions description



## Sense:Correction:Collect:Session:VMC Commands

---

Performs a vector (VMC) calibration on a frequency converting device.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about VMC Calibration](#)
- [Synchronizing the PNA and Controller](#)

---

**SENSe<ch>:CORRection:COLlect:SESSion<n> :VMC:FSIMulator:NETWork<x>:MODE  
<char>**



(Read-Write) Allows you to embed (add) or de-embed (remove) circuit network effects on the input and output of your mixer measurement. [Learn more.](#)

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <x> Apply network to input or output of mixer. Choose from:  
1 - Input of mixer  
2 - Output of mixer
- <char> Choose from:  
**NONE** - Do nothing with effects of S2P file.  
**EMBed** - Add effects of S2P file from the measurement results.  
**DEEMbed** - Remove effects of S2P file from the measurement results.

#### Examples

```
SENS:CORR:COLL:SESS:VMC:FSIM:NETW1:MODE EMB
```

**Query Syntax** SENS<ch>:CORRection:COLLect:SESSion<n>:VMC:FSIMulator:NETWork<x>:MODE?

**Return Type** Character

**Default** NONE

---

**SENSe<ch>:CORRection:COLLect:SESSion<n>:VMC:FSIMulator:NETWork<x>:FiLename<string>**

(Read-Write) Specifies the S2P filename to embed or de-embed on the input or output of your mixer measurement. [Learn more.](#)

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <x> Apply network to input or output of mixer. Choose from:  
1 - Input of mixer  
2 - Output of mixer

<string> Filename of the S2P used for embedding or de-embedding. Use the full path name, file name, and .S2P suffix, enclosed in quotes.

**Examples** `SENS:CORR:COLL:SESS:VMC:FSIM:NETW1:FIL "C:/Program Files/Agilent/Network Analyzer/Documents/WaveguideAdapt.S2P"`

**Query Syntax** `SENS<ch>:CORRection:COLLect:SESSion<n>:VMC:FSIMulator:NETWork<x>:FILename?`

**Return Type** String

**Default** Not Applicable

---

## **SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:MIXer:CHARacterize:CAL:FILename<string>**

(Read-Write) Specifies the .S2P filename used for mixer characterization. Use the [VMC:MIXer:CHARacterize:CAL: OPTion](#) command to load the file for mixer characterization. Once loaded, use this command to query the current filename or set a new filename.

### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<string> Filename of the S2P used for mixer characterization. Use the full path name, file name, and .S2P suffix, enclosed in quotes.

**Examples** `SENS2:CORR:COLL:SESS4:VMC:MIXer:CHAR:CAL:FIL "C:/Program Files/Agilent/Network Analyzer/Documents/MyMixer.S2P"`

**Query Syntax** `SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:FIL?`

**Return Type** String

**Default** C:/Program Files/Agilent/Network Analyzer/Documents/default.s2p

---

## **SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:MIXer:CHARacterize:CAL: OPTion<char>**

(Read-Write) Sets the mixer characterization method to ECal, Mechanical, or read from a file.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> **ECAL** - Electronic Calibration Module

**MECH** - Mechanical Calibration Kit

**FILE, <filename>** - Retrieve a mixer characterization file. Also specify the filename of the S2P used for mixer characterization. Use the full path name, file name, and .S2P suffix. Use the [VMC:CHARacterize:CAL:FILEname](#) command to query the filename..

#### Examples

```
SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:OPT ECAL
```

```
'or
```

```
SENS:CORR:COLL:SESS6:VMC:MIX:CHAR:CAL:OPT FILE, 'C:/Program  
Files/Agilent/Network Analyzer/Documents/Mixer001.s2p'
```

```
file = "SENS:CORR:COLL:SESS6:VMC:MIXer:CHAR:CAL:FIL?" 'Read back  
the filename
```

**Query Syntax** SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:OPT?

**Return Type** String

**Default** MECH

---

## SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:MIXer:CHARacterize:CAL: REVerse<bool>

(Read-Write) Specifies the direction in which to characterize the calibration mixer. [Learn more about the calibration mixer.](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> **OFF (0)** - Characterize the calibration mixer in the SAME direction as that specified in the mixer setup.

**ON (1)** - Characterize the calibration mixer in the REVERSE direction as that specified in the mixer setup.

**Examples** SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:REV 1

**Query Syntax** SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:REV?

**Return Type** Boolean

**Default** OFF

---

## SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:MIXer:ECAL:CHARacteriza <mod>,<char>

(Read-Write) Specifies the ECal module and characterization to be used for the mixer characterization portion of the calibration.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<mod> 1 - Electronic Calibration Module

<char> Characterization number in the specified ECAL module. Choose from:

0 Factory characterization (data that was stored in the ECal module by Agilent). Default if not specified.

1 User characterization #1

2 User characterization #2

3 User characterization #3

4 User characterization #4

5 User characterization #5

**Examples** SENS:CORR:COLL:SESS:VMC:MIX:ECAL:CHAR 1,0

**Query Syntax** SENS:CORR:COLL:SESS:VMC:MIX:ECAL:CHAR?

**Return Type** Numeric

**Default** 1,0

---

## SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:MIXer:ECAL:PORTmap <mod>,<string>

(Read-Write) Sets the port mapping for the mixer characterization with ECal. This command is required if [SENS:CORR:COLL:SESS:VMC:MIX:CHAR:CAL:OPT ECAL](#) is specified.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<mod> 1 - Electronic Calibration Module

<string> Choose from:

"A1" - ECAL module port A is connected to PNA port 1

"B1" - ECAL module port B is connected to PNA port 1

#### Examples

```
SENS:CORR:COLL:SESS:VMC:MIX:ECAL:PORT?  
SENS2:CORR:COLL:SESS:VMC:MIXer:ECAL:PORTmap 1, "A1"
```

**Query Syntax** SENS:CORR:COLL:SESS:VMC:MIX:ECAL:PORTmap?

**Return Type** String

**Default** "A1"

---

## SENSe<ch>:CORRection:COLLect:SESSion<n>:VMC:OPERation <string>

(Read-Write) Perform either full VMC calibration or mixer characterization only.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> 'CAL' - full calibration and mixer characterization

'CHAR' - mixer characterization only (no reference mixer required) - Saves an .S2P file with the filename specified in [SENS<ch>:CORR:COLL:SESSion<n>:VMC:CHARacterize:CAL:FILEname <filename>](#). If none is specified, a filename is automatically generated and can be queried using the filename command.

#### Examples

```
SENS:CORR:COLL:SESS:VMC:OPER 'CAL'
```

**Query Syntax** SENS:CORR:COLL:SESS:VMC:OPER?

**Return Type** String

**Default** CAL

---

**SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:TWOPort:ECAL:CHARacteriza <mod>  
<char>**

(Read-Write) Specifies the ECal module and characterization to be used for the VMC calibration.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Session number of the calibration.  
[Learn about Cal sessions.](#)
- <mod> 1 - Electronic Calibration Module
- <char> Characterization number in the specified ECAL module. Choose from:
  - 0 Factory characterization (data that was stored in the ECal module by Agilent). Default if not specified.
  - 1 User characterization #1
  - 2 User characterization #2
  - 3 User characterization #3
  - 4 User characterization #4
  - 5 User characterization #5

**Examples** SENS:CORR:COLL:SESS:VMC:TWOPort:ECAL:CHAR 1,1

**Query Syntax** SENS:CORR:COLL:SESS:VMC:TWOPort:ECAL:CHAR?

**Return Type** Integer

**Default** 1,0

---

**SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:TWOPort:ECAL:ORlentation[:STATe]  
<bool>**

(Read-Write) Sets ECAL orientation for the VMC ECal.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> **ON or 1** - Perform orientation

**OFF or 0** - Do NOT perform orientation

#### Examples

```
SENS:CORR:COLL:SESS:VMC:TWOP:ECAL:ORI 1
```

**Query Syntax** SENS:CORR:COLL:SESSion:VMC:TWOPort:ECAL:ORlentation[:STATe]?

**Return Type** Integer

**Default** ON

---

### SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:TWOPort:ECAL:PORTmap <mod>, <string>

(Read-Write) Specifies the manual orientation (which ports of the module are connected to which ports of the PNA) when [orientation](#) is turned off.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<mod> **1** - Electronic Calibration Module

<string> Port Map, formatted in the following manner:

**Aw,Bx,Cy,Dz**

where:

- A, B, C, and D are literal ports on the ECAL module.
- w,x,y, z are substituted for PNA port numbers to which the ECAL module port is connected.
- Ports of the module which are not used are omitted from the string.

For example, on a 4-port ECal module with:

- port A connected to PNA port 2
- port B connected to PNA port 3
- port C not connected
- port D connected to PNA port 1

the string would be: A2,B3,D1

If either the receive port or source port (or load port for 2-port cal) of the measurement is not in this string and orientation is OFF, an attempt to perform an ECal will fail.

**Examples** `SENS:CORR:COLL:SESS:VMC:TWOP:ECAL:PORTmap 1, "A2,B1"`

**Query Syntax** `SENS:CORR:COLL:SESS:VMC:TWOP:ECAL:PORTmap?`

**Return Type** string

**Default** "A1,B2"

---

## **SENSe<ch>:CORRection:COLlect:SESSion<n> :VMC:TWOPort:METHod <string>**

(Read-Write) Specifies the guided ECal method for performing the thru portion of the calibration.

### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> **'DEFAULT'** - Default  
**'ADAP'** - Adapter removal  
**'FLUSH'** - Flush Through  
**'UNKN'** - Unknown Thru

**Examples** `SENS:CORR:COLL:SESS:VMC:TWOP:METH 'ADAP'`  
`SENSe2:CORR:COLL:SESSion6:VMC:TWOPort:METHod 'FLUSH'`

**Query Syntax** `SENS:CORR:COLL:SESS:VMC:TWOP:METH?`

**Return Type** String

**Default** DEFAULT

---



## SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:TWOPort:OMITisolat <bool>

(Read-Write) Select to omit or perform the isolation portion of the ECAL.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<bool> ON or 1 - omit isolation  
OFF or 0 - perform isolation

### Examples

```
SENS:CORR:COLL:SESS:VMC:TWOP:OMIT 1
```

**Query Syntax** SENS:CORR:COLL:SESS:VMC:TWOP:OMIT?

**Return Type** Boolean

**Default** ON

---

## SENSe<ch>:CORRection:COLLect:SESSion<n> :VMC:TWOPort:OPTion <string>

(Read-Write) Sets the 2-port calibration option to ECAL or MECHanical

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<n> Session number of the calibration.

[Learn about Cal sessions.](#)

<char> **Choose from:**  
'ECAL' Electronic Calibration Module  
'MECH' Mechanical Calibration Kit

### Examples

```
SENS:CORR:COLL:SESS:VMC:TWOP:OPT 'MECH'  
SENSe2:CORR:COLL:SESSion6:VMC:TWOPort:OPTion 'ECAL'
```

**Query Syntax** SENS:CORR:COLL:SESS:VMC:TWOP:OPT?

**Return Type** String

**Default** "MECH"

---

Last Modified:

24-Feb-2009	Replaced True/False
5-Aug-2008	Edited session description
29-Aug-2007	Edited Char:Cal:Opt File argument

## Sense:Couple Commands

---

### SENSe:COUPle

#### PARAmeter

#### [STATe]

Click on a [blue](#) keyword to view the command details.

#### See Also

- [Example Programs](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

### SENSe<num>:COUPle <ALL | NONE>

(Read-Write) Sets the sweep mode as Chopped or Alternate.

#### [Learn about Alternate Sweep](#)

#### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

<ALL | NONE> **ALL** - Sweep mode set to Chopped - reflection and transmission measured on the same sweep.

**NONE** - Sweep mode set to Alternate - reflection and transmission measured on separate sweeps. Improves Mixer bounce and Isolation measurements. Increases sweep time

#### Examples

```
SENS:COUP ALL
sense2:couple none
```

**Query Syntax** SENSe<num>:COUPle?

**Return Type** Character

**Default** ALL

---

### SENSe<num>:COUPle:PARAmeter[:STATe] <bool>

**(Read-Write)** Turns ON and OFF Time Domain Trace Coupling. All of the measurements in the specified channel are coupled.

- To select Transform parameters to couple, use [CALC:TRAN:COUP:PAR](#)
- To select Gating parameters to couple, use [CALC:FILT:COUP:PAR](#)

Learn more about [Time Domain Trace Coupling](#).

#### Parameters

<cnun> Any existing channel number; if unspecified, value is set to 1.

<bool> **ON (or 1)** - Turns ON Time Domain Trace Coupling.

**OFF (or 0)** - Turns OFF Time Domain Trace Coupling.

#### Examples

```
SENS:COUP:PAR 0
sense2:couple:parameter:state on
```

**Query Syntax** SENSE<cnun>:COUPle:PARAmeter[:STATe]?

**Return Type** Boolean

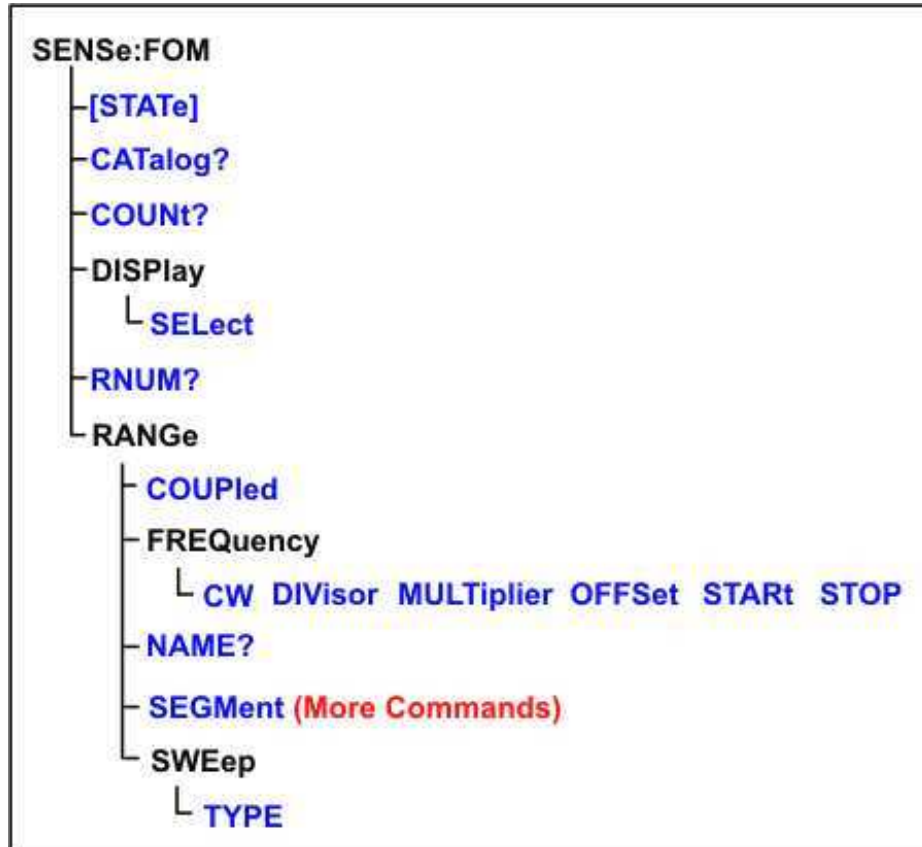
**Default** ON (or 1)

## Sense:FOM (Frequency Offset) Commands

---

Controls the frequency offset settings which cause stimulus and response frequencies to be different.

**Note:** These commands replace the [previous FOM commands](#). Although the old commands will continue to work, they can NOT be mixed with these new commands.



Click on a [blue](#) keyword to view the command details.

### See Also

- [FOM Example Program](#)
- [Learn about Frequency Offset](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

**SENSe<num>:FOM[:STATe] <bool>**

(Read-Write) Turns Frequency Offset ON and OFF. Frequency offset settings are not enabled until this setting is ON.

Send this command (FOM ON) AFTER sending other FOM settings to avoid 'out-of-range' errors.

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<bool> ON (or 1) - turns FOM ON.  
OFF (or 0) - turns FOM OFF.

#### Examples

```
SENS:FOM 1  
sense2:fom:state on
```

**Query Syntax** SENSE<cnum>:FOM:STATE?

**Return Type** Boolean

**Default** OFF

---

## SENSe<cnum>:FOM:CATalog?

(Read-only) Returns a comma-separated list of available range names in the PNA.

Use [SENS:FOM:CAT?](#) to see a list of available range names.

Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.

Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.

Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:FOM:CAT?  
'returns  
"Primary, Source, Receivers"
```

**Return Type** String

**Default** Not Applicable

---

## SENSe<cnum>:FOM:COUNt?

(Read-only) Returns the number of valid range numbers in the PNA.

#### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:FOM:COUN?  
sense2:fom:count?
```

**Query Syntax** SENSE<num>:FOM:COUNT?

**Return Type** Numeric

Default Not Applicable

---

### SENSe<num>:FOM:DISPlay:SElect <string>

(Read-Write) Select the range to be displayed on the PNA x-axis. All traces in the channel have this same x-axis scaling.

#### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

<string> Range name. Case insensitive. Use [SENSe:FOM:CAT?](#) to see a list of available frequency range names.

#### Examples

```
SENS:FOM:DISPlay:SElect "source2"  
sense2:fom:display:select "source"
```

**Query Syntax** SENSe<num>:FOM:DISPlay:SElect?

**Return Type** String

Default Receivers

---

### SENSe<num>:FOM:RNUM? <string>

(Read-only) Returns the number of a specified range name.

The FOM range items are typically numbered as follows:

1. Primary
2. Source
3. Receivers
4. Source2 (if present)

Use [SENS:FOM:CAT?](#) to see a list of available range names.

Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.

Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <string> Range name for which a number is being queried. Case insensitive.

#### Examples

```
SENS:FOM:RNUM? "receivers"  
sense2:fom:rnum? "Source2"
```

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe<num>:FOM:RANGe<n>:COUPled <bool>

(Read-Write) Sets and returns the state of coupling (ON or OFF) of the specified range to the primary range.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number to couple to primary range. An error is returned when attempting to couple to the Primary range (1).  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <bool> **ON** (or 1) - Couple range to primary range.  
**OFF** (or 0) - Do NOT couple to primary range.



**Examples** `SENS:FOM:RANG:COUP 1`  
`sense2:fom:range2:coupled 0`

**Query Syntax** `SENSe<num>:FOM:RANGe<n>:COUPled?`

**Return Type** Boolean

**Default** ON (or 1) Coupled

---

## **SENSe<num>:FOM:RANGe<n>:FREQuency:CW <num>**

(Read-Write) Sets and returns the CW frequency.

This setting is valid for the primary range, or if the specified range is already [uncoupled](#) from the primary range and if the [sweep type](#) is CW.

### **Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNT?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num> CW frequency value in Hz. Choose any frequency within the range of the PNA.

**Examples** `SENS:FOM:RANG:FREQ:CW 1e9`  
`sense2:fom:range2:frequency:cw 10000000`

**Query Syntax** `SENSe<num>:FOM:RANGe:<n>:FREQuency:CW?`

**Return Type** Numeric

**Default** Center frequency of the PNA.

---

## **SENSe<num>:FOM:RANGe<n>:FREQuency:DIVisor <num>**

(Read-Write) Sets and returns the divisor value.

This setting is valid only if the specified range is [coupled](#) to the primary range.

#### Parameters

- <cnm> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num> Divisor value (unitless).

#### Examples

```
SENS:FOM:RANG:FREQ:DIV 3  
sense2:fom:range2:frequency:divisor 0
```

**Query Syntax** SENSE<cnm>:FOM:RANGe<n>:FREQuency:DIVisor?

**Return Type** Numeric

**Default** 1

---

### SENSE<cnm>:FOM:RANGe<n>:FREQuency:MULTiplier <num>

(Read-Write) Sets and returns the multiplier value to be used when coupling this range to the primary range.

This setting is valid only if the specified range is [coupled](#) to the primary range.

#### Parameters

- <cnm> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num> Multiplier value. (Unitless)

#### Examples

```
SENS:FOM:RANG:FREQ:MULT 1  
sense2:fom:range2:frequency:multiplier 2
```

---

**Query Syntax**    SENSE<cnum>:FOM:RANGE<n>:FREQUENCY:MULTplier?

**Return Type**    Numeric

**Default**    1

---

### SENSe<cnum>:FOM:RANGE<n>:FREQUENCY:OFFSet <num>

(Read-Write) Sets and returns the offset value to be used when coupling this range to the primary range. [Learn more about offset value.](#)

This setting is valid only if the specified range is [coupled](#) to the primary range.

#### Parameters

- <cnum>    Any existing channel number; if unspecified, value is set to 1.
- <n>    Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNT?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num>    Offset value. (Unitless)

#### Examples

```
SENS:FOM:RANG:FREQ:OFFS 1E9  
sense2:fom:range2:frequency:offset 10000000
```

**Query Syntax**    SENSE<cnum>:FOM:RANGE<n>:FREQUENCY:OFFSet?

**Return Type**    Numeric

**Default**    0

---

### SENSe<cnum>:FOM:RANGE<n>:FREQUENCY:STARt <num>

(Read-Write) Sets and returns the Start value of frequency range. Also specify [Stop frequency](#).

This setting is valid for the primary range, or if the specified range is already [uncoupled](#) from the primary range and if the [sweep type](#) is LOG or LINear.

#### Parameters

- <cnun> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num> Start value in Hz. Choose any frequency within the range of the PNA.

#### Examples

```
SENS:FOM:RANG:FREQ:STAR 1GHz  
sense2:fom:range2:frequency:start 100000000
```

**Query Syntax** SENSE<cnun>:FOM:RANGe<n>:FREQuency:STARt?

**Return Type** Numeric

**Default** Minimum frequency of the PNA.

---

### SENSE<cnun>:FOM:RANGe<n>:FREQuency:STOP <num>

(Read-Write) Sets and returns the Stop value of frequency range. Also specify [Start frequency](#).

This setting is valid for the primary range, or if the specified range is already [uncoupled](#) from the primary range and if the [sweep type](#) is LOG or LINear.

#### Parameters

- <cnun> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <num> Stop value in Hz. Choose any frequency within the range of the PNA.

**Examples**

```
SENS:FOM:RANG:FREQ:STOP 1e12
sense2:fom:range2:frequency:stop 10000000000
```

**Query Syntax**

SENSe<cnum>:FOM:RANGe<n>:FREQuency:STOP?

**Return Type**

Numeric

**Default**

Maximum frequency of the PNA.

---

**SENSe<cnum>:FOM:RANGe<n>:NAME?**

(Read-only) Returns the name of range<n>.

The FOM range items are typically named as follows:

1. Primary
2. Source
3. Receivers
4. Source2 (if present)

Use [SENS:FOM:CAT?](#) to see a list of available range names.

Use [SENS:FOM:COUNt?](#) to see a list of available range numbers.

Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.

<n> Range number. If unspecified, value is set to 1.

**Examples**

```
SENS:FOM:RANG:NAME?
sense2:fom:range2:name?
```

**Return Type**

String

**Default**

Not Applicable

---

**SENSe<cnum>:FOM:RANGe<n>:SWEep:TYPE <char>**

(Read-Write) Sets and returns the sweep type to be used with the specified range.

This setting is valid only if the specified range is already [uncoupled](#) from the primary range.

Learn about [Unsupported Sweep Type combinations](#).

### Parameters

- <cnun> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.  
Use [SENS:FOM:CAT?](#) to see a list of available range names.  
Use [SENS:FOM:COUNT?](#) to see a list of available range numbers.  
Use [SENS:FOM:RNUM?](#) to see the range number for a specific name.  
Use [SENS:FOM:RANG:NAME?](#) to see the range name for a specific number.
- <char> Sweep type. Choose from:  
**CW** - Also specify [CW frequency](#).  
**LINear** - Also specify frequency Start/Stop or Center/Span  
**LOG** - Also specify frequency Start/Stop or Center/Span  
**POWer** - Also specify power Start/Stop or Center/Span  
**SEGment** - Also specify [segment sweep](#) settings.

### Examples

```
SENS:FOM:RANG:SWE:TYPE LOG  
sense2:fom:range2:sweep:type linear:
```

**Query Syntax** SENSE<cnun>:FOM:RANGe<n>:SWEep:TYPE?

**Return Type** Character

**Default** Linear

---

Last Modified:

24-Sep-2008 Updated with range queries

5-Jul-2007 Fixed state link

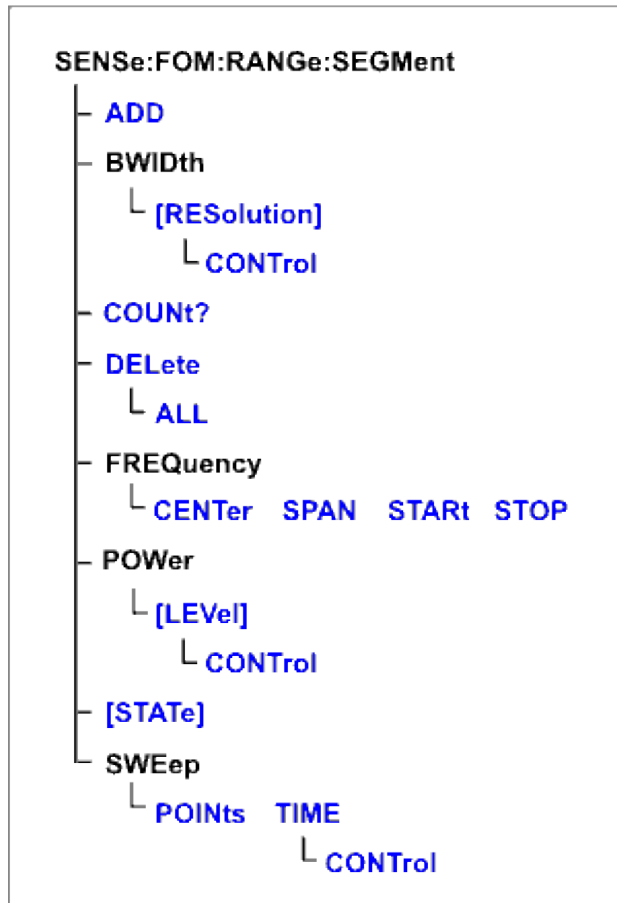
8-Mar-2007 MQX New topic

## Sense:FOM:Range:Segment Commands

---

Constructs a segment table for a specified [UNCOUPLED](#) FOM range.

**Note:** Do NOT use [Sens:Segment](#) commands for FOM segment sweep.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Other SENSE:FOM Commands](#)
- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

**SENSe<num>:FOM:RANGe<n>:SEGMENT<s>:ADD**

(Write-only) Adds a segment.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to add. If unspecified, value is set to 1. Segment numbers must be sequential. If a new number is added where one currently exists, the existing segment and those following are incremented by one.

#### Examples

Two Segments exist (1 and 2). The following command will add a new segment (1). The existing (1 and 2) will become (2 and 3) respectively.

```
sense2:fom:range2:segment:add
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SENSe<num>:FOM:RANGe<n>:SEGMENT<s>:BWIDth[:RESolution] <num>

(Read-Write) Sets the IF Bandwidth for the specified segment. First set [SENS:FOM:RANGe:SEGM:BWIDth:CONTRol ON](#). All subsequent segments that are added assume the new IF Bandwidth value.

Valid either for Receiver range or for Primary range when coupled to Receiver.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number for which to set independent IF Bandwidth.
- <num> IF Bandwidth in Hz. The list of valid IF Bandwidths is different depending on the PNA model. [See the list of valid IFBW values](#). If an invalid number is specified, the analyzer will round up to the closest valid number.

#### Examples

```
SENS:FOM:RANG:SEGM:BWIDth 100
```

```
sense2:fom:range2:segment4:bwidth:resolution 1e3
```

**Query Syntax** SENSe<num>:FOM:RANGe<n>:SEGMENT<s>:BWIDth[:RESolution]?

**Return Type** Numeric

**Default** See [Preset IFBW](#) for your PNA model.



---

## SENSe<cnum>:FOM:RANGe<n>:SEGMent:BWIDth[:RESolution]:CONTrol <bool>

(Read-Write) Specifies whether the IF Bandwidth resolution can be set independently for each segment. When set, each segment added after this will be set to ON automatically.

Valid either for Receiver range or for Primary range. Primary range value is ignored unless Receiver is coupled to Primary.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <bool> **ON** (or 1) - turns Bandwidth control ON. Bandwidth can be set for each segment  
**OFF** (or 0) - turns Bandwidth control OFF. Use the channel IF bandwidth setting instead.

### Examples

```
SENS:FOM:RANG:SEGM:BWIDth:CONT 0  
sense2:fom:range2:segment:bandwidth:resolution:control 1
```

**Query Syntax** SENSe<cnum>:FOM:RANGe<n>:SEGMent:BWIDth[:RESolution]:CONTrol?

**Return Type** Boolean

Default OFF

---

## SENSe<cnum>:FOM:RANGe<n>:SEGMent:COUNt?

(Read-only) Returns the number of segments that exist for the specified range.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.

### Examples

```
SENS:FOM:RANG:SEGM:COUN?  
sense2:fom:range2:segment:count?
```

**Return Type** Numeric

Default Not Applicable

---

## SENSe<cnum>:FOM:RANGe<n>:SEGMent<s>:DELe

(Write-only) Deletes the specified sweep segment.

#### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Number of the segment to delete. If unspecified, value is set to 1.

#### Examples

```
SENS:FOM:RANG:SEGM3:DEL  
sense2:fom:range2:segment4:delete
```

Query Syntax Not Applicable

Default Not Applicable

---

### SENSe<cnum>:FOM:RANGe<n>:SEGMENT:DELeTe:ALL

(Write-only) Deletes all sweep segments in the specified range.

#### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.

#### Examples

```
SENS:FOM:RANG:SEGM:DEL:ALL  
sense2:fom:range2:segment:delete:all
```

Query Syntax Not Applicable

Default Not Applicable

---

### SENSe<cnum>:FOM:RANGe<n>:SEGMENT<s>:FREQuency:CENTer <num>

**(Read-Write)** Sets and returns the center frequency for the specified sweep segment. Also specify segment frequency span.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <num> Center Frequency in Hz. Choose any number between the minimum and maximum frequency of the analyzer.

#### Examples

```
SENS:FOM:RANG:SEGM:FREQ:CENT 1GHz  
sense2:fom:range2:segment4:frequency:center 1e9
```

**Query Syntax** SENSE<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:CENTer?

**Return Type** Numeric

**Default** Stop Frequency of the previous segment. If first segment, start frequency of the analyzer.

---

### SENSe<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:SPAN <num>

**(Read-Write)** Sets and returns the span frequency for the specified sweep segment. Also specify segment center frequency.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <num> Frequency span in Hz. Choose any number between the minimum and maximum frequency of the analyzer.

#### Examples

```
SENS:FOM:RANG:SEGM:FREQ:SPAN 1GHz  
sense2:fom:range2:segment4:frequency:span 1e9
```

**Query Syntax** SENSE<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:SPAN?

**Return Type** Numeric

**Default** If first segment, frequency span of the analyzer. Otherwise 0.

---

## **SENSe<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:STARt <num>**

(Read-Write) Sets and returns the start frequency for the specified sweep segment. Also specify segment stop frequency.

All other segment Start and Stop Frequency values that are larger than this frequency are changed to this frequency.

### **Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <num> Start frequency in Hz. Choose any number between the minimum and maximum frequency of the analyzer.

### **Examples**

```
SENS:FOM:RANG:SEGM:FREQ:STAR 1GHz  
sense2:fom:range2:segment4:frequency:start 1e9
```

**Query Syntax** SENSe<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:STARt?

**Return Type** Numeric

**Default** Stop Frequency of the previous segment. If first segment, start frequency of the analyzer.

---

## **SENSe<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:STOP <num>**

**(Read-Write)** Sets and returns the stop frequency for the specified sweep segment. Also specify segment start frequency.

All other segment Start and Stop Frequency values that are larger than this frequency are changed to this frequency.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <num> Stop frequency in Hz. Choose any number between the minimum and maximum frequency of the analyzer.

#### Examples

```
SENS:FOM:RANG:SEGM:FREQ:STOP 1GHz  
sense2:fom:range2:segment4:frequency:stop 1e9
```

**Query Syntax**    SENSE<num>:FOM:RANGe<n>:SEGMent<s>:FREQuency:STOP?

**Return Type**    Numeric

**Default**    Stop Frequency of the previous segment. If first segment, start frequency of the analyzer.

---

### SENSe<num>:FOM:RANGe<n>:SEGMent<s>:POWer<p>[:LEVel] <num>

**(Read-Write)** Sets the Port Power level for the specified sweep segment. First set SENS:FOM:RANG:SEGM:POW:CONTRol ON.

When [port power is Coupled](#), setting port power for one port will apply port power for all source ports.

All subsequent segments that are added assume the new Power Level value.

Valid either for Source ranges or for Primary range when [coupled](#) to the source.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <p> Port number of the source. If unspecified, value is set to 1.
- <num> Power level in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, send SOUR:POW? MAX and SOUR:POW? MIN. ([SOUR:POW:ATT:AUTO](#) must be set to ON).

Actual achievable leveled power depends on frequency.

**Examples**

```
SENS:FOM:RANG:SEGM:POW -5  
sense2:fom:range2:segment4:power2:level 5
```

**Query Syntax** SENSE<num>:FOM:RANGe<n>:SEGMENT<s>:POWER<p>[:LEVel]?

**Return Type** Numeric

Default 0

---

## SENSe<num>:FOM:RANGe<n>:SEGMENT:POWER[:LEVel]:CONTrol <bool>

(Read-Write) Specifies whether Power Level is to be set independently for each segment.

Valid either for Source ranges or for Primary range. Primary range value is ignored unless Source is [coupled](#) to Primary.

**Parameters**

<num> Any existing channel number; if unspecified, value is set to 1.

<n> Range number. If unspecified, value is set to 1.

<bool> ON (or 1) - Power level will be set for each segment.  
OFF (or 0) - Use the channel power level setting.

**Examples**

```
SENS:FOM:RANG:SEGM:POW:CONT 0  
sense2:fom:range2:segment:power:control on
```

**Query Syntax** SENSE<num>:FOM:RANGe<n>:SEGMENT:POWER[:LEVel]:CONTrol?

**Return Type** Boolean

Default OFF (or 0)

---

## SENSe<num>:FOM:RANGe<n>:SEGMENT<s>[:STATe] <bool>

(Read-Write) Turns the specified sweep segment ON or OFF.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to be turned ON or OFF. Choose any existing segment number.
- <bool> ON (or 1) - turns segment ON.  
OFF (or 0) - turns segment OFF.

#### Examples

```
SENS:FOM:RANG:SEGM 0  
sense2:fom:range2:segment4:state on
```

**Query Syntax** SENSE<num>:FOM:RANGe<n>:SEGMent<s>[STATe]?

**Return Type** Boolean

Default OFF (or 0)

---

### SENSe<num>:FOM:RANGe<n>:SEGMent<s>:SWEep:POINTs <num>

(Read-Write) Sets the number of data points for the specified sweep segment.

#### Parameters

- <num> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number to modify. Choose any existing segment number.
- <num> Number of points in the segment. The total number of points in all segments cannot exceed 20001. A segment can have as few as 1 point.

#### Examples

```
SENS:FOM:RANG:SEGM:SWE:POIN 101  
sense2:fom:range2:segment4:sweep:points 201
```

**Query Syntax** SENSe<num>:FOM:RANGe<n>:SEGMent<s>:SWEep:POINTs?

**Return Type** Numeric

Default 21

## SENSe<cnum>:FOM:RANGe<n>:SEGMENT<s>:SWEep:TIME <num>

(Read-Write) Sets the time the PNA takes to sweep the specified segment.

Valid ONLY for receiver ranges.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <s> Segment number for which to set sweep time.
- <num> Sweep time in seconds. Choose a number between 0 and 100

### Examples

```
SENS:FOM:RANG:SEGM:SWE:TIME 1
sense2:fom:range2:segment3:sweep:time .1
```

**Query Syntax** SENSe<cnum>:FOM:RANGe<n>:SEGMENT<s>:SWEep:TIME?

**Return Type** Numeric

Default Not Applicable

---

## SENSe<cnum>:FOM:RANGe<n>:SEGMENT:SWEep:TIME:CONTROL <bool>

(Read-Write) Specifies whether Sweep Time can be set independently for each sweep segment.

Valid either for Receiver ranges or for Primary range. Primary range value is ignored unless Receiver is [coupled](#) to Primary.

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <n> Range number. If unspecified, value is set to 1.
- <bool> **ON** (or 1) - Sweep time will be set for each segment.  
**OFF** (or 0) - Use the channel sweep time setting.

### Examples

```
SENS:FOM:RANG:SEGM:SWE:TIME:CONT 1
sense2:fom:range2:segment:sweep:time:control off
```

**Query Syntax** SENSe<cnum>:FOM:RANGe<n>:SEGMENT:SWEep:TIME:CONTROL?

**Return Type** Boolean



Default OFF

---

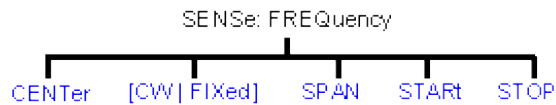
Last Modified:

21-Jun-2007    Increased max number of points

## Sense:Frequency Commands

---

Sets the frequency sweep functions of the analyzer.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example](#) using some of these commands.
- [Learn about Frequency Sweep](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### SENSe<cnum>:FREQuency:CENTer <num>

**(Read-Write)** Sets the center frequency of the analyzer.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Center frequency. Choose any number between the **minimum** and **maximum** frequency limits of the analyzer. Units are Hz.
- This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:FREQ:CENT 1000000
sense2:frequency:center 1mhz
sense2:frequency:center 1e6
```

**Query Syntax** SENSe<cnum>:FREQuency:CENTer?

**Return Type** Numeric

**Default** Center of the analyzer's frequency span

---

### SENSe<cnum>:FREQuency[:CW][:FIXed] <num>

(Read-Write) Sets the Continuous Wave (or Fixed) frequency. Must also send [SENS:SWEEP:TYPE CW](#) to put the analyzer into CW sweep mode.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> CW frequency. Choose any number between the **minimum** and **maximum** frequency limits of the analyzer. Units are Hz.
- This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:FREQ 1000000
SENS:FREQ:CW MIN
sense2:frequency:fixed 1mhz
```

**Query Syntax** SENSE<num>:FREQuency[:CW | :FIXed]?

**Return Type** Numeric

**Default** 1 GHz

---

### SENSe<num>:FREQuency:SPAN <num>

(Read-Write) Sets the frequency span of the analyzer.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> Frequency span in Hz. Choose any number between **0** (minimum) and the **maximum** frequency span of the analyzer.
- This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:FREQ:SPAN 1000000
sense2:frequency:span max
```

**Query Syntax** SENSe<num>:FREQuency:SPAN?

**Return Type** Numeric

**Default** Maximum frequency span of the analyzer

---

### SENSe<num>:FREQuency:STArT <num>

(Read-Write) Sets the start frequency of the analyzer.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Start frequency. Choose any number between the **MIN** and **MAX** frequency limits of the analyzer. Units are Hz
- If FREQ:START is set greater than FREQ:STOP, then STOP is set equal to START.
- This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:FREQ:STAR 1000000
sense2:frequency:start MIN
```

**Query Syntax** SENSE<cnum>:FREQuency:STARt?

If [Sweep type is segment](#), this query returns the start frequency of the first segment.

**Return Type** Numeric

**Default** Minimum frequency of the analyzer

---

## SENSe<cnum>:FREQuency:STOP <num>

(Read-Write) Sets the stop frequency of the analyzer.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Stop frequency. Choose any number between: the **minimum** and **maximum** frequency limits of the analyzer. Units are Hz.
- If FREQ:STOP is set less than FREQ:START, then START will be set equal to STOP.
- This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:FREQ:STOP 1000000
sense2:frequency:stop max
```

**Query Syntax** SENSe<cnum>:FREQuency:STOP?

If [Sweep type is segment](#), this query returns the stop frequency of the last segment.

**Return Type** Numeric

**Default** Maximum frequency of the analyzer

Last Modified:

5-Feb-2008    Modified Max and Min notes

## SENSe:GCSetup Commands

---

Controls the Gain Compression configuration.

**SENSe:GCSetup:**

**AMODe <char>**

**COMPression:**

| **ALGorithm <char>**

| **BACKoff:LEVel <num>**

| **DELTA:X <num>**

| **DELTA:Y <num>**

| **INTerpolation <bool>**

| **LEVel <num>**

| **SATuration:LEVel**

**EOSoperation <string>**

**PMAP**

| **INPut?**

| **OUTPut?**

**POWer:**

| **LINear:INPut:LEVel <num>**

| **REVerse:LEVel <num>**

| **START:LEVel <num>**

| **STOP:LEVel <num>**

**SAFE:**

| **CPADjustment <num>**

| **ENABle <bool>**

| **FPADjustment <num>**

| **FTHReshold <num>**

| **MLIMit <num>**

**SFA?**

**SMART:**

| **MITerations** <num>

| **SITerations** <bool>

| **STIME** <num>

| **TOLerance** <num>

**SWEEP:**

| **FREQuency:POINts** <num>

| **POWER:POINts** <num>

Click on a [blue](#) keyword to view the command details.

### Other Gain Compression commands

The calibration commands listed in this topic are supplemental to the Guided Cal commands.

- [CALC:CUSTom:DEFine](#) - creates a gain compression measurement.
- [CALC:GCMeas:ANAL](#) - Gain Compression Analysis settings
- **GCA Calibration** uses the [Guided Calibration commands](#), except for the following:
- [Sens:Corr:GCS:Power](#) - sets power level for Source Power Cal

### See Also

- **Example Program:** [Create and Cal a Gain Compression Measurement](#)
- [Learn about Gain Compression Application](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

**SENSe<ch>:GCSetup:AMODE <char>**

**(Read-Write)** Set and read the method by which gain compression data is acquired.

**Parameters**

- <ch> Any existing GCA channel. If unspecified, value is set to 1.  
<char> Choose from:

- PFREQuency - 2D Power Per Frequency
- FPOWer - 2D Frequency Per Power
- SMARtsweep - Smart Sweep

**Examples**

```
SENS:GCS:AMOD SMAR  
sense:gcsetup:amode pfrequency
```

**Query Syntax** SENSE<ch>:GCSetup:AMODE ?

**Return Type** Character

**Default** SMARtsweep

---

**SENSe<ch>:GCSetup:COMPression:ALGorithm <char>**

**(Read-Write)** Set and read the algorithm method used to compute gain compression.

**Parameters**

- <ch> Any existing GCA channel. If unspecified, value is set to 1.  
<char> Algorithm method. Choose from:

- **CFLG** - compression from linear gain
- **CFMG** - compression from maximum gain
- **BACKoff** - compression from BackOff
- **XYCOM** - X/Y Compression
- **SAT** - compression from saturation

**Examples**

```
SENS:GCS:COMP:ALG BACK  
sense:gcsetup:compression:algorithm XYcom
```

**Query Syntax** SENSE<ch>:GCSetup:COMPression:ALGorithm?

**Return Type** Character

**Default** CFLG



---

### SENSe<ch>:GCSetup:COMPression:BACKoff:LEVel <num>

(Read-Write) Set and read value for the BackOff compression algorithm.

#### Parameters

- <ch> Any existing GCA channel. If unspecified, value is set to 1.  
<num> Backoff value in dB. Choose from 30 to (-30)

#### Examples

```
SENS:GCS:COMP:BACK:LEV 10  
sense:gcsetup:compression:backoff:level 5
```

Query Syntax SENSe<ch>:GCSetup:COMPression:BACKoff:LEVel?

Return Type Numeric

Default 10

---

### SENSe<ch>:GCSetup:COMPression:DELTA:X <num>

(Read-Write) Set and read the 'X' value in the delta X/Y compression algorithm.

#### Parameters

- <ch> Any existing GCA channel. If unspecified, value is set to 1.  
<num> X value in dB. Choose from 30 to (-30)

#### Examples

```
SENS:GCS:COMP:DELTA:X 9  
sense:gcsetup:compression:delta:X 8
```

Query Syntax SENSe<ch>:GCSetup:COMPression:DELTA:X?

Return Type Numeric

Default 10

---

### SENSe<ch>:GCSetup:COMPression:DELTA:Y <num>

**(Read-Write)** Set and read the 'Y' value in the delta X/Y compression algorithm.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Y value in dB. Choose from 30 to (-30)

**Examples**

```
SENS:GCS:COMP:DELT:Y 9
```

```
sense:gcsetup:compression:delta:Y 8
```

**Query Syntax** SENSE<ch>:GCSetup:COMPression:DELTa:Y?

**Return Type** Numeric

**Default** 9

---

**SENSe<ch>:GCSetup:COMPression:INTerpolation <bool>**

**(Read-Write)** Sets whether or not interpolation should be performed on 2D measured compression data. Applies ONLY to [2D acquisition modes](#).

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<bool> Choose from:

**ON** or **(1)** Interpolate the results

**OFF** or **(0)** Do NOT interpolate the results but return the value closest to compression.

**Examples**

```
SENS:GCS:COMP:INT 1
```

```
sense:gcsetup:compression:interpolation off
```

**Query Syntax** SENSE<ch>:GCSetup:COMPression:INTerpolation?

**Return Type** Boolean

**Default** OFF

---

**SENSe<ch>:GCSetup:COMPression:LEVel <num>**

**(Read-Write)** Set and read the desired gain reduction (from reference gain).

This value is used for Compression from Linear Gain and Compression from Maximum Gain.

Use [SENS:GCS:COMP:ALG CFLG](#) to set this compression algorithm.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Compression level in dB. Choose a value greater than 0.1 dB.

**Examples**

```
SENS:GCS:COMP:LEV 1
```

```
sense:gcsetup:compression:level 3
```

**Query Syntax** SENSE<ch>:GCSetup:COMPression:LEVel?

**Return Type** Numeric

**Default** 1

---

**SENSe<ch>:GCSetup:COMPression:SATuration:LEVel <num>**

**(Read-Write)** Set and read the deviation dB from the maximum Pout. This is the point of saturation.

Use [SENS:GCS:COMP:ALG CFLG](#) to set this compression algorithm.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Saturation level in dB. Choose a value greater than 0.01 dB.

**Examples**

```
SENS:GCS:COMP:SAT:LEV 1
```

```
sense:gcsetup:compression:saturation:level 3
```

**Query Syntax** SENSE<ch>:GCSetup:COMPressionSATuration:LEVel?

**Return Type** Numeric

**Default** .1 dB

---

**SENSe<ch>:GCSetup:EOSoperation <char>**

**(Read-Write)** Set and read the This setting is used to protect a sensitive device from too much power during the sweep retrace. Other instrument settings or channels may over-ride this setting. [Learn more.](#)

#### Parameters

- <ch> Any existing GCA channel. If unspecified, value is set to 1.
- <char> End Of Sweep operation. Choose from:
- STANdard Use the default PNA method. [Learn more.](#)
  - POFF Always turn power OFF while waiting.
  - PSTArt Sweep Start power
  - PSTOp Sweep Stop power.

#### Examples

```
SENS:GCS:EOS PSTA
sense:gcsetup:eosoperation standard
```

**Query Syntax** SENSE<ch>:GCSetup:EOSoperation?

**Return Type** Character

**Default** STANdard

---

### SENSe<ch>:GCSetup:PMAP <in>,<out>

**(Write-only)** Set and read the DUT-to-PNA port mapping for the Gain Compression measurement.

#### Parameters

- <ch> Any existing GCA channel. If unspecified, value is set to 1.
- <in> PNA port which is connected to the DUT input.
- <out> PNA port which is connected to the DUT output.

#### Examples

```
SENS:GCS:PMAP 1,2
sense:gcsetup:pmap 2,1
```

**Query Syntax** Not Applicable

**Default** 1,2

---

### SENSe<ch>:GCSetup:PMAP:INPut?

**(Read-only)** Read the PNA port number to be connected to the DUT Input.

Use [SENS:GCS:PORTMap](#) to set the port mapping.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

**Examples**

```
SENS:GCS:PMAP:INP?  
sense:gcsetup:pmap:input?
```

**Return Type** Numeric

**Default** 1

---

**SENSe<ch>:GCSetup:PMAP:OUTPut?**

**(Read-only)** Read the PNA port number to be connected to the DUT Output.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

**Examples**

```
SENS:GCS:PMAP:OUTP?  
sense:gcsetup:pmap:output?
```

**Return Type** Numeric

**Default** 2

---

**SENSe<ch>:GCSetup:POWer:LINear:INPut:LEVel <num>**

**(Read-Write)** Set and read the input power at which Linear Gain and all S-parameters are measured.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Input power level in dBm. Choose a value from +30 to (-30).

**Examples**

```
SENS:GCS:POW:LIN:INP:LEV 0  
sense:gcsetup:power:linear:input:level -10
```

**Query Syntax** SENSe<ch>:GCSetup:POWer:LINear:INPut:LEVel?

**Return Type** Numeric

**Default** -25 dBm

---

### SENSe<ch>:GCSetup:POWer:REVerse:LEVel <num>

(Read-Write) Set and read the reverse power level to the DUT. This is applied to the DUT output port when making reverse measurements like S22.

#### Parameters

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Reverse power level in dBm. Choose a value from +30 to (-30).

#### Examples

```
SENS:GCS:POW:REV:LEV 0
```

```
sense:gcsetup:power:reverse:level -5
```

Query Syntax SENSe<ch>:GCSetup:POWer:REVerse:LEVel?

Return Type Numeric

Default -5

---

### SENSe<ch>:GCSetup:POWer:STARt:LEVel <num>

(Read-Write) Set and read the start power level.

#### Parameters

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Start power level in dBm. Choose a value from +30 to (-30).

#### Examples

```
SENS:GCS:POW:STAR:LEV 0
```

```
sense:gcsetup:power:start:level -5
```

Query Syntax SENSe<ch>:GCSetup:POWer:STARt:LEVel?

Return Type Numeric

Default -30

---

### SENSe<ch>:GCSetup:POWer:STOP:LEVel <num>

**(Read-Write)** Set and read the stop power level.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Stop power level in dBm. Choose a value from +30 to (-30).

**Examples**

```
SENS:GCS:POW:STOP:LEV 0
```

```
sense:gcsetup:power:stop:level -5
```

**Query Syntax** SENSE<ch>:GCSetup:POWer:STOP:LEVel?

**Return Type** Numeric

**Default** -5

---

**SENSe<ch>:GCSetup:SAFE:CPADjustment <num>**

**(Read-Write)** Set and read the Safe Sweep COARSE power adjustment. [Learn more.](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Coarse power adjustment setting in dBm. Choose a value from +30 to (-30).

**Examples**

```
SENS:GCS:SAFE:CPAD 2
```

```
sense:gcsetup:safe:cpadjustment 3.5
```

**Query Syntax** SENSE<ch>:GCSetup:SAFE:CPADjustment?

**Return Type** Numeric

**Default** 3.0

---

**SENSe<ch>:GCSetup:SAFE:ENABLE <bool>**

**(Read-Write)** Set and read the (ON | OFF) state of Safe Sweep mode. [Learn more](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> **(Boolean)** - Safe Sweep state. Choose from:

**OFF** (or **0**) - Disable Safe Sweep

**ON** (or **1**) - Enable Safe Sweep

**Examples**

```
SENS:GCS:SAFE:ENAB 0
```

```
sense:gcsetup:safe:enable 1
```

**Query Syntax** SENSE<ch>:GCSetup:SAFE:ENABLE?

**Return Type** Boolean

**Default** 0

---

**SENSe<ch>:GCSetup:SAFE:FPADjustment <num>**

**(Read-Write)** Set and read the Safe Sweep FINE power adjustment. [Learn more](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Fine power adjustment setting in dBm. Choose a value from +30 to (-30).

**Examples**

```
SENS:GCS:SAFE:FPAD 2
```

```
sense:gcsetup:safe:fpadjustment .5
```

**Query Syntax** SENSe<ch>:GCSetup:SAFE:FPADjustment?

**Return Type** Numeric

**Default** 1.0 dBm

---

**SENSe<ch>:GCSetup:SAFE:FTHReshold <num>**



**(Read-Write)** Set and read the compression level in which Safe Sweep changes from the COARSE power adjustment to the FINE power adjustment. [Learn more](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Threshold setting in dB. Choose a value from +30 to (-30).

**Examples**

```
SENS:GCS:SAFE:FTHR .1
```

```
sense:gcsetup:safe:fthreshold .75
```

**Query Syntax** SENSE<ch>:GCSetup:SAFE:FTHReshold?

**Return Type** Numeric

**Default** 0.5 dB

---

**SENSe<ch>:GCSetup:SAFE:MLimit <num>**

**(Read-Write)** When the PNA port that is connected to the DUT Output measures this value, the input power to the DUT is no longer incremented at that frequency. Safe Mode must be enabled with [SENS:GCS:SAFE:ENAB ON](#) [Learn more](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Maximum power limit in dBm. Choose a value from -100 to +100

**Examples**

```
SENS:GCS:SAFE:MLIM 20
```

```
sense:gcsetup:safe:mlimit 30
```

**Query Syntax** SENSe<ch>:GCSetup:SAFE:MLIMit?

**Return Type** Numeric

**Default** 30

---

**SENSe<ch>:GCSetup:SFA?**

**(Read-only)** Returns a comma-separated list of the frequency indexes that were out of tolerance for SMART Sweep mode, or at the power limit for 2D acquisition modes. Zero (0) is the first frequency data point.

Must be Single triggered. Invalid results occur if the GCA channel is continuously sweeping.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

**Examples**

```
SENS:GCS:SFA?
```

```
sense:gcsetup:sfa?
```

**Return Type** Comma-separated list of frequency indexes.

**Default** Not Applicable

---

**SENSe<ch>:GCSetup:SMART:MITerations <num>**

**(Read-Write)** Set and read the maximum permitted number of iterations which SMART Sweep may utilize to find the desired compression level, to within the specified tolerance.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Maximum number of iterations. Choose a value between 1 and 10e8

**Examples**

```
SENS:GCS:SMAR:MIT 5
```

```
sense:gcsetup:smart:miterations 3
```

**Query Syntax** SENSe<ch>:GCSetup:SMART:MITerations?

**Return Type** Numeric

**Default** 20

---

**SENSe<ch>:GCSetup:SMART:SITerations <bool>**

**(Read-Write)** Set and read enable for showing intermediate results for each iteration of SMART Sweep

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<bool> Choose from:

**ON** or **(1)** Compression traces are updated after each iteration.

**OFF** or **(0)** Compression traces are updated after ALL iterations are complete.

**Examples**

```
SENS:GCS:SMAR:SIT 1
```

```
sense:gcsetup:smart:siterations off
```

**Query Syntax** SENSE<ch>:GCSetup:SMART:SITerations?

**Return Type** Boolean

**Default** OFF

---

**SENSe<ch>:GCSetup:SMART:STIME <num>**

**(Read-Write)** Set and read the amount of time SMART Sweep will dwell at the first point where the input power changes by the Backoff or X level. Applies only to SMART Sweep when Backoff or XY compression methods are selected. [Learn more.](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<bool> Settling time in seconds. Choose any positive value.

**Examples**

```
SENS:GCS:SMAR:STIM 1
```

```
sense:gcsetup:smart:stime .1
```

**Query Syntax** SENSe<ch>:GCSetup:SMART:STIME?

**Return Type** Numeric

**Default** 0

---

**SENSe<ch>:GCSetup:SMART:TOLerance <num>**

**(Read-Write)** Set and read the acceptable range SMART Sweep will allow for the measured compression level.

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Tolerance level in dBm. Choose a value between .01 and 10

**Examples**

```
SENS:GCS:SMAR:TOL .1
```

```
sense:gcsetup:smart:tolerance .05
```

**Query Syntax** SENSE<ch>:GCSetup:SMART:TOLerance?

**Return Type** Numeric

**Default** .05

---

**SENSe<ch>:GCSetup:SWEep:FREQuency:POINts <num>**

**(Read-Write)** Set and read the number of data points in each frequency sweep. [Learn more](#)

**Parameters**

<ch> Any existing GCA channel. If unspecified, value is set to 1.

<num> Frequency points. Do not exceed the max number of data points.

[See Data Points Limit](#)

**Examples**

```
SENS:GCS:SWE:FREQ:POIN 201
```

```
sense:gcsetup:sweep:frequency:points 101
```

**Query Syntax** SENSe<ch>:GCSetup:SWEep:FREQuency:POINts?

**Return Type** Numeric

**Default** 201

---

**SENSe<ch>:GCSetup:SWEep:POWer:POINts <num>**

**(Read-Write)** Set and read the number of data points in each power sweep. Applies ONLY to 2D [acquisition modes](#).

**Parameters**

- <ch> Any existing GCA channel. If unspecified, value is set to 1.
- <num> Power points. Do not exceed the max number of data points.

[See Data Points Limit](#)

**Examples**

```
SENS:GCS:SWE:POW:POIN 50  
sense:gcsetup:sweep:power:points 21
```

**Query Syntax** SENSE<ch>:GCSetup:SWEep:POWer:POINts?

**Return Type** Numeric

**Default** 26

---

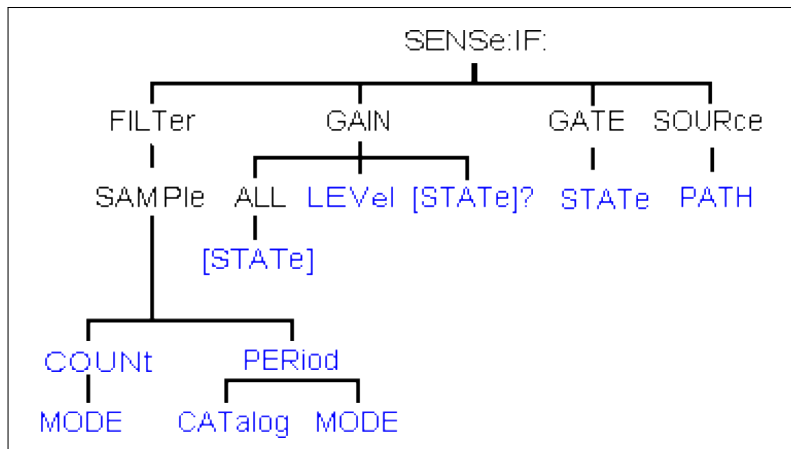
Last Modified:

9-Nov-2007    MX New topic

## Sense:IF Commands

Controls the IF source and gain settings for use with the **E836xC H11** Option.

**Note:** See the [Sense:IF commands](#) for the PNA-X.



Click on a [blue](#) keyword to view the command details.

### See Also

- Learn about the [Narrowband \(Pulse\) Application](#)
- See [IF Access User Interface Settings](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

## SENSe<num>:IF:FILTer:SAMPlE:COUNt <num>

**(Read-Write)** Sets and returns the number of taps in the IF filter. The IF filter sample count setting is only used when [SENSe:IF:FILTer:SAMPlE:COUNt:MODE](#) is set to MANUAL.

For use with the **E836xC Opt H11**

### Parameters

- <num> Existing channel number to manipulate. If unspecified, <num> is set to 1.
- <num> An integer value. (MIN and MAX return the minimum and maximum allowed values, respectively.)

**Examples** `SENS:IF:FILT:SAMP:COUN 40`  
`sense2:if:filter:sample:count maximum`

**Query Syntax** `SENS:IF:FILTer:SAMPlE:COUNt?`

**Return Type** Numeric

**Default** Instrument dependent.

## SENSe<num>:IF:FILTer:SAMPlE:COUNT:MODE <char>

(Read-Write) Sets and returns the IF filter sample count mode to the specified value. When in MANUAL mode, the value specified for the [IF Filter sample count](#) is used as the number of taps in the IF filter.

For use with the **E836xC Opt H11**

### Parameters

<num> Existing channel number to manipulate. If unspecified, <num> is set to 1.

<char> Choose either AUTO or MANUAL.

**Examples** `SENS:IF:FILT:SAMP:COUN:MODE MANUAL`  
`sense2:if:filter:sample:count:mode AUTO`

**Query Syntax** SENS:IF:FILTer:SAMPlE:COUNT:MODE?

**Return Type** Character

**Default** AUTO

---

## SENSe<num>:IF:FILTer:SAMPlE:PERiod <num>

(Read-Write) Sets and returns the IF filter sample period. The IF filter sample period setting is only used by the instrument when the [SENS:IF:FILT:SAMP:PER:MODE](#) is set to MANUAL.

For use with the **E836xC Opt H11**

### Parameters

<num> Existing channel number to manipulate. If unspecified, <num> is set to 1.

<num> Sample period. Choose from values returned from the [SENS:IF:FILTer:SAMPlE:PERiod:CAT?](#) command.

**Examples** `SENS:IF:FILT:SAMP:PER 6 us`  
`sense2:if:filter:sample:period maximum`

**Query Syntax** SENS:IF:FILTer:SAMPlE:PERiod?

**Return Type** Numeric

**Default** Instrument dependent.

---

## SENSe<num>:IF:FILTer:SAMPlE:PERiod:CATalog?

(Read-Only) Returns the list of allowed IF filter sample periods for this instrument.

For use with the **E836xC Opt H11**

**Parameters**

<num> Existing channel number to manipulate. If unspecified, <num> is set to 1.

**Examples** `SENS:IF:FILT:SAMP:PER:CAT?`

`sense2:if:filter:sample:period:catalog?`

**Query Syntax** `SENS:IF:FILT:er:SAMPle:PERiod:CATalog?`

**Return Type** String

Default AUTO

---

**SENSe<num>:IF:FILT:er:SAMPle:PERiod:MODE <char>**

(Read-Write) Sets and returns the IF filter sample period mode to the specified value.

For use with the **E836xC Opt H11**

**Parameters**

<num> Existing channel number to manipulate. If unspecified, <num> is set to 1.

<char> Sample period mode. Choose from **AUTO** or **MANUAL**.

**Examples** `SENS:IF:FILT:SAMP:PER:MODE MANUAL`

`sense2:if:filter:sample:period:mode AUTO`

**Query Syntax** `SENS:IF:FILT:er:SAMPle:PERiod:MODE?`

**Return Type** Character

Default AUTO

---

**SENSe<num>:IF:GAIN:ALL[:STATe] <char>**

(Write only) Sets the gain state for ALL receivers to Auto or Manual.

For use with the **E836xC Opt H11**

**Parameters**

<num> Existing channel number to manipulate. If unspecified, <num> is set to 1.

<char> Choose from **AUTO** or **MANUAL**

**Examples** `SENS:IF:GAIN:ALL AUTO`

`sense:if:gain:all:state manual`

**Query Syntax** Not Applicable

**Return Type** Character



Default AUTO

---

## SENSe<cnum>:IF:GAIN:LEVel <id>, <level>

(Read Write) Manually sets the gain level for the specified receiver. This setting can be made for E836x and selected N5230A models. [Learn more.](#)

### Parameters

<cnum> Existing channel number to manipulate. If unspecified, <cnum> is set to 1.

<id> (String) Choose from: 'A', 'B', 'R1', 'R2'..

<level> Gain level. Choose from:

0 - Lowest gain setting

1

2 - Highest gain setting

**Examples** SENS:IF:GAIN:LEVel 'A', 1

**Query Syntax** SENSe<cnum>:IF:GAIN:LEVel? <id>

**Return Type** Numeric

Default 0

---

## SENSe<cnum>:IF:GAIN[:STATE?], <id>

(Read only) Returns the gain state for the specified receiver. Use [SENS:IF:GAIN:ALL](#). to set the gain state for all receivers in the channel. This setting can be made for E836x and selected N5230A models.

### Parameters

<cnum> Existing channel number to manipulate. If unspecified, <cnum> is set to 1.

<id> Choose from: 'A', 'B', 'R1', 'R2'.

**Note:** The A and R1 receivers are always switched together. B and R2 are also always switched together. For example, if you specify 'A', R1 will also be switched.

**Examples** SENS:IF:GAIN? 'A'

**Return Type** Character - Returns **AUTO** or **MAN**. See [SENS:IF:GAIN:ALL](#).

Default AUTO

## SENSe<cnum>:IF:GATE[:STATe] <boolean>

(Read-Write) Sets or returns the IF gate state.

For use with the **E836xC Opt H11**

### Parameters

<cnum> Existing channel number to manipulate. If unspecified, <cnum> is set to 1.

<boolean> Choose from ON or OFF

### Examples

```
SENS:IF:GATE ON
sense2:if:gate:state 0
```

**Query Syntax** SENS:IF:GATE[:STATe]?

**Return Type** Boolean

Default

---

## SENSe<cnum>:IF:SOURce:PATH <id>, <char>

(Read Write) Sets the source path for the specified receiver. An error is returned if <id> is not valid, or if <char> is not valid for the specified <id>.

For use with the **E836xC Opt H11**

### Parameters

<cnum> Existing channel number to manipulate. If unspecified, <cnum> is set to 1.

<id> Choose from: 'A', 'B', 'R1', 'R2'

**Note:** The A and R1 receivers are always switched together. B and R2 are also always switched together. For example, if you specify "A", R1 will also be switched.

<char> Choose from:

- **NORMal** - the PNA decides the appropriate IF input paths.
- **EXTernal** - always use the rear panel IF inputs.

### Examples

```
SENS:IF:SOUR:PATH 'A', Ext
```

**Query Syntax** SENSe<cnum>:IF:SOURce:PATH? <id>

**Return Type** Character

Default Normal

---

Last Modified:

15-Mar-2010 Clarified E836x models

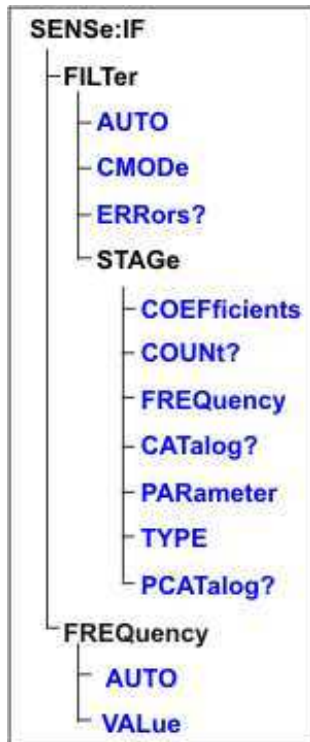
14-Apr-2008 Clarified gain settings.

## Sense:IF Commands

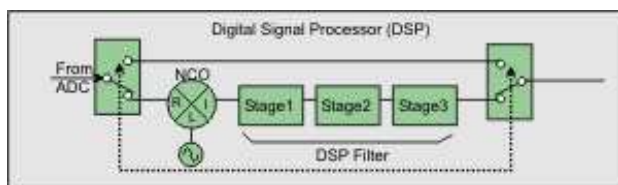
---

Controls the DSP filters for use with the PNA X.

See the [Sense:IF commands](#) for the E836xC models.



- Click on a [blue](#) keyword to view the command details.
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)



### PNA-X DSP Block diagram

All of the Sense:IF commands, except the frequency commands, make settings related to the DSP section of the IF Path.

See the entire [IF Path Configuration](#).

**Note:** For any of the Filter "Stage" parameters to take effect, [SENS:IF:FILT:AUTO](#) must be set to OFF (MANUAL) and mode, and [SENS:IF:FILT:CMOD](#) must be set to OFF.

**Critical Note:** These commands act on the selected measurement. You can select one measurement for each channel using [Calc:Par:Select](#)

---

### **SENSe<cnum>:IF:FILTer:AUTO <bool>**

**(Read-Write)** Sets and returns whether the PNA configures the 3-stage digital filter settings or they will be configured manually. When making manual settings, also send [SENS:IF:FILT:CMOD OFF](#) which routes the IF through the 3-stage filter.

#### Critical Note

##### **Parameters**

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

<bool> (Boolean)

**ON** (or **1**) - Automatic: PNA controls digital filter settings.

**OFF** (or **0**) - Manual: You control digital filter settings using other Sens:IF commands.

##### **Examples**

```
SENS:IF:FILT:AUTO 1  
sense2:if:filter:auto 0
```

##### **Query Syntax**

SENSe<cnum>:IF:FILTer:AUTO?

##### **Return Type**

Boolean

##### **Default**

ON

---

### **SENSe:IF:FILTer:CMODE <bool>**

**(Read-Write)** Sets and returns the ADC capture mode modeled as a 2-pole switch in the above diagram. The switch either bypasses or routes the IF through the 3-stage digital filter.

#### Critical Note

##### Parameters

<bool> (Boolean)

**ON (or 1)** - The digital filters are bypassed and the raw ADC readings are taken directly. A maximum of 4096 data points per sweep can be acquired.

**OFF (or 0)** - The digital filters are used to process IF information. The filters can be configured automatically or manually using [SENS:IF:FILT:AUTO](#).

##### Examples

```
SENS:IF:FILT:CMOD 1  
sense2:if:filter:cmode 0
```

##### Query Syntax

SENSe<cnum>:IF:FILTer:CMODE?

##### Return Type

Boolean

##### Default

OFF

---

#### **SENSe<cnum>:IF:FILTer:ERRors?**

**(Read-only)** Returns the error string associated with the digital filters. The return string has three fields separated by commas: "stage1 status, stage2 status, stage3 status"

Each of these fields can contain one or more of the following error codes:

- **NO ERROR**
- **\*NUMBER-OF-COEFFICIENTS** - the number of coefficients is excessive for that filter-stage
- **\*COEFFICIENT VALUE** - one or more coefficients are out of range for that filter-stage
- **\*SUM-OF-COEFFICIENTS** - the sum of all coefficients is excessive for that filter-stage,
- **\*FREQUENCY** - the frequency for Stage 1 is out of range (only applies stage1 field),
- **\*PARAMETER** - one or more parameters are out of range (only applies to stage 3 field)

#### Critical Note

##### Parameters

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

##### Examples

```
SENS:IF:FILT:ERR?  
  
'example return strings"  
  
NO ERROR, NO ERROR, NO ERROR
```

indicates no errors,

\*SUM-OF-COEFFICIENTS, NO ERROR, NO ERROR

indicates that the sum of all filter coefficients exceed the maximum value for the Stage-1 filter,

\*COEFFICIENT \*SUM-OF-COEFFICIENTS, NO ERROR, \*PARAMETER

indicates a problems with Stage 1 coefficients and a problem with one or more of the parameters associated with the Stage 3 filter.

**Return Type** String  
**Default** Not Applicable

---

## **SENSe<cnum>:IF:FILTer:STAGE<n>:COEFficients <coef>**

(Read-Write) Sets and returns the digital filter coefficients of the specified stage.

### Critical Note

#### **Parameters**

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.  
<n> Stage number. Choose 1 or 2  
<coef> An array of real numbers. Filter coefficients

#### **Examples**

```
SENS:IF:FILT:STAG2:COEF 0,0.1,0.7,0.7,0.1
```

```
sense2:if:filter:stage1:coefficients
```

#### **Query Syntax**

```
SENSe<cnum>:IF:FILTer:STAGE<n>:COEFficients?
```

#### **Example**

```
SENS:IF:FILT:STAG2:COEF?
```

```
sense2:if:filter:stage1:coefficients?
```

**Return Type** Numeric  
**Default** Stage dependent

---

## **SENSe<cnum>:IF:FILTer:STAGE<n>:COUNT? [char]**

**(Read-only)** Returns the number of taps in the digital filter of the specified stage. The filter sample count setting is only used when [SENSe:IF:FILTer:AUTO](#) is set to False (MANUAL).

#### Critical Note

##### Parameters

<num> Existing channel number. If unspecified, <num> is set to 1.  
<n> Stage number. Choose 1 or 2  
[char] Optional parameter. Choose from:  
not specified - returns the current number of coefficients for the specified stage.

**MIN** - returns the minimum number of coefficients for the specified stage. Stage1: **10**, Stage2: **1**

**MAX** - returns the maximum number of coefficients for the specified stage. Stage1 & 2: **1024**

##### Examples

```
SENS:IF:FILT:STAG2:COUN?  
sense2:if:filter:stage1:count? max
```

##### Return Type

Numeric

##### Default

Stage dependent

---

#### **SENSe<num>:IF:FILTer:STAGE1:FREQuency <value>**

**(Read-Write)** Sets and returns the Numerically Controlled Oscillator (NCO) frequency. This command is only used when [SENSe:IF:FILTer:AUTO](#) is set to False (Manual).

#### Critical Note

##### Parameters

<num> Existing channel number. If unspecified, <num> is set to 1.  
<value> Stage 1 Frequency. Min value= 0 Hz, Max value= 15 MHz. Or programmatically use the Max and Min queries to determine the range of settable values.

##### Examples

```
SENS:IF:FILT:STAG1:FREQ 9e6  
sense2:if:filter:stage1:frequency 9.2e6
```

##### Query Syntax

SENSe<num>:IF:FILTer:STAGE1:FREQuency?

'returns the current parameter value

SENSe<num>:IF:FILTer:STAGE1:FREQuency? Min

'returns the minimum frequency value.



SENSe<cnum>:IF:FILTer:STAGE1:FREQuency? Max

'returns the maximum frequency value.

**Return Type** Numeric

**Default** 9 MHz

---

### SENSe<cnum>:IF:FILTer:STAGE3:CATalog?

(Read-only) Returns a list of strings for the currently supported filter types that can be used for the stage 3 filter. This command is only used when [SENSe:IF:FILTer:AUTO](#) is set to False (Manual). See [SENS:IF:FILT:STAGE3:TYPE](#) for a list of currently supported filter types.

#### Critical Note

##### Parameters

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

##### Examples

```
SENS:IF:FILT:STAGE3:CAT?  
sense2:if:filter:stage3:catalog?
```

**Return Type** String array

**Default** Not Applicable

---

### SENSe<cnum>:IF:FILTer:STAGE3:TYPE <value>

(Read-Write) Sets and returns the Stage 3 filter type. This command is only used when [SENSe:IF:FILTer:AUTO](#) is set to False (Manual).

#### Critical Note

##### Parameters

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

<value> (String) Filter type. Chose from:

- "RECT" Rectangular Window Filter
- "TUKEY" Tukey Filter
- "PWIN" Pulse Window Filter

##### Examples

```
SENS:IF:FILT:STAGE3:TYPE RECT  
sense2:if:filter:stage3:type pwin
```

**Query Syntax** SENSe<cnum>:IF:FILTer:STAGE3:TYPE?

**Return Type** String

**Default** "TUKEY"

---

## **SENSe<cnUm>:IF:FiLTer:STAGe3:PARAmeter <p>, <value>**

**(Read-Write)** Sets and returns the Stage 3 filter parameters.

Must first select the filter type ([SENS:IF:FILT:STAGe3:TYPE](#)) before setting these parameters

Use [SENSe:IF:FILT:STAGe3:PCAT?](#) to return a list of the available parameters for the currently selected filter type.

### Critical Note

#### **Parameters**

<cnUm> Existing channel number. If unspecified, <cnUm> is set to 1.

<p> (String) Filter parameter. Choose from:

"C" - Tap count (Tukey, RECT, PWIN)

"P" - Period (PWIN ONLY)

"D" - Delay (PWIN ONLY)

"W" - Width (PWIN ONLY)

"R" - Ramp Count (PWIN ONLY)

<value> (String) Parameter Value for the specified stage 3 parameter. Use the query form to return the minimum and maximum values for the specified parameter.

#### **Examples**

```
SENS:IF:FILT:STAGe3:PAR "C", 64
sense2:if:filter:stage3:parameter "d", 0.5E-6
```

#### **Query Syntax**

SENSe<cnUm>:IF:FiLTer:STAGe3:PARAmeter? <p>

returns the current parameter value

SENSe<cnUm>:IF:FiLTer:STAGe3:PARAmeter? <p>,Min

returns the minimum parameter value.

SENSe<cnUm>:IF:FiLTer:STAGe3:PARAmeter? <p>,Max

returns the maximum parameter value.

#### **Examples**

```
SENS:IF:FILT:STAGe3:PAR? "C"
sense2:if:filter:stage3:parameter? "d",min
```

#### **Return Type**

Numeric

**Default** RECT: C = 1

PWIN: C=1E6, P=10ms, D=50us, W=50us, R=7

TUKEY: C=1

---

### SENSe<cnum>:IF:FILTer:STAGe3:PCATalog?

(Read-only) Returns a list of the available parameters for the currently selected filter type.

#### Critical Note

##### Parameters

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

##### Examples

```
SENS:IF:FILT:STAGe3:PCAT?  
sense2:if:filter:stage3:pcatalog?
```

Return Type String

**Default** Not Applicable

---

### SENSe<cnum>:IF:FREQuency:AUTO <bool>

(Read-Write) Sets and returns the method for specifying the way the IF Frequency is determined.

#### Critical Note

##### Parameters

<cnum> Existing channel number. If unspecified, <cnum> is set to 1.

<bool> (Boolean)

ON (or 1) - Automatic. PNA determines the setting for the IF frequency.

OFF (or 0) - Manual. Use [SENS:IF:FREQ](#) to set the frequency.

##### Examples

```
SENS:IF:FREQ:AUTO 1  
sense2:if:frequency:auto 0
```

Query Syntax SENSe<cnum>:IF:FREQuency:AUTO?

Return Type Boolean

**Default** ON

---

### SENSe<cnum>:IF:FREQuency[:VALue] <value>

**(Read-Write)** Sets and returns the IF frequency for ALL receiver paths being used for the specified channel. To set this frequency, [SENS:IF:FREQ:AUTO](#) must be set to OFF (Manual).

Also returns the maximum and minimum allowable frequency settings.

### Critical Note

#### **Parameters**

<num> Existing channel number. If unspecified, <num> is set to 1.  
<value> (Numeric) Frequency value.

#### **Examples**

```
SENS:IF:FREQ  
sense2:if:frequency
```

#### **Query Syntax**

SENSe<num>:IF:FREQuency?

'returns the current frequency setting

SENSe<num>:IF:FREQuency? Max

'returns the maximum allowable frequency setting

SENSe<num>:IF:FREQuency? Min

'returns the minimum allowable frequency setting

#### **Return Type**

Numeric

#### **Default**

9 MHz

---

Last Modified:

18-Jan-2007    MX New topic

## Sense:IMD Commands

---

Controls an IMD or IMDx measurement configuration.

### SENSe:IMD

#### SWEEP:TYPE

##### CSO:

|NDPProducts

|NORMalized:POWer

|OFFSet

##### CTB

|NCARriers

|NORMalized:POWer

|OFFSet

##### FREQuency

|DFREquency

|[:CW]

|STARt

|STOP

|F1[:CW]

|F2[:CW]

|FCENter

|[:CW]

|STARt

|STOP

|CENTer

|SPAN

#### HOPRduct?

#### IFBWidth

| [MAIN](#)

| [IMTone](#)

**NORMalized**

| [MODE](#)

**PMAP**

| [INPut](#)

| [OUTPut](#)

**TPOWer**

| [COUPle\[:STATe\]](#)

| [F1](#)

| [F2](#)

| [F1:START](#)

| [F1:STOP](#)

| [F2:START](#)

| [F2:STOP](#)

Click on a [blue](#) keyword to view the command details.

### Other Swept IMD commands

- [CALC:CUSTom:DEFine](#) - creates a gain compression measurement.
- [Swept IMD Calibration](#) - these are supplemental to the [Guided Cal](#) commands

### See Also

- **Example** [Create and Cal an IMD Measurement](#)
  - [IM Spectrum commands](#)
  - [Learn about Swept IMD](#)
  - [Learn about Measurement Class](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
-

## SENSe<cnum>:IMD:SWEep:TYPE <char>

(Read-Write) Sets and returns the sweep type for the IMD measurement.

### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<char> Sweep type. Choose from:

- **FCENter** - (Center Frequency) Maintaining a constant tone spacing (DeltaF) and tone powers (P1 and P2), the center frequency (FC) is swept from Start to Stop.
- **DFrequency** - (Delta Frequency) The center frequency (FC) is held constant. The tone spacing is increased from StartDeltaF to StopDeltaF.
- **POWer** - The main tone frequencies are specified as either F1 and F2, or as FC and DeltaF. These frequencies are held constant while the power of each tone is stepped from the Start Power to the Stop Power.
- **CW** - The main tone frequencies (F1 and F2) and power levels (P1 and P2) are held constant. Measurements are taken for the specified Number of Points.
- **SEGment** - **Not available for IMDx**. Same as FCenter sweep, except that the center frequencies for the sweep are constructed using the standard [segment sweep commands](#).
- **LOPower** - All frequencies are fixed while the LO power is swept from Start to Stop power.

For each sweep type, use the commands that follow:

### FCENter:

- [SENS:IMD:FREQ:FCEN:STAR](#)
- [SENS:IMD:FREQ:FCEN:STOP](#)
- [SENS:IMD:FREQ:FCEN:CEN](#)T
- [SENS:IMD:FREQ:FCEN:SPAN](#)
- [SENS:IMD:FREQ:DFR:\[CW\]](#)
- [SENS:IMD:TPOW:F1](#)
- [SENS:IMD:TPOW:F2](#)

### DFrequency

- [SENS:IMD:FREQ:DFR:STAR](#)

- [SENS:IMD:FREQ:DFR:STOP](#)
- [SENS:IMD:FREQ:FCEN:\[CW\]](#)
- [SENS:IMD:TPOW:F1](#)
- [SENS:IMD:TPOW:F2](#)

## **POWer**

- [SENS:IMD:FREQ:F1:\[CW\]](#)
- [SENS:IMD:FREQ:F2:\[CW\]](#)
- [SENS:IMD:FREQ:FCEN:\[CW\]](#)
- [SENS:IMD:FREQ:DFR:\[CW\]](#)
- [SENS:IMD:TPOW:F1:STAR](#)
- [SENS:IMD:TPOW:F1:STOP](#)
- [SENS:IMD:TPOW:F2:STAR](#)
- [SENS:IMD:TPOW:F2:STOP](#)

## **CW**

- [SENS:IMD:FREQ:F1:\[CW\]](#)
- [SENS:IMD:FREQ:F2:\[CW\]](#)
- [SENS:IMD:FREQ:FCEN:\[CW\]](#)
- [SENS:IMD:FREQ:DFR:\[CW\]](#)
- [SENS:IMD:TPOW:F1](#)
- [SENS:IMD:TPOW:F2](#)

## **SEGMENT - Not available for IMDx.**

- [SENS:IMD:FREQ:DFR:\[CW\]](#)
- [SENS:IMD:TPOW:F1](#)
- [SENS:IMD:TPOW:F2](#)
- Use the standard [segment sweep commands](#) to set freq start and stop

## **LOPower - IMDx ONLY**



- [SENS:MIX:LO:POW:STAR](#)
- [SENS:MIX:LO:POW:STOP](#)
- [SENS:IMD:FREQ:F1:\[CW\]](#)
- [SENS:IMD:FREQ:F2:\[CW\]](#)
- [SENS:IMD:FREQ:FCEN:\[CW\]](#)
- [SENS:IMD:FREQ:DFR:\[CW\]](#)
- [SENS:IMD:TPOW:F1](#)
- [SENS:IMD:TPOW:F2](#)

**Examples**

```
SENS:IMD:SWEep:TYPE CW
sense2:imd:sweep:type power
```

**Query Syntax** SENSE<cnum>:IMD:SWEep:TYPE?

**Return Type** Character

**Default** FCENTer

## SENSe<cnum>:IMD:CSO:NDPRoducts <num>

**(Read-Write)** Sets and returns the “N = number of distortion products” value for the calculation of the CSO parameter.  
[Learn more.](#)

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Number of distortion products.

**Examples**

```
SENS:IMD:CSO:NDPR 30
sense2:imd:cso:ndproducts 7
```

**Query Syntax** SENSe<cnum>:IMD:CSO:NDPRoducts?

**Return Type** Numeric

**Default** 40

## SENSe<cnum>:IMD:CSO:NORMalized:POWer <num>

**(Read-Write)** Sets and returns the CSO Power for POWER normalization mode. Valid only with measurement parameters: CSO2Lo and CSO2Hi and for Normalization Modes **dBm** and **dBmV**. [Learn more.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Power level. The units are determined by [Sens:IMD:Norm:Mode](#), which must be set first.

**Examples**

```
SENS:IMD:CSO:NORM:POW 0  
  
sense2:imd:cso:normalized:power -5
```

**Query Syntax** SENSE<num>:IMD:CSO:NORMALized:POWer?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:IMD:CSO:OFFSet <num>**

**(Read-Write)** Sets and returns the offset that is applied to CSO measurements. Valid only with measurement parameters: CSO2Lo and CSO2Hi. [Learn more.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Offset value in dBm

**Examples**

```
SENS:IMD:CSO:OFFS 3  
  
sense2:imd:cso:offset 7
```

**Query Syntax** SENSE<num>:IMD:CSO:OFFSet?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:IMD:CTB:NCARriers <num>**

**(Read-Write)** Sets and returns the “N = Total number of carriers” value used in the calculation of the XMOD parameter. [Learn more.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Number of carriers.

**Examples**

```
SENS:IMD:CTB:NCAR 10  
sense2:imd:ctb:ncarriers 50
```

**Query Syntax** SENSE<num>:IMD:CTB:NCARriers?

**Return Type** Numeric

**Default** 40

---

**SENSe<num>:IMD:CTB:NORMalized:POWer <num>**

**(Read-Write)** Sets and returns the CTB Power. Valid only with measurement parameters: CTBLo and CTBHi and for Normalization Modes **dBm** and **dBmV**. [Learn more.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Power level. The units are determined by [Sens:IMD:Norm:Mode](#), which must be set first.

**Examples**

```
SENS:IMD:CTB:NORM:POW 0  
sense2:imd:ctb:normalized:power -5
```

**Query Syntax** SENSE<num>:IMD:CTB:NORMalized:POWer?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:IMD:CTB:OFFSet <num>**

**(Read-Write)** Sets and returns the offset that is applied to CTB measurements. Valid only with measurement parameters: CTB, CTBLo, CTBHi, CTBE, CTBLo, and CTBEHi. [Learn more.](#)

**Parameters**

<num> Any existing channel number; if unspecified, value is set to 1.

<num> Offset value in dBm

**Examples**

```
SENS:IMD:CTB:OFFS 3
```

```
sense2:imd:ctb:offset 7
```

**Query Syntax** SENSE<num>:IMD:CTB:OFFSet?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:IMD:FREQuency:DFRequency[:CW] <num>**

**(Read-Write)** Sets and returns fixed tone spacing.

**Parameters**

<num> Any existing channel number; if unspecified, value is set to 1.

<num> Tone spacing frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:DFR 1e6
```

```
sense2:imd:frequency:dfrequency:cw 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:DFRequency[:CW]?

**Return Type** Numeric

**Default** 1 MHz

---

**SENSe<num>:IMD:FREQuency:DFRequency:START <num>**

**(Read-Write)** Sets and returns the starting main tone separation for sweep type = DFRequency (delta frequency).

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Starting tone separation between F1 and F2 in Hz. Both F1 and F2 tones MUST be within the frequency range of the PNA where:

$$F1 \text{ (start)} = \text{FREQ:FCEN} - \text{DFR:Start} / 2$$

$$F2 \text{ (start)} = \text{FREQ:FCEN} + \text{DFR:Start} / 2$$

**Examples**

```
SENS:IMD:FREQ:DFR:STAR 1e6  
sense2:imd:frequency:dfrequency:start 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:DFRequency:STARt?

**Return Type** Numeric

**Default** 1 MHz

---

**SENSe<num>:IMD:FREQuency:DFRequency:STOP <num>**

**(Read-Write)** Sets and returns the stopping main tone separation for sweep type = DFRequency (delta frequency).

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Stopping tone separation between F1 and F2 in Hz. Both F1 and F2 tones MUST be within the frequency range of the PNA where:

$$F1 \text{ (stop)} = \text{FREQ:FCEN} - \text{DFR:Stop} / 2$$

$$F2 \text{ (stop)} = \text{FREQ:FCEN} + \text{DFR:Stop} / 2$$

**Examples**

```
SENS:IMD:FREQ:DFR:STOP 1e6  
sense2:imd:frequency:dfrequency:stop 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:DFRequency:STOP?

**Return Type** Numeric

**Default** 10 MHz

---

**SENSe<num>:IMD:FREQuency:F1[:CW] <num>**

**(Read-Write)** Sets and returns the frequency of the F1 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> F1 tone frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:F1 1e9  
sense2:imd:frequency:F1:cw 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:F1[:CW]?

**Return Type** Numeric

**Default** .9995 GHz

---

**SENSe<num>:IMD:FREQuency:F2[:CW] <num>**

**(Read-Write)** Sets and returns the frequency of the F2 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> F2 tone frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:F2 1e9  
sense2:imd:frequency:F2:cw 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:F2[:CW]?

**Return Type** Numeric

**Default** 1.0005 GHz

---

**SENSe<num>:IMD:FREQuency:FCENter[:CW] <num>**

**(Read-Write)** Sets and returns the center frequency of the main tones.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Tone center frequency in Hz. The F1 and F2 tones **MUST** be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:FCEN 1e9  
sense2:imd:frequency:fcenter:cw 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:FCENter[:CW]?

**Return Type** Numeric

**Default** 1.0 GHz

---

**SENSe<num>:IMD:FREQuency:FCENter:CENTer <num>**

**(Read-Write)** Sets and returns the sweep center frequency when sweeping the main tones.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Center frequency in Hz. The F1 and F2 tones **MUST** be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:FCEN:CENT 1e9  
sense2:imd:frequency:fcenter:center 2e7
```

**Query Syntax** SENSe<num>:IMD:FREQuency:FCENter:CENTer?

**Return Type** Numeric

**Default** 13.255 GHz

---

**SENSe<num>:IMD:FREQuency:FCENter:SPAN <num>**

**(Read-Write)** Sets and returns the frequency span when sweeping the main tones.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Frequency span in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:FCEN:SPAN 1e9  
sense2:imd:frequency:fcenter:span 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:FCENter:SPAN?

**Return Type** Numeric

**Default** 26.489 GHz

---

**SENSe<num>:IMD:FREQuency:FCENter:STARt <num>**

**(Read-Write)** Sets and returns the start frequency when sweeping the main tones.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Start frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:FCEN:STAR 1e9  
sense2:imd:frequency:fcenter:start 2e7
```

**Query Syntax** SENSE<num>:IMD:FREQuency:FCENter:STARt?

**Return Type** Numeric

**Default** 10.5 MHz

---

**SENSe<num>:IMD:FREQuency:FCENter:STOP <num>**



**(Read-Write)** Sets and returns the stop frequency when sweeping the main tones.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Stop frequency in Hz. The F1 and F2 tones MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMD:FREQ:FCEN:STOP 1e9  
sense2:imd:frequency:fcenter:stop 2e9
```

**Query Syntax** SENSE<num>:IMD:FREQuency:FCENter:STOP?

**Return Type** Numeric

**Default** 26.4995 GHz

---

## SENSe:IMD:HOPRduct?

**(Read-only)** Returns the highest order product that can be measured by SweptIMD.

**Parameters** None

**Examples**

```
SENS:IMD:HOPR?  
'always returns 9
```

**Return Type** Numeric

**Default** 9

---

## SENSe<num>:IMD:IFBWidth:MAIN <num>

**(Read-Write)** Sets and returns the IF Bandwidth for measurement of the main F1 and F2 tones. [Learn more about setting IFBW for IMD.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <num> Choose from: 1 | 2 | 3 | 5 | 7 | 10 | 15 | 20 | 30 | 50 | 70 | 100 | 150 | 200 | 300 | 500 | 700 | 1k | 1.5k | 2k | 3k | 5k | 7k | 10k | 15k | 20k | 30k | 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k
- If an invalid number is specified, the analyzer will round up to the closest valid number.

**Examples**

```
SENS:IMD:IFBW:MAIN 280e3  
sense2:imd:ifbwidth:main 150K
```

**Query Syntax** SENSE<cnum>:IMD:IFBWidth:MAIN?

**Return Type** Numeric

**Default** 1 kHz

---

**SENSe<cnum>:IMD:IFBWidth:IMTone <num>**

(Read-Write) Sets and returns the IF Bandwidth for measurement of the intermodulation products.  
[Learn more about setting IFBW for IMD.](#)

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.

<num> Choose from: 1 | 2 | 3 | 5 | 7 | 10 | 15 | 20 | 30 | 50 | 70 | 100 | 150 | 200 | 300 | 500 | 700 | 1k | 1.5k | 2k | 3k | 5k | 7k | 10k | 15k | 20k | 30k | 50k | 70k | 100k | 150k | 200k | 280k | 360k | 600k

If an invalid number is specified, the analyzer will round up to the closest valid number.

**Examples**

```
SENS:IMD:IFBW:IMT 50  
sense2:imd:ifbwidth:imtone 200
```

**Query Syntax** SENSE<cnum>:IMD:IFBWidth:IMTone?

**Return Type** Numeric

**Default** 1 kHz

---

**SENSe<cnum>:IMD:NORMalized:MODE <char>**

**(Read-Write)** Sets and returns the method by which CTB and CSO calculations are performed.

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<char> Normalization mode. Choose from:

**NONE** - the normalized power is not used in calculation

**NCARrier** - CTB and CSO is corrected by subtracting  $10 \cdot \log(N/2)$ , where

- N = # of carriers for CTB
- N = # of distortion products for CSO

**DBM** - the composited normalized power for CTB or CSO is treated as a dBm value

**DBmV** - the composited normalized power for CTB or CSO is treated as a dBmV value.

**Note:** Power values are stored using the currently-set units. Therefore, first set units with this command, then set power values using:

[SENS:IMD:CSO:NORM:POW](#) or [SENS:IMD:CTB:NORM:POW](#)

#### Examples

```
SENS:IMD:NORM:MODE NCAR
```

```
sense2:imd:normalized:mode none
```

**Query Syntax** SENSE<cnum>:IMD:NORMalized:MODE?

**Return Type** Character

**Default** NCARrier

---

**SENSe<cnum>:IMD:PMAP <input>,<output>**

**(Write-only)** Sets the input port and output port of an IMD or IMDx channel. This setting is necessary only when using the **limited port mapping feature**. [Learn more.](#)

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <input> PNA port connected to the DUT Input. Choose from 1 or 3. When input is 3, an external combiner must be used.
- <output> PNA port connected to the DUT Output.

When input is 1, output must be 2.

When input is 3, output must be 4.

**Examples**

```
SENS:IMD:PMAP 3,4
```

```
sense2:imd:pmap 3,4
```

**Query Syntax** Not Applicable

**Default** 1,2

---

**SENSe<num>:IMD:PMAP:INPut?**

**(Read-only)** Returns the PNA test port to be connected to the DUT input for an IMD or IMDx channel. Set the PNA port to DUT mapping using [SENS:IMD:PMAP](#)

**Parameters**

- <num> IMD channel number. If unspecified, value is set to 1.

**Examples**

```
SENS:IMD:PMAP:INP?
```

```
sense2:imd:pmap:input?
```

**Default** 1

---

**SENSe<num>:IMD:PMAP:OUTPut?**

**(Read-only)** Returns the PNA test port to be connected to the DUT output for an IMD or IMDx channel. Set the PNA port to DUT mapping using [SENS:IMD:PMAP](#)

**Parameters**

<cnun> IMD channel number. If unspecified, value is set to 1.

**Examples**

```
SENS:IMD:PMAP:OUTP?
```

```
sense2:imd:pmap:output?
```

**Default** 2

---

**SENSe<cnun>:IMD:TPOWer:COUPle[:STATe] <bool>**

**(Read-Write)** Sets and returns the state of power coupling for F1 and F2.

**Parameters**

<cnun> Any existing channel number; if unspecified, value is set to 1.

<bool> Tone power level coupling state. Choose from:

**ON (or 1)** - F1 and F2 power is coupled.

**OFF (or 0)** - F1 and F2 power is NOT coupled. Set power levels individually.

**Examples**

```
SENS:IMD:TPOW:COUP 0
```

```
sense2:imd:tpower:couple:state ON
```

**Query Syntax** SENSe<cnun>:IMD:TPOWer:COUPle[:STATe]?

**Return Type** Boolean

**Default** ON

---

**SENSe<cnun>:IMD:TPOWer:F1 <num>**

**(Read-Write)** Sets and returns the power level of the F1 tone. When [SENS:IMD:TPOW:COUP ON](#) (tone power is coupled), setting either F1 or F2 power sets both.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Tone power level in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F1 0  
sense2:imd:tpower:F1 -10
```

**Query Syntax** SENSE<cnum>:IMD:TPOWer:F1?

**Return Type** Numeric

**Default** -20 dBm

---

**SENSe<cnum>:IMD:TPOWer:F2 <num>**

**(Read-Write)** Sets and returns the power level of the F2 tone. When [SENS:IMD:TPOW:COUP ON](#) (tone power is coupled), setting either F1 or F2 power sets both.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Tone power level in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F2 0  
sense2:imd:tpower:F2 -10
```

**Query Syntax** SENSe<cnum>:IMD:TPOWer:F2?

**Return Type** Numeric

**Default** -20 dBm

---

**SENSe<cnum>:IMD:TPOWer:F1:STARt <num>**

**(Read-Write)** Sets and returns the start power level of the F1 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Start power in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F1:STAR 0  
sense2:imd:tpower:F1:start -10
```

**Query Syntax** SENSE<num>:IMD:TPOWer:F1:START?

**Return Type** Numeric

**Default** -20 dBm

---

**SENSe<num>:IMD:TPOWer:F1:STOP <num>**

**(Read-Write)** Sets and returns the stop power level of the F1 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Stop power in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F1:STOP 0  
sense2:imd:tpower:F1:stop 10
```

**Query Syntax** SENSE<num>:IMD:TPOWer:F1:STOP?

**Return Type** Numeric

**Default** -10 dBm

---

**SENSe<num>:IMD:TPOWer:F2:STARt <num>**

**(Read-Write)** Sets and returns the start power level of the F2 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Start power in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F2:STAR 0  
sense2:imd:tpower:F2:start -10
```

**Query Syntax** SENSE<num>:IMD:TPOWer:F2:START?

**Return Type** Numeric

**Default** -20 dBm

---

**SENSe<num>:IMD:TPOWer:F2:STOP <num>**

**(Read-Write)** Sets and returns the stop power level of the F2 tone.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.  
<num> Stop power in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMD:TPOW:F2:STOP 0  
sense2:imd:tpower:F2:stop 10
```

**Query Syntax** SENSE<num>:IMD:TPOWer:F2:STOP?

**Return Type** Numeric

**Default** -10 dBm

---

Last Modified:

11-Aug-2009	Added PMAP
13-Mar-2009	Edited for IMDx
15-Aug-2008	MX New topic



## Sense:IMS Commands

---

Controls IM Spectrum measurement configuration.

### SENSe:IMS

#### PMAP

|INPut

|OUTPut

#### RBW

#### RESPonse

|STARt

|STOP

|CENTer

|SPAN

#### STIMulus

|DFRequency

|FCENter

|F1FRequency

|F2FRequency

|TPOWer

|F1

|F2

#### SWEEp:

|TYPE

|ORDer

#### TPOWer

|COUPle[:STATe]

|F1

|F2

## TRACking

| [CHANnel](#)

| [MSEnable](#)

| [SINdex](#)

| [STATe](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [About IM Spectrum measurements](#)
- [IMD Application](#)
- [Learn about Measurement Class](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## SENSe<cnum>:IMS:PMAP <input>,<output>

**(Write-only)** Sets the input port and output port of an IMS or IMSx channel. This setting is necessary only when using the **limited port mapping feature**. [Learn more.](#)

### Parameters

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <input> PNA port connected to the DUT Input. Choose from 1 or 3. When input is 3, an external combiner must be used.
- <output> PNA port connected to the DUT Output.
- When input is 1, output must be 2.
- When input is 3, output must be 4.

### Examples

```
SENS:IMS:PMAP 3,4  
  
sense2:ims:pmap 3,4
```

**Query Syntax** Not Applicable

**Default** 1,2

### SENSe<cnum>:IMS:PMAP:INPut?

**(Read-only)** Returns the PNA test port to be connected to the DUT input for an IMS or IMSx channel. Set the PNA port to DUT mapping using [SENS:IMS:PMAP](#)

#### Parameters

<cnum> IMS channel number. If unspecified, value is set to 1.

#### Examples

```
SENS:IMS:PMAP:INP?
```

```
sense2:ims:pmap:input?
```

**Default** 1

---

### SENSe<cnum>:IMS:PMAP:OUTPut?

**(Read-only)** Returns the PNA test port to be connected to the DUT output for an IMS or IMSx channel. Set the PNA port to DUT mapping using [SENS:IMS:PMAP](#)

#### Parameters

<cnum> IMS channel number. If unspecified, value is set to 1.

#### Examples

```
SENS:IMS:PMAP:OUTP?
```

```
sense2:ims:pmap:output?
```

**Default** 2

---

### SENSe<cnum>:IMS:RBW <num>

**(Read-Write)** Sets and returns the Resolution Bandwidth for the IM Spectrum measurement.

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<num> Resolution BW in Hz. Choose from: 60k | 100k | 150k | 300k, 600k | 1.0M | 3.0M

If an invalid number is specified, the PNA will round up to the closest valid number.

#### Examples

```
SENS:IMS:RBW 600e3
```

```
sense2:ims:rbw 1MHz
```

**Query Syntax** SENSe<cnum>:IMS:RBW?

**Return Type** Numeric

**Default** 600 kHz

---

### **SENSe<cnum>:IMS:RESPonse:STARTt <num>**

**(Read-Write)** Sets and returns the receiver Start frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

#### **Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.  
<num> Start frequency in Hz. Choose a frequency within the range of the PNA.

#### **Examples**

```
SENS:IMS:RESP:STAR 1e9  
sense2:ims:response:start 100e6
```

**Query Syntax** SENSe<cnum>:IMS:RESPonse:STARTt?

**Return Type** Numeric

**Default** 950 MHz

---

### **SENSe<cnum>:IMS:RESPonse:STOP <num>**

**(Read-Write)** Sets and returns the receiver Stop frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

#### **Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.  
<num> Stop frequency in Hz. Choose a frequency within the range of the PNA.

#### **Examples**

```
SENS:IMS:RESP:STOP 26e9  
sense2:ims:response:stop 100e6
```

**Query Syntax** SENSe<cnum>:IMS:RESPonse:STOP?

**Return Type** Numeric

**Default** 1.05 MHz

---

### **SENSe<cnum>:IMS:RESPonse:CENTer <num>**

**(Read-Write)** Sets and returns the receiver Center frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Center frequency in Hz. Choose a frequency within the range of the PNA.

**Examples**

```
SENS:IMS:RESP:CENT 26e9  
sense2:ims:response:center 100e6
```

**Query Syntax** SENSE<cnum>:IMS:RESPonse:CENTer?

**Return Type** Numeric

**Default** 1.0 GHz

---

**SENSe<cnum>:IMS:RESPonse:SPAN <num>**

**(Read-Write)** Sets and returns the Span of receiver frequencies for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled and when Sweep Type = Linear. Otherwise, this setting is ignored.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Frequency span in Hz. All receiver frequencies should be within the range of the PNA.

**Examples**

```
SENS:IMS:RESP:SPAN 10e9  
sense2:ims:response:span 100e6
```

**Query Syntax** SENSe<cnum>:IMS:RESPonse:SPAN?

**Return Type** Numeric

**Default** 100 MHz

---

**SENSe<cnum>:IMS:STIMulus:DFrequency <num>**

**(Read-Write)** Sets and returns the DeltaF (tone spacing) for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled. Otherwise, this setting is ignored.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Delta frequency in Hz. All stimulus settings MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMS:STIM:DFR 1e6  
sense2:ims:stimulus:dfrequency 100e6
```

**Query Syntax** SENSE<cnum>:IMS:STIMulus:DFrequency?

**Return Type** Numeric

**Default** 10 MHz

---

**SENSe<cnum>:IMS:STIMulus:FCENter <num>**

**(Read-Write)** Sets and returns the Center frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled. Otherwise, this setting is ignored.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Center frequency in Hz. All stimulus settings MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMS:STIM:FCEN 1e6  
sense2:ims:stimulus:fcenter 100e6
```

**Query Syntax** SENSe<cnum>:IMS:STIMulus:FCENter?

**Return Type** Numeric

**Default** 1.0 GHz

---

**SENSe<cnum>:IMS:STIMulus:F1FRequency <num>**

**(Read-Write)** Sets and returns the F1 frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled. Otherwise, this setting is ignored.

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.

<num> F1 frequency in Hz. All stimulus settings MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMS:STIM:F1FR 1e6
```

```
sense2:ims:stimulus:f1frequency 100e6
```

**Query Syntax** SENSE<cnum>:IMS:STIMulus:F1FRequency?

**Return Type** Numeric

**Default** 995 MHz

---

**SENSe<cnum>:IMS:STIMulus:F2FRequency <num>**

**(Read-Write)** Sets and returns the F2 frequency for the IM Spectrum measurement. Valid ONLY when Tracking is NOT enabled. Otherwise, this setting is ignored.

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.

<num> F2 frequency in Hz. All stimulus settings MUST be within the frequency range of the PNA.

**Examples**

```
SENS:IMS:STIM:F2FR 1e6
```

```
sense2:ims:stimulus:f2frequency 100e6
```

**Query Syntax** SENSE<cnum>:IMS:STIMulus:F2FRequency?

**Return Type** Numeric

**Default** 1.005 MHz

---

**SENSe<cnum>:IMS:SWEep:TYPE <char>**

**(Read-Write)** Sets and returns the method in which the spectrum of signals to view are specified. When Tracking is enabled the frequency of the main tones (f1 and f2) are always determined by the Swept IMD channel.

#### Parameters

<num> Any existing channel number; if unspecified, value is set to 1.

<char> IM Spectrum sweep type. Choose from:

- **LINEar** When Tracking is enabled, allows tuning the Response Settings (receiver) to any values within the frequency range of the PNA. When Tracking is NOT enabled also allows setting the Stimulus (sources) to any values within the frequency range or the PNA.
- **SECond** The receiver is tuned to view the 2nd order products (f2-f1 and f1+f2) of the main tones that are currently specified in Stimulus Settings. When Tracking is enabled, the main tones are specified in the Swept IMD channel.
- **THIRd** The receiver is tuned to view the 3rd order products (2f1 -f2 and 2f2-f1) of the main tones that are currently specified in Stimulus Settings. When Tracking is enabled, the main tones are specified in the Swept IMD channel.
- **NTH** The frequency range is set to  $N * \Delta F$ . This algorithm will NOT tune the receivers to view the EVEN order products.

#### Examples

```
SENS:IMS:SWEep:TYPE LIN  
sense2:ims:sweep:type nth
```

**Query Syntax** SENSE<num>:IMS:SWEep:TYPE?

**Return Type** Character

**Default** NTH

---

**SENSe<num>:IMS:SWEep:ORDer <num>**



**(Read-Write)** Sets and returns the order number of signals to view when [SENS:IMS:SWE:TYPE NTH](#) is specified.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <num> Order number of IM products to view. The frequency range is set to N (this number) x DeltaF (set with [SENS:IMS:STIM:DFR](#)).

**Examples**

```
SENS:IMS:SWEep:ORD 5  
sense2:ims:sweep:order 12
```

**Query Syntax** SENSE<cnum>:IMS:SWEep:ORDER?

**Return Type** Numeric

**Default** 9

---

**SENSe<cnum>:IMS:TPOWer:COUPle[:STATe] <bool>**

**(Read-Write)** Sets and returns the state of power coupling for F1 and F2.

**Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <bool> Tone power level coupling. Choose from:
- ON (or 1)** - F1 and F2 power is coupled.
- OFF (or 0)** - F1 and F2 power is NOT coupled. Set power levels individually.

**Examples**

```
SENS:IMS:TPOW:COUP 0  
sense2:ims:tpower:couple:state ON
```

**Query Syntax** SENSe<cnum>:IMS:TPOWer:COUPle[:STATe]?

**Return Type** Boolean

**Default** ON

---

**SENSe<cnum>:IMS:STIMulus:TPOWer:F1 <value>**

**(Read-Write)** Sets and returns the power level of the F1 tone. When [SENS:IMS:TPOW:COUP ON](#) (tone power is coupled), setting either F1 or F2 power sets both. This setting is ignored if [SENS:IMS:TRAC:STAT](#) is enabled.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <value> F1 tone power level in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMS:STIM:TPOW:F1 -10  
sense2:ims:stimulus:tpower:f1 0
```

**Query Syntax** SENSE<num>:IMS:STIMulus:TPOWer:F1?

**Return Type** Numeric

**Default** -20

---

**SENSe<num>:IMS:STIMulus:TPOWER:F2 <value>**

**(Read-Write)** Sets and returns the power level of the F2 tone. When [SENS:IMS:TPOW:COUP ON](#) (tone power is coupled), setting either F1 or F2 power sets both. This setting is ignored if [SENS:IMS:TRAC:STAT](#) is enabled.

**Parameters**

- <num> Any existing channel number; if unspecified, value is set to 1.
- <value> F2 tone power level in dBm. Choose a value between +30 dBm and -30 dBm.

**Examples**

```
SENS:IMS:STIM:TPOW:F2 -10  
sense2:ims:stimulus:tpower:f2 0
```

**Query Syntax** SENSe<num>:IMS:STIMulus:TPOWer:F2?

**Return Type** Numeric

**Default** -20

---

**SENSe<num>:IMS:TRACking:CHANnel <num>**

**(Read-Write)** Sets and returns the IMD channel number to which the IM Spectrum channel is coupled.

**Parameters**

- <num> IMS channel. If unspecified, value is set to 1.  
<num> Existing IMD channel to which frequency and power settings are coupled.

**Examples**

```
SENS:IMS:TRAC:CHAN 2  
sense2:ims:tracking:channel 1
```

**Query Syntax** SENSE<num>:IMS:TRACking:CHANnel?

**Return Type** Numeric

**Default** First IMD channel

---

**SENSe<num>:IMS:TRACking:MSEnable <bool>**

**(Read-Write)** Sets and returns the step sweep mode for the IM Spectrum channel.

**Parameters**

- <num> IMS channel. If unspecified, value is set to 1.  
<bool> Choose from:

**OFF (or 0)** - Automatic Step

**ON (or 1)** - Manual Step

**Examples**

```
SENS:IMS:TRAC:MSEN 1  
sense2:ims:tracking:mseenable 0
```

**Query Syntax** SENSE<num>:IMS:TRACking:MSEnable?

**Return Type** Boolean

**Default** 0 - Automatic

---

**SENSe<num>:IMS:TRACking:SINDeX <num>**

**(Read-Write)** When [SENS:IMS:TRAC:MSEN = Manual](#), sets and returns the data point number at which a sweep is performed.

**Parameters**

- <num> IMS channel. If unspecified, value is set to 1.
- <num> Step index. Choose from 1 to the specified number of points.

**Examples**

```
SENS:IMS:TRAC:SINDeX 201  
sense2:ims:tracking:sindex 1
```

**Query Syntax** SENSE<num>:IMS:TRACking:SINDeX?

**Return Type** Numeric

**Default** 1

---

**SENSe<num>:IMS:TRACking:STATe <bool>**

**(Read-Write)** When an IMD channel exists, allows the IM Spectrum frequency and power setting to track (couple with) the IMD channel settings.

**Parameters**

- <num> IMS channel. If unspecified, value is set to 1.
- <bool> Tracking state. Choose from:
- ON (or 1)** - IM Spectrum frequency and power settings track the IMD channel settings.
- OFF (or 0)** - IM Spectrum frequency and power settings are specified in the IMS channel.

**Examples**

```
SENS:IMS:TRAC:STAT 0  
sense2:ims:tracking:state ON
```

**Query Syntax** SENSe<num>:IMS:TRACking:STATe?

**Return Type** Boolean

**Default** OFF

---

Last Modified:

11-Aug-2009 Added PMAP (9.0)  
3-Jun-2010 MX New topic



## Sense:Mixer Commands

---

Performs Mixer setup and configuration.

**SENSe:MIXer:**

**APPLy**

**AVOIdspurs**

**CALC**

**DISCard**

**ELO** - [More Commands](#)

**IF:FREQ:**

| **SIDeband**

| **STARt**

| **STOP**

**INPut:FREQ:**

| **DENominator**

| **FIXed**

| **MODE**

| **NUMerator**

| **STARt**

| **STOP**

**INPut:POWer**

| **USENominal**

**LO:FREQ:**

| **DENominator**

| **FIXed**

| **ILTI**

| **MODE**

| **NUMerator**

| **STARt**

| **STOP**

**LO:NAME**

**LO:POWer**

| **STARt**

| **STOP**

**LOAD**

**NORMalize**

**OUTPut:FREQ:**

| **FIXed**

| **MODE**

| **SIDeband**

| **STARt**

| **STOP**

[PHASe](#)  
[PMAp](#)  
     | [INPut](#)  
     | [OUTPut](#)  
[ROLE](#)  
     | [CATalog?](#)  
     | [DEvice](#)  
[REVerse](#)  
[SAVE](#)  
[STAGe](#) (number of LOs)  
[XAXis](#)

Click on a [blue](#) keyword to view the command details.

#### See Also

- [Example Programs](#)
- [Learn about the Frequency Converter Application](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Note:** If you are changing several mixer configuration settings, you can make all the changes first and then issue the [Calculate](#) and [Apply](#) commands as you would do from the user interface.

## SENSe<ch>:MIXer:APPLy

(Write only) Applies the mixer setup settings and turns the channel ON. (Performs the same function as the Apply button on the [mixer setup dialog box](#)).

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples**     **SENS:MIX:APPL**

**Query Syntax**     Not Applicable

**Default**     Not Applicable

## SENSe<ch>:MIXer:AVOIdspurs <bool>

**(Read Write)** Sets and returns the state of the avoid spurs feature. [Learn more about avoid spurs.](#)

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> Avoid spurs state. Choose from
- 0 - Avoid spurs OFF
  - 1 - Avoid spurs ON

**Examples**

```
SENS:MIX:AVO  
sense2:mixer:avoidspurs 1
```

**Query Syntax** SENSE<ch>:MIXer:AVOidspurs?

**Return Type** Boolean

**Default** 0 (OFF)

---

**SENSe<ch>:MIXer:CALCulate <char>**

**(Write only)** Calculates the Input, IF, or Output frequencies of the mixer setup and updates the channel settings.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <char> Mixer port to be calculated. Choose from:

<char>	1st or only stage requires:	In addition, 2nd stage requires:
INPut	<ul style="list-style-type: none"><li>• Output Start and Stop frequencies</li><li>• LO frequency</li><li>• Output sideband (High or Low)</li></ul>	<ul style="list-style-type: none"><li>• IF Start and Stop frequencies</li><li>• 2nd LO frequency</li><li>• IF sideband (High or Low)</li></ul>
BOTH	NA	<ul style="list-style-type: none"><li>• IF Start and Stop frequencies</li><li>• Both LO frequencies</li></ul>
OUTPut	<ul style="list-style-type: none"><li>• Input Start and Stop frequencies</li></ul>	<ul style="list-style-type: none"><li>• IF Start and Stop frequencies</li></ul>



	<ul style="list-style-type: none"> <li>• LO frequency</li> <li>• Output sideband (High or Low)</li> </ul>	<ul style="list-style-type: none"> <li>• 2nd LO frequency</li> <li>• IF sideband (High or Low)</li> </ul>
<b>LO_1</b>	<ul style="list-style-type: none"> <li>• Input Start and Stop frequencies</li> <li>• Output frequency</li> <li>• Output sideband (High or Low)</li> </ul>	<ul style="list-style-type: none"> <li>• IF Start and Stop frequencies</li> <li>• 2nd LO frequency</li> <li>• IF sideband (High or Low)</li> </ul>
<b>LO_2</b>	NA	<ul style="list-style-type: none"> <li>• Input Start and stop frequencies</li> <li>• 1st LO start and stop frequencies</li> <li>• Output frequency</li> <li>• IF sideband(High or Low)</li> <li>• Output sideband(High or Low)</li> </ul>

**Examples** `SENS:MIX:CALC` Output

**Query Syntax** Not Applicable

**Default** Not Applicable

## **SENSe<ch>:MIXer:DISCard**

(Write only) Cancels changes that have been made to the Converter setup and reverts to the previously-saved setup. Same as the Cancel button on the [mixer setup dialog box](#).

### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples** `SENS:MIX:DISC`



**Query Syntax** Not Applicable

**Default** Not Applicable

## **SENSe<ch>:MIXer:IF:FREQuency:SIDeband <char>**

**(Read-Write)** When [two LO stages are used](#), sets or returns whether to select the sum or difference for the IF1 product. (Input + or - LO1 = IF1)



- This setting corresponds to the   buttons on LO1 on the [Mixer setup dialog](#)
- This setting is ignored when [ONE LO stage](#) is selected.
- Also set [SENS:MIX:OUTP:FREQ:SID](#) to LOW or HIGH to determine the output frequency of the mixer.

#### [See Note](#)

##### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Sideband value. Choose from

**LOW** - Difference (-)

**HIGH** - Sum (+)

##### Examples

```
SENS:MIX:IF:FREQ:SID LOW
SENSe2:MIXer:IF:FREQ:SIDeband HIGH
```

**Query Syntax** SENSe<ch>:MIXer:IF:FREQuency:SIDeband?

**Return Type** Character

**Default** LOW

---

#### **SENSe<ch>:MIXer:IF:FREQuency:STARt <num>**

**(Read-Write)** Sets or returns the IF start frequency value of the mixer. [See Note](#)

##### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<num> IF Start Frequency value

##### Examples

```
SENS:MIX:IF:FREQ:STAR 1e9
SENSe2:MIXer:IF:FREQ:STARt 1000000000
```

**Query Syntax** SENSe<ch>:MIXer:IF:FREQuency:STARt?

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe<ch>:MIXer:IF:FREQuency:STOP <num>

(Read-Write) Sets or returns the stop frequency value of the mixer IF frequency. [See Note](#)

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<num> IF Stop Frequency value

### Examples

```
SENS:MIX:IF:FREQ:STOP 2e9
SENSe2:MIXer:IF:FREQ:STOP 2000000000
```

**Query Syntax** SENSe<ch>:MIXer:IF:FREQuency:STOP?

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe<ch>:MIXer:INPut:FREQuency:DENominator <value>

(Read-Write) Sets or returns the denominator value of the Input Fractional Multiplier. [See Note](#)

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input denominator value.

### Examples

```
SENS:MIX:INP:FREQ:DEN 5
SENSe2:MIXer:INPut:FREQ:DENominator 4
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQuency:DENominator?

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe<ch>:MIXer:INPut:FREQuency:FIXed<value>

**(Read-Write)** Sets or returns the fixed frequency of the input. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input frequency.

**Examples**

```
SENSe:MIXer:INPut:FREQ:FIXed 1e9  
SENSe2:MIXer:INPut:FREQ:FIXed 1000000000
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQuency:FIXed?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:INPut:FREQuency:MODE <char>**

**(Read-Write)** Sets or returns the Input sweep mode.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Input sweep mode. Choose either **FIXED** or **SWEPT**

**Examples**

```
SENS:MIX:INP:FREQ:MODE FIXED  
SENSe2:MIXer:INP:FREQ:MODE swept
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQuency:MODE?

**Return Type** Character

**Default** Fixed

---

**SENSe<ch>:MIXer:INPut:FREQuency:NUMerator <value>**

**(Read-Write)** Sets or returns the numerator value of the Input Fractional Multiplier. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input numerator value.

**Examples**

```
SENS:MIX:INP:FREQ:NUM 3  
SENSe2:MIXer:INPut:FREQ:NUMerator 1
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQ:NUMerator?

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:MIXer:INPut:FREQuency:STARt <value>

(Read-Write) Sets or returns the Input start frequency value of the mixer. [See Note](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input Start frequency

#### Examples

```
SENS:MIX:INP:FREQ:STAR 1e9  
SENSe2:MIXer:INPut:FREQ:STARt 1000000000
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQuency:STARt?

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:MIXer:INPut:FREQuency:STOP <value>

(Read-Write) Sets or returns the Input stop frequency value of the mixer. [See Note](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input stop frequency

#### Examples

```
SENS:MIX:INP:FREQ:STOP 2e9  
SENSe2:MIXer:INPut:FREQ:STOP 2000000000
```

**Query Syntax** SENSe<ch>:MIXer:INPut:FREQuency:STOP?

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:MIXer:INPut:POWer <value>

**(Read-Write)** Sets or returns the value of the Input Power.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Input power in dBm.

**Examples**

```
SENS:MIX:INP:POW 9
SENSe2:MIXer:INPut:POWer 5
```

**Query Syntax** SENSe<ch>:MIXer:INPut:POWer?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:INPut:POWer:USENominal <bool>**

**(Read-Write)** Toggles the Nominal Incident Power setting ON and OFF. This setting is ONLY to be used with SMC measurements, not VMC. [Learn more about Nominal Incident Power.](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> **(boolean)** - Nominal Incident Power State. Choose from:

ON (1) - Turn nominal incident power ON

OFF (0) - Turn nominal incident power OFF

**Examples**

```
SENS:MIX:INP:POW:USEN 1
SENSe2:MIXer:INPut:POWer:USENominal OFF
```

**Query Syntax** SENSe<ch>:MIXer:INPut:POWer:USENominal?

**Return Type** Boolean

**Default** OFF

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:DENominator <value>**

**(Read-Write)** Sets or returns the denominator value of the LO Fractional Multiplier. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose 1 or 2.

<value> LO denominator.

**Examples**

```
SENS:MIX:LO:FREQ:DEN 5
SENSe2:MIXer:LO2:FREQ:DENominator 4
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:DENominator?

**Return Type** Numeric

**Default** 1

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:FIXed <value>**

**(Read-Write)** Sets or returns the fixed frequency of the specified mixer LO. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose 1 or 2

<value> LO frequency.

**Examples**

```
SENS:MIX:LO:FREQ:FIX 1e9
SENSe2:MIXer:LO2:FREQ:FIXed 1000000000
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:FIXed?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:ILTI <bool>**

**(Read-Write)** Specifies whether to use the Input frequency that is **greater than** the LO or **less than** the LO. To learn more, see the [mixer setup](#) dialog box help.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1.
- <n> LO stage number. Choose **1** or **2**
- <bool> **ON (1)** - Use the Input that is Greater than the specified LO.
- OFF (0)** - Use the Input that is Less than the specified LO.

**Examples**

```
SENS:MIX:LO1:FREQ:ILTI 1
sense2:mixer:lo2:frequency:ilti ON
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:ILTI?

**Return Type** Boolean

**Default** OFF

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:MODE <char>**

**(Read-Write)** Sets or returns the LO sweep mode.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1.
- <n> LO stage number. Choose **1** or **2**
- <char> LO sweep mode. Choose either **FIXED** or **SWEPT**

**Examples**

```
SENS:MIX:LO:FREQ:MODE FIXED
SENSe2:MIXer:LO2:FREQ:MODE swept
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:MODE?

**Return Type** Character

**Default** Fixed

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:NUMerator <value>**



**(Read-Write)** Sets or returns the numerator value of the LO Fractional Multiplier. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose **1** or **2**

<value> LO Numerator.

**Examples**

```
SENS:MIX:LO:FREQ:NUM 5
SENSe2:MIXer:LO2:FREQ:NUMerator 4
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:NUMerator?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:STARt <value>**

**(Read-Write)** Sets or returns the LO start frequency value. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose **1** or **2**

<value> LO Start Frequency in Hertz.

**Examples**

```
SENS:MIX:LO:FREQ:STAR 5E9
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:STARt?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:LO<n>:FREQuency:STOP <value>**

**(Read-Write)** Sets or returns the LO stop frequency value. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose **1** or **2**

<value> LO Stop Frequency in Hertz.

**Examples**

```
SENS:MIX:LO:FREQ:STOP 5E9
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:FREQuency:STOP?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:LO<n>:NAME <value>**

**(Read-Write)** Sets or returns the name of the PNA internal source or external source to use as the LO in an FCA measurement.

See [Remotely Specifying a Source Port](#).

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage number. Choose 1 or 2.

<value> **(string)** - LO Source name. Use [Source:CAT?](#) to return a list of valid source ports. An external source must be configured and selected to be valid. [Learn more about external source configuration](#).

**Examples**

```
SENS:MIX:LO:NAME "MySource"
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:NAME?

**Return Type** String

**Default** Not Controlled

---

**SENSe<ch>:MIXer:LO<n>:POWer <value>**

**(Read-Write)** Sets or returns the value of the LO Power.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1.  
<n> LO stage. Choose 1 or 2  
<value> LO Power in dBm

**Examples**

```
SENS:MIX:LO:POW 9
```

**Query Syntax** SENSE<ch>:MIXer:LO<n>:POWer?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:LO<n>:POWer:START <value>**

**(Read-Write)** For an IMDx LO power sweep, sets or returns the LO power start value.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1.  
<n> LO stage. Choose 1  
<value> LO start power in dBm

**Examples**

```
SENS:MIX:LO1:POW:STAR -10
```

**Query Syntax** SENSE<ch>:MIXer:LO1:POWer:START?

**Return Type** Numeric

**Default** - 20 dBm

---

**SENSe<ch>:MIXer:LO<n>:POWer:STOP <value>**

**(Read-Write)** For an IMDx LO power sweep, sets or returns the LO power stop value.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> LO stage. Choose 1

<value> LO stop power in dBm

**Examples**

```
SENS:MIX:LO1:POW:STOP 10
```

**Query Syntax** SENSE<ch>:MIXer:LO1:POWer:STOP?

**Return Type** Numeric

**Default** -10 dBm

---

**SENSe<ch>:MIXer:LOAD <name>**

**(Write-only)** Loads a previously-configured mixer attributes file (.mxr)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<name> Path and file name (including .mxr extension) to load.

**Examples**

```
SENSe:MIXer:LOAD "C:/Program Files/Agilent/Network  
Analyzer/Documents/Mixer/MyMixer.mxr"
```

**Default** Not Applicable

---

**SENSe<ch>:MIXer:NORMalize <value>**

**(Read-Write)** Sets or returns the data point for normalizing the phase measurement. [Learn more.](#)

**Parameters**

<ch> Channel number of the SMC measurement. If unspecified, value is set to 1.

<value> Normalization data point. Choose a data point number between 1 and the max number of data points in the sweep that has the least amount of expected noise.

**Examples**

```
SENS:MIX:NORM 101  
  
sense2:mixer:normalize 50
```

**Query Syntax** SENSE<ch>:MIXer:NORMalize?

**Return Type** Numeric

**Default** Middle point in the sweep

---

---

### SENSe<ch>:MIXer:OUTPut:FREQuency:FIXed <value>

(Read-Write) Sets or returns the output fixed frequency of the mixer. [See Note](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Output fixed frequency in Hertz.

**Examples** `SENS:MIX:OUTP:FREQ:FIX 5e9`

**Query Syntax** `SENSe<ch>:MIXer:OUTPut:FREQuency:FIXed?`

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:MIXer:OUTPut:FREQuency:MODE <char>

(Read-Write) Sets or returns the Output sweep mode.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Output sweep mode. Choose either **FIXED** or **SWEPT**

**Examples** `SENS:MIX:OUT:FREQ:MODE FIXED`  
`SENSe2:MIXer:OUTput:FREQuency:MODE swept`

**Query Syntax** `SENSe<ch>:MIXer:OUTPut:FREQuency:MODE?`

**Return Type** Character

**Default** Fixed

---

### SENSe<ch>:MIXer:OUTPut:FREQuency:SIDeband <value>

**(Read-Write)** Specify whether to select the sum (High) or difference (Low) products.

- When one LO is used: Input + or - LO1 = Output frequency
- When two LOs are used: IF1 + or - LO2 = Output frequency

Use [SENS:MIX:IF:FREQ:SID](#) when two LOs are used to determine the IF1 frequency.

Use [Sens:Mixer:Stage](#) to set 1 or 2 LOs

[See Note](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Sideband value. Choose from

**LOW** - Low or Difference (-)

**HIGH** - High or Sum (+)

#### Examples

```
SENS:MIX:OUTP:FREQ:SID LOW
SENSe2:MIXer:OUTPut:FREQ:SIDeband HIGH
```

**Query Syntax** SENSE<ch>:MIXer:OUTPut:FREQuency:SIDeband?

**Return Type** Character

**Default** LOW

---

### SENSe<ch>:MIXer:OUTPut:FREQuency:STARt <value>

**(Read-Write)** Sets or returns the Output start frequency of the mixer. [See Note](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Output start frequency

#### Examples

```
SENS:MIX:OUTP:FREQ:STAR 1e9
SENSe2:MIXer:OUTPut:FREQ:STARt 1000000000
```

**Query Syntax** SENSE<ch>:MIXer:OUTPut:FREQuency:STARt?

**Return Type** Numeric

**Default** Not Applicable

---

### SENSe<ch>:MIXer:OUTPut:FREQuency:STOP <value>

**(Read-Write)** Sets or returns the Output stop frequency of the mixer. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<value> Output stop frequency

**Examples**

```
SENS:MIX:OUTP:FREQ:STOP 1e9
SENSe2:MIXer:OUTPut:FREQ:STOP 1000000000
```

**Query Syntax** SENSe<ch>:MIXer:OUTPut:FREQuency:STOP?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<ch>:MIXer:PHASe <bool>**

**(Read Write)** Sets and returns the state of SMC Phase measurements and calibrations. [Learn more.](#)

In the User Interface, there are two "enable phase" checkboxes: in the [Phase Settings dialog](#) and in the [Calibration Wizard](#). Checking one enables both. This single command also enables both.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<bool> Include Phase measurement state. Choose from

ON or 1 - Include phase in SMC measurements

OFF or 0 - Do NOT include phase in SMC measurements

**Examples**

```
SENS:MIX:PHAS 1
sense2:mixer:phase off
```

**Query Syntax** SENSe<ch>:MIXer:PHASe?

**Return Type** Boolean

**Default** 0 (OFF)

---

**SENSe<ch>:MIXer:PMAP <in>,<out>**

**(Write-only)** Sets the PNA to DUT port map for FCA measurements. Use `SENSe<ch>:MIXer:PMAP:INPut?` and `SENSe<ch>:MIXer:PMAP:OUTPut?` to read these values. Learn about [selectable FCA DUT ports](#).

Changing the ports may limit your ability to use an internal second source. If a selected port is shared by one of the sources, then that source will not be available as an LO source. [Learn more about Internal second sources](#).

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<in> PNA port to connect to the DUT input.

- For SMC, choose any unused PNA port.
- For VMC, set to 1.

<output> PNA port to connect to the DUT output. Choose any unused port for SMC and VMC.

#### Examples

```
SENS:MIX:PMAP 2,1  
sense2:mixer:pmap 4,2
```

**Query Syntax** Not Applicable

**Default** 1,2

---

### SENSe<ch>:MIXer:PMAP:INPut?

**(Read-only)** Returns the PNA port that is mapped to the DUT input. Use [SENS:MIX:PMAP](#) to set this value.

Learn about [selectable FCA DUT ports](#).

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

#### Examples

```
SENS:MIX:PMAP:INP?  
sense2:mixer:pmap:input?
```

**Default** 1

---

### SENSe<ch>:MIXer:PMAP:OUTPut?



**(Read-only)** Returns the PNA port that is mapped to the DUT output. Use [SENS:MIX:PMAP](#) to set this value.

Learn about [selectable FCA DUT ports](#).

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

#### Examples

```
SENS:MIX:PMAP:OUTP?  
sense2:mixer:pmap:output?
```

**Default** 2

---

### SENSe<ch>:MIXer:ROLE:CATalog?

**(Read-only)** Returns a list of valid roles for the IMD Converter application.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

#### Examples

```
SENS:MIX:ROLE:CAT?  
sense2:mixer:role:catalog?
```

**Default** Not Applicable

---

### SENSe<ch>:MIXer:ROLE:DEvice <role>,<source>

**(Read-Write)** Assigns a configured source to the specified role for the IMD Converter application.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<role> (String) Role to which the source is assigned. Choose from:

"RF2"

"LO1"

<source> String) Source name from [Source Configuration dialog](#).

#### Examples

```
SENS:MIX:ROLE:DEV "LO1", "LO1Name"  
sense2:mixer:role:device "LO1", "LO1Name"
```

**Query Syntax** SENSe<ch>:MIXer:ROLE:DEvice? <source>

**Return Type** String

**Default** Not Applicable

---

## SENSe<ch>:MIXer:REVerse <bool>

(Read-Write) Sets whether to include SC12 sweeps during measurements.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<bool> (Boolean) Choose from:

**ON (1)** - Include the SC12 (reverse) sweep.

**OFF (0)** - Do NOT Include the SC12 (reverse) sweep.

### Examples

```
SENS:MIX:REV 1
sense2:mixer:reverse ON
```

**Query Syntax** SENSe<ch>:MIXer:REVerse?

**Return Type** Boolean

**Default** ON (1)

---

## SENSe<ch>:MIXer:SAVE <name>

(Write-only) Saves the settings for the mixer/converter test setup to a mixer attributes file.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<name> Path and file name (including .mxr extension) to save.

### Examples

```
SENSe:MIXer:SAVE "C:/Program Files/Agilent/Network
Analyzer/Documents/Mixer/MyMixer.mxr"
```

**Default** Not Applicable

---

## SENSe<ch>:MIXer:STAGe <n>

(Read-Write) Number of IF stages (LOs) used in the mixer. [See Note](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Number of stages. Choose either **1** or **2**

**Examples**

```
SENSe:MIX:STAG 2  
SENSe:MIXer:STAGE 1
```

**Query Syntax** SENSe<ch>:MIXer:STAGe?

**Return Type** Numeric

**Default** 1

---

**SENSe<ch>:MIXer:XAXis <char>**

(Read-Write) Sets or returns the swept frequency range to display on the X-axis for the IMDx channel.

For FCA measurements, use [CALC:MIXer:XAXis](#)

**Parameters**

<ch> Channel number of the IMDx Converter measurement. If unspecified, value is set to 1.

<char> Frequency range to display on the X-Axis. NOT case-sensitive. Choose from:

- **INPUT** - Input frequency range
- **LO\_1** - LO frequency range
- **OUTPUT** - Output frequency range

If the specified frequency range is not swept, the default swept range is used.

**Examples**

```
SENSe:MIXer:XAXis INPUT  
sense2:mixer:xaxis LO_1
```

**Return Type** Character

**Default** Search is made in the following order until a swept range is found:

1. OUTPUT
2. INPUT (If the OUTPUT is fixed)
3. Number of Points (If ALL ranges are fixed)

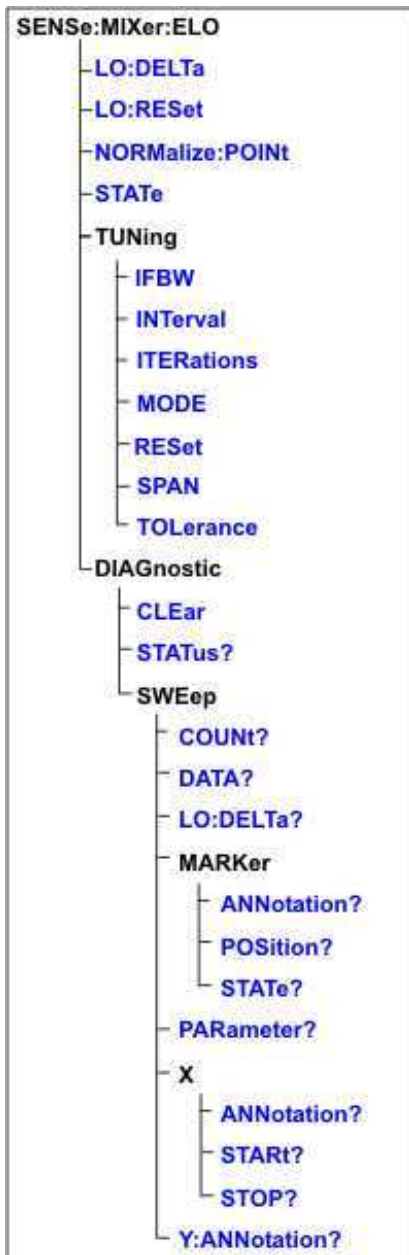
---

Last Modified:

25-Mar-2010	Added Include Phase (9.2)
12-Feb-2010	Added Include Reverse
12-Oct-2009	Clarified Stages
13-Mar-2009	Added roles and X-axis for IMDx
3-Mar-2009	Fixed Mix:Calc for INP and OUTP
24-Feb-2009	Replaced True/False
17-Dec-2008	Fixed map & typo
2-Oct-2008	Clarification of sideband commands
5-May-2008	Added Port Map commands
6-Mar-2008	Added note to page top
23-Jul-2007	Clarified LO Name command

## Sense:Mixer:ELO Commands

Allows measurements of a Mixer with an Embedded LO. These settings can be made from [FCA](#) and [Swept IMDx](#).



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Embedded LO Settings](#)
- [Synchronizing the PNA and Controller](#)

- [SCPI Command Tree](#)

**Note:** The Embedded LO **DIAGnostic** commands read data from the various broadband and precise tuning sweeps, similar to the textual and graphical data that are available in the user interface.

## SENSe<ch>:MIXer:ELO:LO:DELTA <num>

(Read-Write) Sets and returns LO Frequency Delta. There is usually no need to set this value. Read this value to determine the difference between the LO Frequency that is stated in the Mixer dialog box and the last measured LO Frequency.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<num> LO Frequency delta in Hertz.

**Examples** `SENS:MIX:ELO:LO:DELT 10.3`

**Query Syntax** `SENSe<ch>:MIXer:ELO:LO:DELTA?`

**Return Type** Numeric

**Default** Not Applicable

## SENSe<ch>:MIXer:ELO:LO:RESet

(Write-only) Resets the LO Delta Frequency to 0 (zero).

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples** `SENS:MIX:ELO:LO:RES`  
`sense2:mixer:elo:lo:reset`

**Query Syntax** Not Applicable

**Default** Not Applicable

## SENSe<ch>:MIXer:ELO:NORMALize:POINT <num>

(Read-Write) Sets and returns the sweep data point around which to perform broadband and precise tuning.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Mixer Sweep data point. Choose a data point number, between 1 and the max number of data points in the sweep, that has the least amount of expected noise.

#### Examples

```
SENS:MIX:ELO:LO:NORM:POIN 200  
sense2:mixer:elo:normalize:point 101
```

**Query Syntax** SENSE<ch>:MIXer:ELO:NORMAlize:POINT?

**Return Type** Numeric

Default Center point in the sweep span

---

### SENSe<ch>:MIXer:ELO:STATe <bool>

(Read-Write) Sets and returns the ON | OFF state of Embedded LO.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> ON | OFF state. Choose from  
0 - Embedded LO OFF  
1 - Embedded LO ON

#### Examples

```
SENS:MIX:ELO:STAT 1  
sense2:mixer:elo:state 0
```

**Query Syntax** SENSe<ch>:MIXer:ELO:STATe?

**Return Type** Boolean

Default OFF

---

### SENSe<ch>:MIXer:ELO:TUNing:IFBW <num>

(Read-Write) Sets and returns the IF Bandwidth for Broadband and Precise tuning sweeps.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1  
<num> IF Bandwidth

#### Examples

```
SENS:MIX:ELO:TUN:IFBW 10kHz  
sense2:mixer:elo:tuning:ifbw 20e3
```

**Query Syntax** SENSE<ch>:MIXer:ELO:TUNing:IFBW?

**Return Type** Numeric

Default 30kHz

---

### SENSe<ch>:MIXer:ELO:TUNing:INTERval <num>

(Read-Write) Sets and returns how often a tuning sweep is performed.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1  
<num> Tuning sweep interval

#### Examples

```
SENS:MIX:ELO:TUN:INT 2  
sense2:mixer:elo:tuning:interval 1
```

**Query Syntax** SENSE<ch>:MIXer:ELO:TUNing:INTERval?

**Return Type** Numeric

Default 1

---

### SENSe<ch>:MIXer:ELO:TUNing:ITERations <num>



(Read-Write) Sets and returns the maximum number of tuning iterations to achieve the precise tolerance.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Number of tuning iterations. Choose a number between 1 and 100.

#### Examples

```
SENS:MIX:ELO:TUN:ITER 5  
sense2:mixer:elo:tuning:iterations 3
```

**Query Syntax** SENSE<ch>:MIXer:ELO:TUNing:ITERations?

**Return Type** Numeric

Default 5

---

### SENSe<ch>:MIXer:ELO:TUNing:MODE <char>

(Read-Write) Sets and returns the method used to determine the embedded LO Frequency.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <char> Tuning mode. Choose from:
- BROadband** Both broadband and precise tuning
  - PRECise** Precise tuning only
  - NONE** No tuning; just apply the LO Frequency Delta value.

#### Examples

```
SENS:MIX:ELO:TUN:MODE BRO  
sense2:mixer:elo:tuning:mode precise
```

**Query Syntax** SENSE<ch>:MIXer:ELO:TUNing:MODE?

**Return Type** Character

Default BROadband

---

### SENSe<ch>:MIXer:ELO:TUNing:RESet

(Write-only) Resets the tuning parameters to their default values.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:MIX:ELO:TUN:RES  
sense2:mixer:elo:tuning:reset
```

**Query Syntax** Not Applicable

Default Not Applicable

---

**SENSe<ch>:MIXer:ELO:TUNing:SPAN <num>**

(Read-Write) Sets and returns the frequency span for the broadband tuning sweep.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<num> Broadband frequency span in Hz.

**Examples**

```
SENS:MIX:ELO:TUN:SPAN 1e6  
sense2:mixer:elo:tuning:span 1mhz
```

**Query Syntax** SENSe<ch>:MIXer:ELO:TUNing:SPAN?

**Return Type** Numeric

Default 3 MHz

---

**SENSe<ch>:MIXer:ELO:TUNing:TOLerance <num>**

(Read-Write) Sets and returns the tuning tolerance for precise tuning.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Tuning tolerance in Hz. Choose a number between .001 and 1e3.

#### Examples

```
SENS:MIX:ELO:TUN:TOL .5  
sense2:mixer:elo:tuning:tolerance 1
```

**Query Syntax** SENSE<ch>:MIXer:ELO:TUNing:TOLerance?

**Return Type** Numeric

Default 1 Hz

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:CLEar

(Write-only) Clears current diagnostic information.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1

#### Examples

```
SENS:MIX:ELO:DIAG:CLEar  
sense2:mixer:elo:diagnostic:clear
```

**Query Syntax** Not Applicable

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:STATus?

(Read-only) Returns a string that describes the result of the last tuning sweeps.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:MIX:ELO:DIAG:STAT?  
sense2:mixer:elo:diagnostic:status
```

**Return Type** String

Default Not Applicable

---

**SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep:COUNT?**

(Read-only) Returns the number of tuning sweeps used for the latest embedded LO measurement.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:MIX:ELO:DIAG:SWEep:COUNT  
sense2:mixer:elo:diagnostic:sweep:count?
```

**Return Type** Numeric

Default Not Applicable

---

**SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:DATA?**

(Read-only) Returns an array of data that describes the data retrieved from the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:DATA?  
sense2:mixer:elo:diagnostic:sweep1:data?
```

**Return Type** Array

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:LO:DELTA?

(Read-only) Returns the LO frequency delta from the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:LO:DELTA?  
sense2:mixer:elo:diagnostic:sweep1:lo:delta?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:MARKer:ANNotation?

(Read-only) Returns the Y-axis marker value from the specified tuning sweep. This command assumes that a marker was used. Use SENS:MIX:ELO:DIAG:SWE:MARK:STATE? to confirm if a marker was used for the tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:MARKer:ANN?  
sense2:mixer:elo:diagnostic:sweep1:marker:annotation?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:MARKer:POSition?

(Read-only) Returns the X-axis marker position from the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:MARK:POS?  
sense2:mixer:elo:diagnostic:sweep1:marker:position?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:MARKer:STATe?

(Read-only) Returns whether or not a marker was used for the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:MARK:POS?  
sense2:mixer:elo:diagnostic:sweep1:marker:position?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:PARAmeter?

(Read-only) Returns the name of the parameter of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:PAR?  
sense2:mixer:elo:diagnostic:sweep1:parameter?
```

**Return Type** String - either "VC21" or "B,1"

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:TITLe?

(Read-only) Returns the title of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:TITL?  
sense2:mixer:elo:diagnostic:sweep1:title?
```

**Return Type** String

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:X:ANNotation?

(Read-only) Returns the X-Axis annotation of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:X:ANN?  
sense2:mixer:elo:diagnostic:sweep1:x:annotation?
```

**Return Type** String - either "Hz" or "s"

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:X:START?



(Read-only) Returns the start value of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNT?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:X:STAR?  
sense2:mixer:elo:diagnostic:sweep1:x:start?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:X:STOP?

(Read-only) Returns the stop value of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNT?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:X:STOP?  
sense2:mixer:elo:diagnostic:sweep1:x:stop?
```

**Return Type** Numeric

Default Not Applicable

---

### SENSe<ch>:MIXer:ELO:DIAGnostic:SWEep<n>:Y:ANNotation?

(Read-only) Returns Y-axis annotation value of the specified tuning sweep.

#### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <n> Tuning sweep number. Use [SENS:MIX:ELO:DIAG:SWE:COUNt?](#) to find the number of sweeps taken.

#### Examples

```
SENS:MIX:ELO:DIAG:SWE2:Y:ANN?  
sense2:mixer:elo:diagnostic:sweep1:y:annotation?
```

**Return Type** String - either "U" or "Phase"

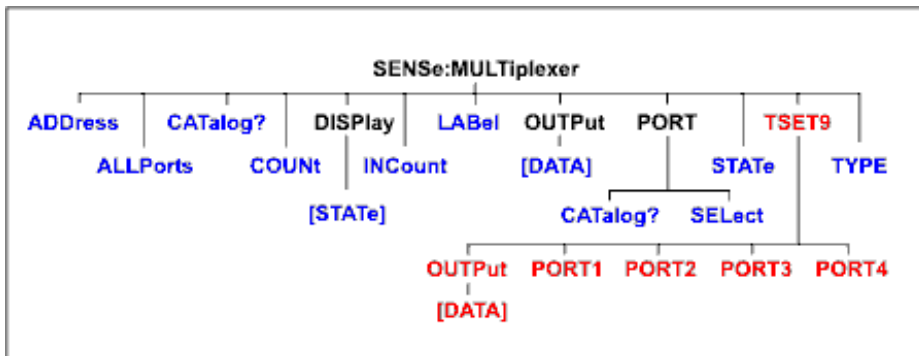
**Default** Not Applicable

---

## Sense:Multiplexer Commands

---

Controls External Test Sets (N44xx, E5091A, "Z", and "H" series).



Click on a [blue](#) keyword to view the command details.

**Red** commands are [superseded](#).

See Also

- [See an example program](#) using these commands.
- [Learn about External Test Set Control](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### SENSe:MULTiplexer<id>:ADDRess <address>

**(Read-Write)** Sets and returns the address for the external test set at the specified ID. This command should be immediately preceded by the [SENSe:MULT:TYPE](#) command.

#### Parameters

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

<address> Integer The test set address.

- For a GPIB test set (N44xx and some specials), this is the GPIB address.
- For a test set I/O test set (some specials), it is the position of the test set in the chain (starting at 0).
- For USB test sets (E5091A), the address is set by DIP switches on the rear panel.

**Examples**

```
SENS:MULT1:TYPE "Z5623A_K66" ' use K66 test set, and reference it
through ID 1
SENS:MULT1:ADDR 0 ' first test set in sequence
' All subsequent commands using SENS:MULT1 will refer to this
test set
```

**Query Syntax** SENSE:MULTiplexer<id>:ADDRESS?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<cnum>:MULTiplexer<id>:ALLPorts <string>**

(Read-Write) Sets or gets the port selections for all available ports on the specified channel.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1.

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

<string> Comma-separated list of port selections, one for each port. Each port selection must correspond to one of the values returned by [SENS:MULT:PORT:CAT?](#).

Do NOT include + and - .

**Examples**

```
' for channel 5 and test set 1, set port 1 to T1,
' port 2 to A, port 3 to R2+, port 4 to R3-.
SENS5:MULT1:ALLP "T1,A,R2,R3 "
```

**Query Syntax** SENSe<cnum>:MULTiplexer<id>:ALLPorts?

**Return Type** STRING

**Default** Not Applicable

---

**SENSe:MULTiplexer:CATalog?**

(Read-Only) Returns a comma-separated list of the external test sets models that are currently supported. Choose one of these items to send [SENS:MULT1:TYPE](#).

**Examples**

```
SENS:MULT:CAT?
```

**Return Type** String

**Default** Not Applicable

---

## SENSe:MULTiplexer<id>:COUNT?

(Read-Only) Returns the total number of ports of the specified test set.

Returns 0 if no test set is connected (GPIB test sets only).

### Parameters

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

### Examples

```
SENS:MULT1:COUN?  
sense:multiplexer2:count?
```

**Return Type** Numeric

**Default** Not Applicable

---

## SENSe:MULTiplexer<id>:DISPlay[:STATe] <bool>

(Read-Write) Turns ON and OFF the display of the test set control status bar. This status bar indicates the test set that is being controlled and the current port mappings. This setting is turned ON automatically when the test set is enabled.

### Parameters

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

<bool> ON(1) Turns ON the display.  
OFF (0) Turns OFF the display.

### Examples

```
SENS:MULT1:DISP 1  
sense:multiplexer2:display:state on
```

**Query Syntax** SENSe:MULTiplexer<id>:DISPlay[:STATe]?

**Return Type** Boolean

**Default** OFF (0)

---

## SENSe:MULTiplexer<id>:INCount?

**(Read-Only)** Returns the number of input ports for the specified test set.

- For test sets such as the E5091A that do NOT use jumper cables to route the stimulus and response signals, this command returns the number of test set ports that can be connected to the PNA.
- For test sets that DO use jumper cables to route the stimulus and response signals, such as the N44xx, the return value is not valid.

**Parameters**

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

**Examples**

```
SENS3:MULT1:INC? ' returns the number of input ports for test set 1 on channel 3
```

**Return Type** Numeric

**Default** Not Applicable

---

## **SENSe<cnum>:MULTiplexer:LABel <string>**

**(Read-Write)** Sets and returns the display label for the testset on the specified channel. The label appears in a status bar at the bottom of the PNA display when [SENS:MULT:DISP](#) is set to ON.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1.

<string> Display label text.

**Examples**

```
SENS3:MULT:LAB 'High-power output'
```

**Query Syntax** SENSe<cnum>:MULTiplexer:LABel?

**Return Type** String

**Default** Not Applicable

---

## **SENSe<cnum>:MULTiplexer<id>:OUTPut[:DATA] <num>**

**(Read-Write)** Sets or returns the control line value for the specified channel.

### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1.
- <id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.
- <numr> An integer specifying the decimal value of the control line. Values are obtained by adding weights from the following table that correspond to individual lines.

Line	Weight
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128

### Note:

- The E5091A interprets SENS:MULT1:OUTP 0 as all lines OFF.
- All "Z" and "H" series test sets interpret SENS:MULT1:OUTP 0 as all lines ON.

Refer to your test set documentation for setting control line values.

### Examples

**SENS3:MULT1:OUTP 48** 'For Z5623A K64, lines 5 and 6 are OFF; all other lines are set to ON state.'

**Query Syntax** SENSE<num>:MULTiplexer<id>:OUTPut[:DATA]?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe:MULTiplexer<id>:PORT<pnum>CATalog?**

**(Read-Only)** Returns a comma-separated list of valid port selections for the specified port.

**Parameters**

- <id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.
- <pnum> Integer - Input port number for which to return valid Output port selections. Read the number of input ports for the test set using [SENS:MULT:INCount?](#)

**Examples**

```
SENS:MULT1:PORT3:CAT? ' returns the valid port selections for port 3
```

**Return Type** String

**Default** Not Applicable

---

**SENSe<cnum>:MULTiplexer<id>:PORT<logi>SElect <phys>**

**(Write-Only)** Sets and returns a port mapping for a single port. If this command creates a conflict with an existing port, the PNA will resolve the conflict.

**Note:** This command is currently not supported for the Z5623AK44.

**Parameters**

- <cnum> Channel number of the measurement. If unspecified, value is set to 1.
- <id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.
- <logi> Integer - Logical port number.
- <string> String - Physical port number.

**Examples**

```
SENS:MULT1:PORT3:SEL "4" 'sets logical port 3 to physical port 4.
```

**Return Type** String

**Default** Not Applicable

---

**SENSe:MULTiplexer<id>:STATe <bool>**



**(Read-Write)** Enables and disables (ON/OFF) the port mapping and control line output of the specified test set.

If the specified test set is not connected or not ON, then setting State ON will report an error. All other properties can be set when the test set is not connected.

When this command is set to ON, then the display of the test set status bar ([SENS:MULT:DISP](#)) is also set to ON.

#### Parameters

<id> Id of the external test set. If unspecified, Id is assumed to be 1. Must be previously set by the [SENSe:MULT:TYPE](#) command.

<bool> ON(1) Enables test set control.

OFF (0) Disables test set control.

#### Examples

```
SENS:MULT1:STAT 1  
sense2:multiplexer2:state on
```

**Query Syntax** SENSe<cnum>:MULTiplexer<id>:STATe?

**Return Type** Boolean

**Default** OFF (0)

---

**SENSe<cnum>:MULTiplexer<id>:TSET9:OUTPut[:DATA] <data>** **Superseded**

**Note:** This command is replaced with [SENS:MULT:OUTP](#)

**(Read-Write)** Sets the control lines of the specified E5091A. Control lines, provided through a E5091A front panel connector, are used to control external equipment such as a part handler. See your E5091A documentation to learn more about control lines.

#### Parameters

<cnum> Channel number of the measurement. If unspecified, value is set to 1.

<id> Id of the E5091A test set. Choose from 1 or 2. [Learn how to set ID value.](#)

<data> Data value used to set control lines. Values are obtained by adding weights from the following table that correspond to individual lines. HIGH =1; LOW=0.

Line	Weight
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128

0 - Sets all lines low

255 - Sets all lines high

#### Examples

'The following sets line 3 and 4 high. All other lines low.

```
SENS:MULT1:TSET9:OUTP 12
```

**Query Syntax** SENSE<cnum>:MULTiplexer<id>:TSET9:OUTPut[:DATA]?

**Return Type** Numeric

**Default** 0

### SENSe<cnum>:MULTiplexer<id>:TSET9:PORT1 <char> **Superseded**

**Note:** This command is replaced with [SENS:MULT:ALLPorts](#) which sets ALL ports to the specified outputs.

**(Read-Write)** Switches Port 1 of the specified E5091A to one of the available outputs.

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

<id> Id of the E5091A test set. Choose from 1 or 2. [Learn how to set ID value.](#)

<char> Output port to be switched to. Choose from:

**A**

**T1** - (If Port 2 already is connected to T1, then Port 2 will be switched to T2.)

#### Examples

```
SENS:MULT1:TSET9:PORT1 A
```

**Query Syntax** SENSE<cnum>:MULTiplexer<id>:TSET9:PORT1?

**Return Type** Character

**Default** A

---

### **SENSe<cnum>:MULTiplexer<id>:TSET9:PORT2 <char> Superseded**

**Note:** This command is replaced with [SENS:MULT:ALLPorts](#) which sets ALL ports to the specified outputs.

**(Read-Write)** Switches Port 2 of the specified E5091A to one of the available outputs.

#### **Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <id> Id of the E5091A test set. Choose from 1 or 2. [Learn how to set ID value.](#)
- <char> Output port to be switched to. Choose from:
- T1** - If Port 1 already is connected to T1, then Port 1 will be switched to A.
- T2**

#### **Examples**

```
SENS:MULT1:TSET9:PORT2 T2
```

**Query Syntax** SENSe<cnum>:MULTiplexer<id>:TSET9:PORT2?

**Return Type** Character

**Default** T1

---

### **SENSe<cnum>:MULTiplexer<id>:TSET9:PORT3 <char> Superseded**

**Note:** This command is replaced with [SENS:MULT:ALLPorts](#) which sets ALL ports to the specified outputs.

**(Read-Write)** Switches Port 3 of the specified E5091A to one of the available outputs.

#### **Parameters**

- <cnum> Any existing channel number; if unspecified, value is set to 1.
- <id> Id of the E5091A test set. Choose from 1 or 2. [Learn how to set ID value.](#)
- <char> Output port to be switched to. Choose from:
- R1** (R1+)
- R2** (R2+)

**R3** (R3+) If option 007 (7port), R2 is selected.

**Examples**

**SENS:MULT1:TSET9:PORT3 R2**

**Query Syntax**

SENSe<cnum>:MULTiplexer<id>:TSET9:PORT3?

**Return Type**

Character

**Default**

R1

---

**SENSe<cnum>:MULTiplexer<id>:TSET9:PORT4 <char> Superseded**

**Note:** This command is replaced with [SENS:MULT:ALLPorts](#) which sets ALL ports to the specified outputs.

**(Read-Write)** Switches Port 4 of the specified E5091A to one of the available outputs.

**Parameters**

<cnum> Any existing channel number; if unspecified, value is set to 1.

<id> Id of the E5091A test set. Choose from 1 or 2. [Learn how to set ID value.](#)

<char> Output port to be switched to. Choose from:

**R1** (R1-)

**R2** (R2-)

**R3** (R3-) If option 007 (7port), R2 is selected.

**Examples**

**SENS:MULT1:TSET9:PORT4 R2**

**Query Syntax**

SENSe<cnum>:MULTiplexer<id>:TSET9:PORT4?

**Return Type**

Character

**Default**

R1

---

**SENSe:MULTiplexer<id>:TYPE <name>**

**(Read-Write)** Loads a configuration file for the specified type of external test set.

This command should be immediately followed by the [SENSe:MULT:ADDRess](#) command.

#### Parameters

- <name> String The name of the type of test set. Must be one of the items in the list returned by the [SENSe:MULT:CATalog?](#) query.
- <id> Id of the external test set. Set by this command. Use consecutive values starting at 1.

#### Examples

```
SENS:MULT1:TYPE "Z5623AK66" ' use K66 test set, and reference  
it through ID 1
```

**Query Syntax** SENSE:MULTiplexer<id>:TYPE?

**Return Type** String

**Default** Not Applicable

---

Last Modified:

17-Aug-2007    Modified Data command for differences in active high and low

## SENSe:NOISe Commands

---

Controls the Noise Figure / NFX configuration and calibration.

### SENSe:NOISe:

**AVERage** <num>

| **STATe** <bool>

**BWIDth** <num>

**CALibration:METHod** <string>

**ENR:FILEname** <string>

**GAIN** <num>

**IMPedance:COUNt** <num>

**PMAP** <in>,<out>

| **INPut?**

| **OUTPut?**

**PULL[:STATe]** <bool>

**RECeiver** <char>

### SOURce:

| **CKIT** <string>

| **CONNector** <string>

**TEMPerature:AMBient** <num>

### TUNer:

| **ID** <string>

| **INPut** <string>

| **ORient[:STATe]**

| **OUTPut** <string>

Click on a [blue](#) keyword to view the command details.

### Other Noise Figure SCPI commands

The calibration commands listed in this topic are supplemental to the [Guided Cal commands](#).

- [CALC:CUSTom:DEFine](#) - creates a noise figure measurement.
- [CONtrol:NOISe:SOURce](#) or [OUTPut:MANual:NOISe\[:STATe\]](#) - turns the Noise Source ON and OFF.
- [MMEMory:LOAD:ENR](#) and [MMEM:STORe:ENR](#) - load and save ENR files.
- [SENSe:PATH:CONF:ELEMe\[n\]t\[:STATe\]](#) - sets the port 1 and port 2 noise switches.
- [SENS:CORR:ENR:CAL:-](#) manage ENR data - usually not necessary.
- [SYST:PREF:ITEM:SWIT:DEF](#) - Sets the default setting of the Noise Tuner switch
- [SENS:CORR:NOISe](#) commands - noise calibration
- [SENS:CORR:Guided](#) commands - performs most of noise cal.

#### See Also

- **Examples:**
    - [Create and Cal a Noise Figure Measurement](#)
    - [Create and Cal an NFX Measurement](#)
  - [Learn about Noise Figure Application](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

### SENSe<ch>:NOISe:AVEAge <num>

(Read-Write) Set and read the averaging factor for the noise receiver. [Learn more](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<num> Averaging value. Choose any number from 1 to 99.

#### Examples

```
SENS:NOIS:AVeR 20
```

```
sense:noise:average 10
```

**Query Syntax** SENSe:NOISe:AVEAge?

**Return Type** Numeric

**Default** 1

---

## SENSe<ch>:NOISe:AVERAge:STATe <bool>

(Read-Write) Turns noise averaging ON and OFF.

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> Averaging state. Choose from
- 0 - Noise averaging OFF
  - 1 - Noise averaging ON

### Examples

```
SENS:NOIS:AVER:STAT 0  
sense:noise:average:state 1
```

Query Syntax SENSE:NOISe:AVERAge:STATe?

Return Type Boolean

Default 0 - OFF

---

## SENSe<ch>:NOISe:BWIDth[:RESolution] <num>

(Read-Write) Set and read the bandwidth of the noise receiver. [Learn more](#)

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Bandwidth value. Choose from:

For [Sens:Noise:Receiver](#) = **NOISe** (Opt 029) choose from: 800 KHz, 2 MHz, 4 MHz, 8 MHz, or 24 MHz or the numerical equivalent, such as 8e6 and so forth.

For [Sens:Noise:Receiver](#) = **NORMal** (Opt 028) choose from: 720 kHz or 1.2 MHz

If the value does not match one of these, it is rounded up to the next valid bandwidth value.

### Examples

```
SENS:NOIS:BWID 2e6  
sense:noise:bwid:resolution 8mhz
```

Query Syntax SENSE:NOISe:BWIDth[:RESolution]?

Return Type Numeric



**Default** 4 MHz for Noise Receiver  
1.2 MHz for Normal Receiver

---

### **SENSe<ch>:NOISe:CALibration:METHod <string>**

**(Read-Write)** Set and read the method for performing a calibration on a noise channel.

#### **Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <string> Calibration method. Note case-sensitive. Choose from:
- "VectorFull" or "Vector"
  - "SParameter" (Not available for NFX measurements)
  - "ScalarFull" or "Scalar"

#### **Examples**

```
SENS:NOIS:CAL:METH "Vector"  
sense:noise:calibration:method "SParameter"
```

**Query Syntax** SENSe:NOISe:CALibration:METHod?

**Return Type** String

**Default** "VectorFull"

---

### **SENSe<ch>:NOISe:ENR:FILENAME <string>**

**(Read-Write)** Set and read the path and name of the ENR file associated with the noise source.

#### **Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <string> Full path, filename, and extension of the ENR file.

#### **Examples**

```
SENS:NOIS:ENR:FIL "c:/ProgramFiles/Agilent/Network  
Analyzer/Documents/ENR/346C.enr"  
sense:noise:enr:filename "c:/ProgramFiles/Agilent/Network  
Analyzer/Documents/ENR/346C.enr"
```

**Query Syntax** SENSe:NOISe:ENR:FILENAME?

**Return Type** String

**Default** Not applicable

---

## SENSe<ch>:NOISe:GAIN <num>

(Read-Write) Set and read the amount of gain for the noise receiver. This setting is NOT used when [Sens:Noise:Receiver](#) = **NORMal** (Opt 028)

### Parameters

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Gain value. Choose from:
- **0** - Low gain; select if the gain of your DUT is relatively high (>35 dB).
  - **15** - Medium gain; select if the gain of your DUT is about average (20 dB to 45 dB)
  - **30** - High gain; select if the gain of your DUT is relatively low (<30 dB).

[Learn more about Noise Receiver Gain setting.](#)

If the value does not match one of these, it is rounded up to the next legal value.

### Examples

```
SENS:NOIS:GAIN 15
```

```
sense:noise:gain 0
```

**Query Syntax** SENSe:NOISe:GAIN?

**Return Type** Numeric

**Default** 30

---

## SENSe<ch>:NOISe:IMPedance:COUNt <num>

**(Read-Write)** Sets the number of impedance states to use during calibrated measurements.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <num> Number of impedance states to use. Choose between 4 and the maximum number allowed by the noise tuner device. The more states that are used, the more accurate, and slower, the measurement. If the specified number exceeds the capability of the device, the measurement will use the maximum number of states the device allows.

**Examples**

```
SENS:NOIS:IMP:COUN 5  
  
sense:noise:impedance:count 7
```

**Query Syntax** SENSE:NOISE:IMPedance:COUNT?

**Return Type** Numeric

**Default** 4

---

**SENSe<ch>:NOISe:PMAP <in>,<out>**

**(Write-only)** Set and read the DUT-to-PNA port mapping for Noise Figure. Port mapping is allowed ONLY when [SENS:NOIS:REC](#) is set to **NORM** (standard PNA-X receiver).

**Parameters**

- <ch> Any existing NF or NFX channel. If unspecified, value is set to 1.
- <in> PNA port which is connected to the DUT input.
- <out> PNA port which is connected to the DUT output.

**Examples**

```
SENS:NOIS:PMAP 1,2  
  
sense:noise:pmap 2,1  
  
See example program
```

**Query Syntax** Not Applicable

**Default** 1,2

---

**SENSe<ch>:NOISe:PMAP:INPut?**

**(Read-only)** Read the PNA port number to be connected to the DUT Input.

Use [SENS:NOIS:PMAP](#) to set the port mapping.

**Parameters**

<ch> Any existing NF or NFX channel. If unspecified, value is set to 1.

**Examples**

```
SENS:NOIS:PMAP:INP?  
sense:noise:pmap:input?
```

**Return Type** Numeric

**Default** 1

---

**SENSe<ch>:NOIS:PMAP:OUTPut?**

**(Read-only)** Read the PNA port number to be connected to the DUT Output.

**Parameters**

<ch> Any existing NF or NFX channel. If unspecified, value is set to 1.

**Examples**

```
SENS:NOIS:PMAP:OUTP?  
sense:noise:pmap:output?
```

**Return Type** Numeric

**Default** 2

---

**SENSe<ch>:NOIS:PULL[:STATe] <bool>**

**(Read-Write)** Enables and disables the use of source pull technique to compute S22. [Learn more.](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<bool> Source pull technique state. Choose from:

**OFF** or **0** - Disable use of source pull technique.

**ON** or **1** - Enable use of source pull technique.

**Examples**

```
SENS:NOIS:PULL 0  
sense2:noise:pull:state ON
```

**Query Syntax** SENSe:NOIS:PULL[:STATe]?

**Return Type** Boolean

**Default** 0 - OFF

---

### **SENSe<ch>:NOISe:RECeiver <char>**

**(Read-Write)** Sets and returns the receiver to use for noise measurements.

#### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<char> Noise receiver. Choose from:

**NORMal** The standard PNA receiver. (Opt 028)

**NOISe** The low-noise receiver. (Opt 029)

#### **Examples**

```
SENS:NOIS:REC NORM
```

```
sense2:noise:receiver noise
```

**Query Syntax** SENSe:NOISe:RECeiver?

**Return Type** Character

**Default** NOISe

---

### **SENSe<ch>:NOISe:SOURce:CKIT <string>**

**(Read-Write)** Set and read the Cal Kit that will be used for the Noise Source adapter.

An adapter is always necessary to connect a 346C Noise Source to the PNA port 2. Select a Cal Kit that is the same type and gender as the noise source connector.

If the Noise Source mates directly to PNA port 2, then set this type to "None".

#### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<string> Cal Kit. Case sensitive.

To read possible cal kit strings for the adapter:

- Change the port connector type to that of the noise source using:  
[SENS:CORR:COLL:GUID:CONN:PORT<n>](#)
- Then read the possible cal kit strings for that port using:  
[SENS:CORR:COLL:GUID:CKIT:PORT<n>:CAT?](#)

**Examples**

```
SENS:NOIS:SOUR:CKIT "N4691-60004 ECal"  
  
sense:noise:source:ckit "
```

**Query Syntax** SENSE:NOISE:SOURce:CKIT?**Return Type** String**Default** Not applicable

---

**SENSe<ch>:NOISe:SOURce:CONNector <string>**

(Read-Write) Set and read the Noise Source connector type and gender. The Agilent 346C has an "APC 3.5 male" connector.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<string> Noise source connector type and gender. Case sensitive.

Use [SENS:CORR:COLL:GUID:CONN:CAT?](#) to read possible connector strings.

**Examples**

```
SENS:NOIS:SOUR:CONN "APC 3.5 male"  
  
sense:noise:source:connector "APC 3.5 female"
```

**Query Syntax** SENSe:NOISe:SOURce:CONNector?**Return Type** String**Default** Not applicable

---

**SENSe<ch>:NOISe:TEMPerature:AMBient <num>**

(Read-Write) Sets the temperature at which the current noise measurement is occurring. [Learn more](#)

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1

<num> Ambient temperature in Kelvin.

**Examples**

```
SENS:NOIS:TEMP:AMB 292  
  
sense:noise:temperature 289
```

**Query Syntax** SENSe:NOISe:TEMPerature:AMBient?**Return Type** Numeric**Default** 295

---

### SENSe<ch>:NOISe:TUNer:ID <string>

(Read-Write) Set and read the identity of the noise tuner. This is an ECal model and serial number string. To read the identities of the connected ECal modules, use

[SENSe:CORRection:CKIT:ECAL:LIST?](#) and

[SENSe:CORRection:CKIT:ECAL<mod>:INFormation?](#)

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<string> ECal model and serial number string. The ECal module must be connected when this command is sent.

#### Examples

```
SENS:NOIS:TUN:ID "N4691-60004 ECal 02822"
```

```
sense:noise:tuner:id ""N4691-60004 ECal 02822"
```

Query Syntax SENSE:NOISe:TUNer:ID?

Return Type String

Default Not applicable

---

### SENSe<ch>:NOISe:TUNer:INPut <string>

(Read-Write) Sets and reads the port of the ECal noise tuner that is connected to the PNA SOURCE OUT.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1

<string> ECal port identifier. Case sensitive.

#### Examples

```
SENS:NOIS:TUN:INP "B"
```

```
sense:noise:tuner:input "A"
```

Query Syntax SENSE:NOISe:TUNer:INPut?

Return Type String

Default "B"

---

### SENSe<ch>:NOISe:TUNer:ORlent[:STATe] <bool>

**(Read-Write)** Sets the state of auto orientation for a noise tuner during Noise Figure for NFX.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <bool> Auto-orientation state. Choose from:
- OFF** or **0** - Disable Auto-orientation
- ON** or **1** - Enable Auto-orientation

**Examples**

```
SENS:NOIS:TUN:ORI 0  
sense2:noise:tuner:orient:state ON
```

**Query Syntax** SENSE:NOISE:TUNer:ORient[:STATe]?

**Return Type** Boolean

**Default** 1 - ON

---

**SENSe<ch>:NOISe:TUNer:OUTPut <string>**

**(Read-Write)** Sets and reads the port of the ECal noise tuner that is connected to the CPLR THRU.

**Parameters**

- <ch> Any existing channel number. If unspecified, value is set to 1
- <string> ECal port identifier. Case sensitive.

**Examples**

```
SENS:NOIS:TUN:OUTP "B"  
sense:noise:tuner:output "A"
```

**Query Syntax** SENSE:NOISE:TUNer:OUTput?

**Return Type** String

**Default** "A"

---

Last Modified:

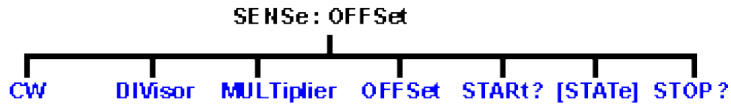
2-Mar-2010 Added Noise Receiver command (9.2)  
6-Oct-2009 Added NFX commands (9.1)  
21-Jun-2007 MX New topic



## Sense:Offset Commands **Superseded**

**Note:** These commands are replaced by the [Sense:FOM](#) commands which include the features of the new FOM dialog. Although these old commands will continue to work, they can NOT be mixed with the new commands.

Sets the offset frequency functions, causing the stimulus and response frequencies to be different.



Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Frequency Offset](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

### SENSe<num>:OFFSet:CW <bool>

**(Read-Write)** Turns stimulus CW Override mode ON or OFF. Use this setting to establish a fixed (CW) stimulus frequency while measuring the Response over a swept frequency range.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <bool> ON (or 1) - turns CW override ON.  
OFF (or 0) - turns CW override OFF.

#### Examples

```
SENS:OFFS:CW ON
sense2:offset:cw off
```

**Query Syntax** SENSE<num>:OFFSet:CW?

**Return Type** Boolean

**Default** OFF

---

### SENSe<num>:OFFSet:DIVisor <num>

(Read-Write) Specifies (along with the multiplier) the value to multiply by the stimulus.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Divisor value. Range is 1 to 1000

#### Examples

```
SENS:OFFS:DIV 3  
sense2:offset:divisor 2
```

**Query Syntax** SENSE<cnum>:OFFSet:DIVisor?

**Return Type** Numeric

Default 1

---

### SENSe<cnum>:OFFSet:MULTiplier <num>

(Read-Write) Specifies (along with the divisor) the value to multiply by the stimulus.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Multiplier value. Range is +/- 1000. Negative multipliers cause the stimulus to sweep in decreasing direction. For mixer measurements, this would be for setups requiring the RF frequency to be less than LO frequency

#### Examples

```
SENS:OFFS:MULT 2  
sense2:offset:multiplier 4
```

**Query Syntax** SENSe<cnum>:OFFSet:MULTiplier?

**Return Type** Numeric

Default 1

---

### SENSe<cnum>:OFFSet:OFFSet <num>

(Read-Write) Specifies an absolute offset frequency in Hz. For mixer measurements, this would be the LO frequency.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> Offset frequency. Range is +/- 1000 GHz. Offsets can be positive or negative

**Examples**

```
SENS:OFFS:OFFS 1GHz  
sense2:offset:offset 1e9
```

**Query Syntax** SENSE<num>:OFFSet:OFFSet?

**Return Type** Numeric

Default 0 Hz

---

## SENSe<num>:OFFSet:START?

(Read-Only) Returns the response start frequency

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:OFFS:STAR?  
sense2:offset:start?
```

**Return Type** Numeric

Default Not applicable

---

## SENSe<num>:OFFSet:[STATe] <bool>

**(Read-Write)** Enables Frequency Offset Mode on ALL measurements that are present on the active channel. This immediately causes the source and receiver to tune to separate frequencies. The receiver frequencies are specified with the other SENS:OFFSet commands. To make the stimulus settings use the [SENS:FREQ](#) commands.

Tip: To avoid unnecessary errors, first make other offset frequency settings, then set Frequency Offset ON.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<bool> ON (or 1) - turns Frequency Offset ON.  
OFF (or 0) - turns Frequency Offset OFF.

#### Examples

```
SENS:OFFS ON  
sense2:offset:state off
```

**Query Syntax** SENSE<cnum>:OFFSet:[STATe]?

**Return Type** Boolean

Default OFF (0)

---

## SENSe<cnum>:OFFSet:STOP?

**(Read-Only)** Returns the response stop frequency.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

#### Examples

```
SENS:OFFS:STOP  
sense2:offset:stop
```

**Return Type** Numeric

Default Not applicable

---

Last Modified:

23-Jul-2007    Superseded Note.

## Sense:Path:Config Commands

---

Controls the path configuration settings.

**SENSe:PATH:CONFig**

- | [CATalog?](#)
- | [DELeTe](#)
- | [DTEExt](#)
- | [ELEMeNt](#)
  - | [CATalog?](#)
  - | [\[STATe\]](#)
  - | [VALue:CATalog](#)
- | [NAME](#)
- | [SELeCt](#)
- | [STORe](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Path Configuration](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## SENSe:PATH:CONFig:CATalog?

**(Read-only)** Returns a list of configuration names stored in the PNA.

**Examples**

**SENS : PATH : CONF : CAT ?**

**Return Type**    Comma-separated list of double-quoted strings

**Default**    Not Applicable

---

## SENSe:PATH:CONFig:DELeTe <string>

(Write-only) Deletes the specified configuration name from the PNA. The factory configurations cannot be deleted. This is the only method of distinguishing a factory configuration from a user-named configuration.

#### Parameters

<string> Configuration name to be deleted.

**Examples** `SENS:PATH:CONF:DEL "MyMixer"`

**Return Type** Not Applicable

Default Not Applicable

---

### SENSe:PATH:CONFig:DTEXT <string>

(Read-Write) Write and read descriptive text associated with the configuration. This text is displayed in the path configuration dialog. Text is generally used to describe external connections that must be made manually to complete the configuration setup.

#### Parameters

<string> Descriptive text enclosed in quotes. Double quotes are not allowed within the descriptive text.

**Examples** `SENS:PATH:CONF:DTEXT "Connect J1 jumper on the rear panel."`

**Query Syntax** `SENSe<ch>:PATH:CONFig:DTEXT?`

**Return Type** String

Default Not Applicable

---

### SENSe<ch>:PATH:CONFig:ELEMent:CATalog?

(Read-only) Returns the names of configurable elements as a comma-delimited list of strings.

See a [list of configurable elements and settings](#) for various PNA models.

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:PATH:CONF:ELEM:CAT?  
  
'returns  
"Combiner", "Src1", "Src2"
```

**Default** Not Applicable

---

### SENSe<ch>:PATH:CONFig:ELEment[:STATe] <elem>, <setting>

(Read-Write) Write or read the setting of a specified element in the current configuration.

See a [list of configurable elements and settings](#) for various PNA models.

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<elem> Name of the element for which a setting is to be made.

<setting> Element setting. Use [SENS:PATH:CONF:ELEM:VAL:CAT?](#) to return a list of valid settings for the specified element.

#### Examples

```
SENS:PATH:CONF:ELEM "Combiner", "Normal"
```

#### Query Syntax

```
SENSe<ch>:PATH:CONFig:ELEment? "Combiner"  
Returns the current state of the Combiner element.
```

**Return Type** String

**Default** Not applicable

---

### SENSe<ch>:PATH:CONFig:ELEment:VALue:CATalog? <element>

(Read-only) Returns the list of valid settings that can be used with the specified element.

See a [list of configurable elements and settings](#) for various PNA models.

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.
- <element> String. Element name for which to return valid settings.

#### Examples

```
SENS:PATH:CONF:ELEM:VAL:CAT? "Combiner"  
'returns "Normal", "Reversed"
```

Default Not Applicable

---

### SENSe<ch>:PATH:CONFig:NAME?

(Read-only) Returns the name of the current configuration only if NO individual element settings had been changed since selecting or storing a configuration. When element settings change, the path configuration name is cleared.

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:PATH:CONF:NAME?  
'returns "Default"
```

**Return Type** String

Default Not Applicable

---

### SENSe<ch>:PATH:CONFig:SElect <string>



(Write only) Loads the named configuration onto the specified channel.

Use [SENS:PATH:CONF:CAT?](#) to return the configuration names that are stored on the PNA.

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.
- <string> Configuration name. "Default" is the default factory configuration.

#### Examples

```
SENS:PATH:CONF:SEL 'default'  
sense2:path:CONFig:select "MyMixer"
```

**Query Syntax** Not Applicable

**Default** "Default"

---

### SENSe<ch>:PATH:CONFig:STORe <name>

(Write only) Saves the path configuration currently associated with channel <ch> to the specified configuration name.

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.
- <name> String. Configuration name. Factory configurations can NOT be overwritten. Specifying the name of a pre-defined factory configuration will result in an error.

#### Examples

```
SENS:PATH:CONF:STOR "MyMixer"
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

Last Modified:

24-Mar-2010 Changed CONFiguration to CONFig

14-Dec-6 MX New Topic

## Sense:Power Command

---

[Learn about Receiver Attenuation](#)

### SENSe<num>:POWer:ATTenuator <recvr>,<num>

(Read-Write) Sets the attenuation level for the specified receiver.

**Note:** Attenuation cannot be set with Sweep Type set to Power

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<recvr> Receiver to get attenuation. Choose from:

- **ARECeiver** - receiver A
- **BRECeiver** - receiver B
- **CRECeiver** - receiver C
- **DRECeiver** - receiver D

Receiver attenuation can NOT be set using [logical receiver notation](#).

<num> Attenuation value in dB. To determine how many receiver attenuators, the maximum receiver attenuation, and attenuation step size, for a PNA model, see [PNA Models and Options](#).

If a number other than these is entered, the analyzer will select the next lower valid value. For example, if 19 is entered for the E8361A/C, then 10 dB attenuation will be selected.

#### Examples

```
SENS:POW:ATT AREC,10  
sense2:power:attenuator breceiver,30
```

**Query Syntax** SENSe<num>:POWer:ATTenuator? <rec>

**Return Type** Numeric

**Default** 0

---

Last Modified:

July 9, 2009    Edit syntax  
25-Oct-2007    Edit range of values

## Sense:Pulse Commands

---

Configures the 5 pulse generators in the PNA-X.

### SENSe:PULSe

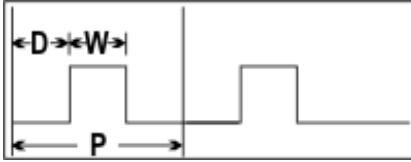
| [DELay](#)  
| [DINCrement](#)  
| [PERiod](#)  
| [STATe](#)  
| [SUBPointtrig](#)  
| [TPOLarity](#)  
| [TTYPe](#)  
| [WIDTh](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [SENS:SWEep:PULSE](#) - configures the channel for pulse measurements
- [PNA-X IF Path Block diagram](#)
- [SENS:IF configuration commands](#)
- [Example Programs](#)
- [PNA-X Pulse Application](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

## Pulse Definitions



- D = Delay; the time before each pulse begins
- W = Width; the time the pulse is ON
- P = Period; one complete pulse cycle
- Duty Cycle =  $W/P$

**Important:** If  $D + W$  is greater than  $P$ , then undefined PNA behavior results. There is NO error message or warning.

## SENSe<ch>:PULSe<n>:DELay <value>

(Read-Write) Sets the pulse delay. The amount of time before a new pulse begins.

[See Pulse Definition diagram.](#)

### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<n> Pulse generator number. Choose from 0 to 4.

0 is the generator that pulses the ADC.

<value> Delay value in seconds. Choose a value from about 33ns to about 70 seconds.

### Examples

```
SENS:PULS1:DEL .5
```

**Query Syntax** SENSe<ch>:PULSe<n>:DELay?

**Return Type** Numeric

**Default** 0

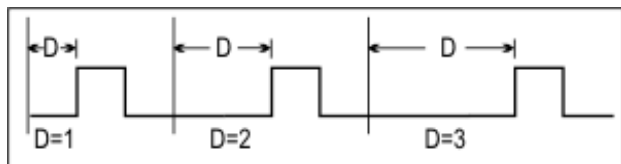
## SENSe<ch>:PULSe<n>:DINCrement <value>

**(Read-Write)** Sets the pulse delay increment. The delay increments with each pulse by the <value> amount.

For example, in this diagram the delay starts as 1. On the second pulse, delay=2. On the third pulse, delay=3.

**Important:** If  $D + W$  is greater than  $P$ , then undefined PNA behavior results. There is NO error message or warning. Delay includes the incremented value.

This is useful for pulse profiling.



[See Pulse Definition diagram.](#)

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<n> Pulse generator number. Choose from 0 to 4.

0 is the generator that pulses the ADC.

<value> Delay increment value in seconds.

#### Examples

```
SENS:PULS1:DINC .5
```

**Query Syntax** SENSE<ch>:PULSe<n>:DINCrement?

**Return Type** Numeric

**Default** 0

---

**SENSe<ch>:PULSe:PERiod <value>**

**(Read-Write)** Sets the pulse-period (1/PRF) for ALL pulse generators.

The resolution of the period is 16.667nS.

[See Pulse Definition diagram.](#)

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<value> Pulse period in seconds. Choose a value from about 33ns to about 70 seconds.

#### Examples

```
SENS:PULS:PERiod .5
```

**Query Syntax** SENSE<ch>:PULSe:PERiod?

**Return Type** Numeric

**Default** 1e-3 sec

---

### SENSe<ch>:PULSe<n>:STATe <bool>

**(Read-Write)** Turns the pulse output ON and OFF.

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<n> Pulse generator number. Choose from 0 to 4.

0 is the generator that pulses the ADC.

<bool> ON (or 1) - turns pulse output ON.

OFF (or 0) - turns pulse output OFF.

#### Examples

```
SENS:PULS1:STAT 1
```

**Query Syntax** SENSE<ch>:PULSe:STATe?

**Return Type** Boolean

**Default** OFF

---

### SENSe<ch>:PULSe<n>:SUBPointtrig <bool>

**(Read-Write)** Enables / Disables subpoint triggering. When enabled and performing [Point Averaging](#), Each rising edge of P0 triggers a subpoint (one of N acquisitions in an N point average). Must also enable the P0 generator using [SENS:PULS0:STAT](#).

[Learn more about the PNA-X pulse generators.](#)

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.
- <n> Pulse generator number. **Must be 0** as this is the generator that triggers the ADC.
- <bool> ON (or 1) - turns subpoint triggering ON.  
OFF (or 0) - turns subpoint triggering OFF.

**Examples** `SENS:PULS0:SUBP 1`

**Query Syntax** `SENSe<ch>:PULSe0:SUBPointtrig?`

**Return Type** Boolean

**Default** OFF

---

### **SENSe<ch>:PULSe:TPOLarity <char>**

**(Read-Write)** Sets the polarity of the trigger signal to which the internal pulse generators will respond when being externally triggered at the PulseSyncIn pin.

**Note:** This feature requires DSP version: **4.0 FPGA: 34** or higher. [Learn more.](#)

[Learn more about the PNA-X pulse generators.](#)

#### Parameters

- <ch> Any existing channel number; if unspecified, value is set to 1.
- <char> Pulse polarity. Choose from:  
**POSitive** - PNA responds to rising edge or HIGH level  
**NEGative** - PNA responds to falling edge or LOW level.  
Set Edge or Level triggering using [SENS:PULS:TTYPe](#).

**Examples** `SENS:PULS:TPOL NEG`

**Query Syntax** `SENSe<ch>:PULSe:TPOLarity?`

**Return Type** Character

**Default** POSitive - Also the polarity used when the PNA-X does not have the required DSP hardware.

---

### **SENSe<ch>:PULSe<n>:TTYPe <char>**

**(Read-Write)** Sets the type of trigger signal to which the internal pulse generators will respond when being externally triggered at the PulseSyncIn pin.

**Note:** This feature requires DSP version: **4.0 FPGA: 34** or higher. [Learn more.](#)

[Learn more about the PNA-X pulse generators.](#)

#### **Parameters**

<ch> Any existing channel number; if unspecified, value is set to 1.

<char> Trigger type. Choose from:

**EDGE** - PNA responds to the edge (rising or falling) of a signal

**LEVel** - PNA responds to the level (HIGH or LOW) of a signal

Set polarity using [SENS:PULS:TPOL](#)

#### **Examples**

```
SENS:PULS:TTYP EDGE
```

**Query Syntax** SENSe<ch>:PULSe:TTYPe?

**Return Type** Character

**Default** LEVel - Also the type used when the PNA-X does not have the required DSP hardware.

---

### **SENSe<ch>:PULSe<n>:WIDTh <value>**



**(Read-Write)** Sets the pulse width. The amount of time that the pulse is ON.

[See Pulse Definition diagram.](#)

#### Parameters

<ch> Any existing channel number; if unspecified, value is set to 1.

<n> Pulse generator number. Choose from 0 to 4.

0 is the generator that pulses the ADC.

<value> Pulse width in seconds. Choose a value from about 33ns to about 70 seconds.

#### Examples

```
SENS:PULS:WIDT .5
```

**Query Syntax** SENSE<ch>:PULSE<n>:WIDTH?

**Return Type** Numeric

**Default** 1e-4 sec

---

#### Last Modified:

9-Dec-2009	Added Trigger Polarity and Type (9.1.3)
20-Jul-2009	Added subpoint trig (8.55.09)
2-Jan-2007	MX New topic

## Sense:Roscillator Command

---

[Learn about the Reference Osc.](#)

### SENSe:ROSCillator:SOURce?

**(Read-only)** Applying a signal to the Reference Oscillator connector automatically sets the Reference Oscillator to EXTernal. This command allows you to check that it worked.

**EXT** is returned when a signal is present at the **Reference Oscillator** connector.

**INT** is returned when **NO** signal is present at the **Reference Oscillator** connector.

#### Examples

```
SENS:ROSC:SOUR?  
sense:roscillator:source?
```

**Return Type** Character

**Default** Not applicable

## Sense:Segment Commands

---

Defines the segment sweep settings.

Enable segment sweep with [SENS:SWE:TYPE SEGMENT](#).

### SENSe:Segment

| **ADD**

| **ARbitrary**

| **BWIDth**

| **[RESolution]**

| **CONTRol**

| **COUNT**

| **DELeTe**

| **ALL**

| **FREQuency**

| **CENTer**

| **SPAN**

| **STARt**

| **STOP**

| **LIST**

| **POWer**

| **[LEVel]**

| **CONTRol**

| **[STATe]**

| **SWEep**

| **POINTs**

| **TIME**

| **CONTRol**

| **X**

Click on a [blue](#) keyword to view the command details.

#### See Also

- Example: [Upload and Download a Segment List](#)
- [Learn about Segment Sweep](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## SENSe<cnum>:SEGMent<snum>:ADD

(Write-only) Adds a segment.

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to add. If unspecified, value is set to 1. Segment numbers must be sequential. If a new number is added where one currently exists, the existing segment and those following are incremented by one.

### Examples

Two Segments exist (1 and 2). The following command will add a new segment (1). The existing (1 and 2) will become (2 and 3) respectively.

```
SENS : SEGM1 : ADD  
sense2 : segment1 : add
```

**Query Syntax** Not applicable. Use Sense:Segment:Count to determine the number of segments in a trace.

**Default** Not Applicable

---

## SENSe<cnum>:SEGMent:ARBitrary <ON | OFF>

**(Read-Write)** Enables you to setup a segment sweep with arbitrary frequencies. The start and stop frequencies of each segment can overlap other segments. Also, each segment can have a start frequency that is greater than its stop frequency which causes a reverse sweep over that segment. Learn more about [Arbitrary Segment Sweep](#).

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <ON | OFF> **ON** (or 1) - Allows the setup of arbitrary segment sweep.  
**OFF** (or 0) - Prevents the setup of arbitrary segment sweep.

#### Examples

```
SENS:SEGM:ARB ON  
sense2:segment:arbitrary off
```

**Query Syntax** SENSE<cnum>:SEGMENT:ARBITRARY?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

### SENSe<cnum>:SEGMENT<snum>:BWIDth[:RESolution] <num>

**(Read-Write)** Sets the IF Bandwidth for the specified segment. First set SENS:SEGM:BWIDth:CONTROL ON. All subsequent segments that are added assume the new IF Bandwidth value.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to modify. Choose any existing segment number.
- <num> IF Bandwidth in Hz. The list of valid IF Bandwidths is different depending on the PNA model. [\(Click to see the lists.\)](#) If an invalid number is specified, the analyzer will round up to the closest valid number.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:SEGM:BWID 1KHZ  
sense2:segment2:bwid:resolution max
```

**Query Syntax** SENSe<cnum>:SEGMENT<snum>:BWIDth[:RESolution]?

**Return Type** Numeric

**Default** See [Preset IFBW](#) for your PNA model.

---

### SENSe<cnum>:SEGMENT:BWIDth[:RESolution]:CONTrol <ON | OFF>

**(Read-Write)** Specifies whether the IF Bandwidth resolution can be set independently for each segment.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns Bandwidth control ON. Bandwidth can be set for each segment  
**OFF** (or 0) - turns Bandwidth control OFF. Use channel bandwidth setting

**Examples**

```
SENS:SEGM:BWID:CONT ON  
sense2:segment:bandwidth:control off
```

**Query Syntax** SENSE<cnum>:SEGMent:BWIDth[:RESolution]:CONTrol?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**SENSe<cnum>:SEGMent:COUNT?**

**(Read-only)** Queries the number of segments that exist in the specified channel.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SENS:SEGM:COUNT?  
sense2:segment:count?
```

**Return Type** Numeric

**Default** 1 segment

---

**SENSe<cnum>:SEGMent<snum>:DELeTe**

**(Write-only)** Deletes the specified sweep segment. When ALL segments are deleted, [SENS:SWE:TYPE](#) is automatically set to Linear because there are no segments to sweep.

<cnum> Any existing channel number. If unspecified, value is set to 1

<snum> Number of the segment to delete. If unspecified, value is set to 1

**Examples**

```
SENS:SEGM:DEL  
sense2:segment2:delete
```

**Query Syntax** Not applicable

**Default** Not Applicable

---

## SENSe<cnum>:SEGMENT:DELeTE:ALL

(Write-only) Deletes all sweep segments. When this command is executed, [SENS:SWE:TYPE](#) is automatically set to Linear because there are no segments to sweep.

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

### Examples

```
SENS:SEGM:DEL:ALL
sense2:segment:delete:all
```

Query Syntax Not applicable

Default Not Applicable

---

## SENSe<cnum>:SEGMENT<snum>:FREQUENCY:CENTer <num>

(Read-Write) Sets the Center Frequency for the specified segment. The Frequency Span of the segment remains the same. The Start and Stop Frequencies change accordingly.

**Note:** All previous segment's Start and Stop Frequencies that are larger than the new Start Frequency are changed to the new Start Frequency. All following segment's start and stop frequencies that are smaller than the new Stop Frequency are changed to the new Stop Frequency.

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<snum> Segment number to modify. Choose any existing segment number.

<num> Center Frequency in Hz. Choose any number between the **minimum** and **maximum** frequency of the analyzer.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
SENS:SEGM:FREQ:CEN 1MHZ
sense2:segment2:frequency:center 1e9
```

Query Syntax SENSe<cnum>:SEGMENT<snum>:FREQUENCY:CENTer?

Return Type Numeric

Default Stop Frequency of the previous segment. If first segment, start frequency of the analyzer.

---

## SENSe<cnum>:SEGMENT<snum>:FREQUENCY:SPAN <num>

**(Read-Write)** Sets the Frequency Span for the specified segment. The center frequency of the segment remains the same. The start and stop frequencies change accordingly.

**Note:** All previous segment's Start and Stop Frequencies that are larger than the new Start Frequency are changed to the new Start Frequency. All following segment's start and stop frequencies that are smaller than the new Stop Frequency are changed to the new Stop Frequency.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to modify. Choose any existing segment number.
- <num> Frequency Span in Hz. Choose any number between the **minimum** and **maximum** frequency of the analyzer.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:SEGM:FREQ:SPAN 1MHZ
sense2:segment2:frequency:span max
```

**Query Syntax** SENSE<cnum>:SEGMent<snum>:FREQuency:SPAN?

**Return Type** Numeric

**Default** If first segment, frequency span of the analyzer. Otherwise 0.

---

### SENSe<cnum>:SEGMENT<snum>:FREQuency:START <num>

**(Read-Write)** Sets the Start Frequency for the specified sweep segment.

#### Notes

All other segment Start and Stop Frequency values that are larger than this frequency are changed to this frequency.

To return the start and stop frequency of the entire sweep (all segments), Use [SENS:FREQ:START?](#) and [SENS:FREQ:STOP?](#)

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to modify. Choose any existing segment number.
- <num> Start Frequency in Hz. Choose any number between the **minimum** and **maximum** frequency of the analyzer.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter.



See [SCPI Syntax](#) for more information.

**Examples**

```
SENS:SEGM:FREQ:STAR 1MHZ  
sense2:segment2:frequency:start minimum
```

**Query Syntax** SENSE<cnum>:SEGMent<snum>:FREQuency:STARt?

**Return Type** Numeric

**Default** Stop Frequency of the previous segment. If first segment, start frequency of the analyzer.

---

**SENSe<cnum>:SEGMent<snum>:FREQuency:STOP <num>**

(Read-Write) Sets the Stop Frequency for the specified sweep segment.

**Notes**

All other segment Start and Stop Frequency values that are larger than this frequency are changed to this frequency.

To return the start and stop frequency of the entire sweep (all segments), Use [SENS:FREQ:STARt?](#) and [SENS:FREQ:STOP?](#)

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to modify. Choose any existing segment number.
- <num> Stop Frequency in Hz. Choose any number between the **minimum** and **maximum** frequency of the analyzer.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
SENS:SEGM:FREQ:STOP 1MHZ  
sense2:segment2:frequency:stop maximum
```

**Query Syntax** SENSE<cnum>:SEGMent<snum>:FREQuency:STOP?

**Return Type** Numeric

**Default** If first segment, stop frequency of the analyzer. Otherwise, start frequency of the segment.

---

**SENSe<cnum>:SEGMent:LIST <char>,<numSegs>,<data>**

(Read-Write) Reads or writes the entire list of values in the segment sweep table.

**Note:** For binary data transfer, specify 64-bit instead of 32-bit using [FORMat\[:DATA\]](#). This is because higher frequencies used on PNA exceed the maximum value that can be represented by a 32-bit floating point number.

When sending/receiving this data as binary (FORMat[:DATA] REAL,64), use [FORMat:BORDER](#) to specify the correct 'endianness' (byte ordering) corresponding to your programming environment / computer platform.

### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

**SSTOP** - Frequency values are Start and Stop for each segment.

**CSPAN** - Frequency values are Center and Span for each segment.

<numSegs> Total number of sweep segments being input. This allows the PNA to determine how many values per-each-segment are in the input <data> block.

<data> A 2-dimensional array of Segment data as a single data block. The data elements within a segment are each represented as real floating-point numbers as follows, and the data block is formed by interleaving all the segments together consecutively:

1. Segment state (Boolean 1 for ON and 0 for OFF)
2. Number of Points in the segment
3. Start Freq (when <char> is SSTOP), or Center Freq (when <char> is CSPAN)
4. Stop Freq (when <char> is SSTOP), or Freq Span (when <char> is CSPAN)
5. IFBW (optional for the Write)
6. Dwell Time (optional for the Write)
7. Power (optional for the Write) - see below.

The first four data elements must always be supplied. After those values, data must be supplied for successive optional elements. For example, to set dwell time values, you must also supply IFBW values, because IFBW (#5) precedes dwell time (#6) in the array order.

The [IF Bandwidth](#), [Sweep Time](#) and [Source Power](#) Control settings do NOT affect the order in which elements are interpreted.

The number of elements to supply for Power depends on the following two

settings:

1. [Source Power Option](#) - ON allows segments to have independent power levels.
2. [Couple Ports](#) = Off allows different power levels for each test port.

CouplePorts	SourcePowerOption	Number of Elements
False	False	Each port has its own channel-wide power setting, which is set using <a href="#">SOURce:POWer[:LEVel]</a> . Provide exactly 7 elements per segment. The last element (power) is ignored.
False	True	<p>Provide 6 elements + total number of ports. The first 7 elements are still interpreted the same.</p> <p>The remaining elements (in-order) are interpreted as the power levels to set on that segment for Ports 2 through N, where N is the total number of ports currently enabled for the PNA or for a PNA with multiport external test set.</p>
True	False	Provide exactly 7 elements per segment. The last element (power) is ignored.
True	True	Provide exactly 7 elements per segment. The last element (power) is honored.

#### Examples

```
SENS:SEGM:LIST SSTOP,1,1,201,10E6,26.5E9,1E3,0,-10 1 segment,  
state ON, 201 points, 10 MHz to 26.5 GHz, 1kHz IFBW, 0 dwell  
time, -10 dBm (port powers coupled)
```

```
sense2:segment:list? cspan
```

See [Upload and Download a Segment List](#) example program

**Query Syntax**    SENSE<cnum>:SEGMent:LIST? [char].

If unspecified, char is set to SSTOP.

The number of data elements per segment returned will be 6 + total number of source ports, regardless of the [IF Bandwidth](#), [Sweep Time](#) and [Source Power](#) Control settings. For the N5264A, which has no source ports, the query will return just 6 values per segment. For all other PNA models, the last elements in each segment correspond to the power level for each port.

**Return Type** Returns block data in the format specified by [FORMat\[:DATA\]](#).

**Default** Not Applicable

---

### **SENSe<cnum>:SEGMENT<snum>:POWER[<port>][:LEVel] <num>**

(Read-Write) Sets the Port Power level for the specified sweep segment. First set SENS:SEGM:POW:CONTRol ON.

When [port power is Coupled](#), setting port power for one port will apply port power for all source ports.

All subsequent segments that are added assume the new Power Level value.

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to modify. Choose any existing segment number.
- <port> Port number of the source. If unspecified, value is set to 1.
- <num> Power level.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, send SOUR:POW? MAX and SOUR:POW? MIN. ([SOUR:POW:ATT:AUTO](#) must be set to ON).

Actual achievable leveled power depends on frequency.

#### **Examples**

```
SENS:SEGM:POW 0
sense2:segment2:power1:level -10
```

**Query Syntax** SENSe<cnum>:SEGMENT<snum>:POWER[<port>][:LEVel]?

**Return Type** Numeric

**Default** 0

---

### **SENSe<cnum>:SEGMENT:POWER[:LEVel]:CONTRol <ON | OFF>**

**(Read-Write)** Specifies whether Power Level can be set independently for each segment.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - turns Power Level control ON. Power level can be set for each segment.  
**OFF** (or 0) - turns Power Level control OFF. Use the channel power level setting.

**Examples**

```
SENS:SEGM:POW:CONT ON  
sense2:segment:power:level:control off
```

**Query Syntax** SENSE<cnum>:SEGMent:POWer[:LEVel]:CONTRol?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**SENSe<cnum>:SEGMent<snum>[:STATe] <ON | OFF>**

**(Read-Write)** Turns the specified sweep segment ON or OFF. At least ONE segment must be ON or [Sweep Mode](#) is automatically set to **Linear**.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <snum> Segment number to be turned ON or OFF
- <ON | OFF> **ON** (or 1) - turns segment ON.  
**OFF** (or 0) - turns segment OFF.

**Examples**

```
SENS:SEGM ON  
sense2:segment2:state off
```

**Query Syntax** SENSe<cnum>:SEGMent<snum>[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**SENSe<cnum>:SEGMent<snum>:SWEep:POINts <num>**

**(Read-Write)** Sets the number of data points for the specified sweep segment.

**Parameters**

- <cnun> Any existing channel number. If unspecified, value is set to 1
- <snum> Any existing segment number. If unspecified, value is set to 1
- <num> Number of points in the segment. The total number of points in all segments cannot exceed **20001**. A segment can have as few as 1 point.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
SENS:SEGM:SWE:POIN 51
sense2:segment2:sweep:points maximum
```

**Query Syntax** SENSE<cnun>:SEGMent<snum>:SWEep:POINTs?

**Return Type** Numeric

**Default** 21

---

**SENSe<cnun>:SEGMent<snum>:SWEep:TIME <num>**

**(Read-Write)** Sets the time the analyzer takes to sweep the specified sweep segment.

**Parameters**

- <cnun> Any existing channel number. If unspecified, value is set to 1
- <snum> Any existing segment number.
- <num> Sweep time in seconds. Choose a number between **0** and **100**

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
SENS:SEGM:SWE:TIME 1ms
sense2:segment2:sweep:time .001
```

**Query Syntax** SENSE<cnun>:SEGMent<snum>:SWEep:TIME?

**Return Type** Numeric

**Default** Not Applicable

---

**SENSe<cnun>:SEGMent:SWEep:TIME:CONTRol <ON | OFF>**

**(Read-Write)** Specifies whether Sweep Time can be set independently for each sweep segment.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns Sweep Time control ON. Sweep Time can be set for each segment.  
**OFF** (or 0) - turns Sweep Time control OFF. Uses the channel Sweep Time setting.

**Examples**

```
SENS:SEGM:SWE:TIME:CONT ON  
sense2:segment:sweep:time:control off
```

**Query Syntax** SENSE<cnum>:SEGMent:SWEep:TIME:CONTRol?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

**SENSe<cnum>:SEGMent<snum>:X:SPACing <char>**

**(Read-Write)** Sets X-axis spacing ON or OFF

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<snum> Any existing segment number. (This parameter is ignored)

<char> **LINear** - turns X-axis point spacing OFF

**OBASe** - turns X-axis point spacing ON

**Examples**

```
SENS:SEGM:X:SPACing LIN  
sense2:segment1:spacing obase
```

**Query Syntax** SENSE<cnum>:SEGMent<snum>:X:SPACing?

**Return Type** Character

**Default** LINear

---

Last Modified:

27-Oct-2009	Added Segment:List note
29-Apr-2009	Added List command (8.6)
13-May-2008	Fixed segment delete links
21-Jun-2007	Increased max number of points

## Sense:Sweep Commands

---

Specifies the sweep functions of the analyzer.

**SENSe:SWEEp:**

**DWELI**

| **AUTO**

| **SDELaY**

**GENeration**

| **POINtsweep**

**GRoups**

| **COUNt**

**MODE**

**POINts**

**PULSe More commands**

**SPEed**

**SRCPort**

**TIME**

| **AUTO**

**TRIGger**

| **DELAY**

| **MODE**

| **POINt**

**TYPE**

| **FACW**

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)



- **Example** [Triggering the PNA using SCPI](#)
  - [Learn about Sweeping](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

## SENSe<num>:SWEep:DWELI <num>

**(Read-Write)** Sets the dwell time between each sweep point.

- Dwell time is **ONLY** available with SENSe:SWEep:GENeration set to **STEPped**; It is **Not** available in **ANALOG**.
- Sending dwell = 0 is the same as setting SENS:SWE:DWEL:AUTO **ON**. Sending a dwell time > 0 sets SENS:SWE:DWEL:AUTO **OFF**.

### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> Dwell time in seconds.

This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
SENS:SWE:DWEL .1
sense2:sweep:dwell min
```

**Query Syntax** SENSe<num>:SWEep:DWELI?

**Return Type** Numeric

**Default** 0 - (**Note:** dwell time set to 0 is the same as dwell:auto ON)

---

## SENSe<num>:SWEep:DWELI:AUTO <ON | OFF>

**(Read-Write)** Specifies whether or not to automatically calculate and set the minimum possible dwell time. Setting Auto **ON** has the same effect as setting dwell time to **0**.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<ON | OFF> **ON** (or 1) - turns dwell ON.  
**OFF** (or 0) - turns dwell OFF.

**Examples**

```
SENS:SWE:DWEL:AUTO ON
sense2:sweep:dwell:auto off
```

**Query Syntax** SENSE<num>:SWEep:DWEL:AUtO?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**SENSe<num>:SWEep:DWEL:SDELaY <num>**

**(Read-Write)** Specifies the time to wait just before acquisition begins for each sweep. This delay is in addition to [Dwell Time](#) and [External Trigger](#) delay if enabled.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<num> Sweep delay in seconds.

This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

**Examples**

```
SENS:SWE:DWEL:SDEL .1
sense2:sweep:dwell:sdelay .5
```

**Query Syntax** SENSE<num>:SWEep:DWEL:SDELaY?

**Return Type** Numeric

**Default** 0

---

**SENSe<num>:SWEep:GENeration <char>**

**(Read-Write)** Sets sweep as Stepped or Analog.

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

**STEPped** - source frequency is CONSTANT during measurement of each displayed point. More accurate than ANALog. Dwell time can be set in this mode.

**ANALog** - source frequency is continuously RAMPING during measurement of each displayed point. Faster than STEPped. Sweep time (not dwell time) can be set in this mode.

**Examples**

```
SENS:SWE:GEN STEP
sense2:sweep:generation analog
```

**Query Syntax** SENSE<num>:SWEep:GENeration?

**Return Type** Character

**Default** Analog

---

**SENSe<num>:SWEep:GENeration:POINTsweep <bool>**

**(Read-Write)** Turns ON and OFF point sweep mode. When enabled, the PNA measures both the forward and reverse parameters at each frequency point before stepping to the next frequency. [Learn more.](#)

**Parameters**

<num> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

**ON** or **(1)** - Enable point sweep mode.

**OFF** or **(0)** - Disable point sweep mode.

**Examples**

```
SENS:SWE:GEN:POIN 1
sense2:sweep:generation:pointsweep off
```

**Query Syntax** SENSE<num>:SWEep:GENeration:POINTsweep?

**Return Type** Boolean

**Default** OFF

---

## SENSe<cnum>:SWEep:GROups:COUNT <num>

(Read-Write) Sets the trigger count (groups) for the specified channel. Set trigger mode to group after setting this count.

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Count (groups) number. Choose any number between: 1 and 2e6 (1 is the same as single trigger)

### Examples

```
SENS:SWE:GRO:COUN 10  
sense2:sweep:groups:count 50
```

**Query Syntax** SENSe<cnum>:SWEep:GROups:COUNT?

**Return Type** Numeric

**Default** 1

---

## SENSe<cnum>:SWEep:MODE <char>

(Read-Write) Sets the number of trigger signals the specified channel will ACCEPT.

See [Triggering the PNA Using SCPI](#).

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <char> Trigger mode. Choose from:

**HOLD** - channel will not trigger

**CONTinuous** - channel triggers indefinitely

**GROups** - channel accepts the number of triggers specified with the last [SENS:SWE:GRO:COUN <num>](#). This is one of the PNA overlapped commands. [Learn more](#).

**SINGLE** - channel accepts ONE trigger, then goes to HOLD.

**Note:** Beginning with 7.50, the SINGle argument makes it no longer necessary to use [SENS:SWE:GRO:COUN 1](#).

### Examples

```
SENS:SWE:MODE CONT  
sense2:sweep:mode hold
```

**Query Syntax** SENSe<cnum>:SWEep:MODE?

**Return Type** Character  
**Default** CONTInuous

---

### SENSe<cnum>:SWEep:POINts <num>

(Read-Write) Sets the number of data points for the measurement.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1  
<num> Choose any number between 1 and the [PNA maximum number of points](#).

This command will accept MIN or MAX instead of a numeric parameter. See [SCPI Syntax](#) for more information.

#### Examples

```
SENS:SWE:POIN 51  
sense2:sweep:points max
```

**Query Syntax** SENSe<cnum>:SWEep:POINts?

**Return Type** Numeric

**Default** 201

---

### SENSe<cnum>:SWEep:SRCPort <1 | 2> **Superseded**

This command is superseded. The [Calc:Par:Def:Ext](#) and [Calc:Par:Mod:Ext](#) can now optionally include the source port.

(Read-Write) Sets the source port when making non S-parameter measurements. Has no effect on S-parameter measurements.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1  
<1 | 2> 1 - Source power comes out Port 1  
2 - Source power comes out Port 2

#### Examples

```
SENS:SWE:SRCP 1  
sense2:sweep:srcport 2
```

**Query Syntax** SENSe<cnum>:SWEep:SRCPort?

**Return Type** Character

**Default** 1

---

## SENSe<cnum>:SWEep:SPEed <char>

(Read-Write) Sets and returns the state of Fast Sweep mode. [Learn more about Fast Sweep.](#)

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<char> Fast Sweep mode. Choose from:

**FAST** - turns Fast Sweep Mode ON

**NORMal** - turns Fast Sweep Mode OFF (Normal Mode).

### Examples

```
SENS:SWE:SPE NORM
sense2:sweep:speed fast
```

**Query Syntax** SENSe<cnum>:SWEep:SPEed?

**Return Type** Character

**Default** NORMal

---

## SENSe<cnum>:SWEep:TIME <num>

(Read-Write) Sets the time the analyzer takes to complete one sweep. If sweep time accuracy is critical, use **ONLY** the values that are attained using the up and down arrows next to the sweep time entry box. [See Sweep Time.](#)

### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Sweep time in seconds. Choose a number between **0** and **86,400** (24hrs),

To select the fastest sweep speed, either send MIN as an argument to this command, or send SENS:SWE:TIME:AUTO 1.

This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

### Examples

```
SENS:SWE:TIME 1ms
sense2:sweep:time .001
```

**Query Syntax** SENSe<cnum>:SWEep:TIME?

**Return Type** Numeric

**Default** NA

## SENSe<cnum>:SWEep:TIME:AUTO <ON | OFF>

(Read-Write) Turns the automatic sweep time function ON or OFF.

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - turns the automatic sweep time ON.
- OFF** (or 0) - turns the automatic sweep time OFF.

### Examples

```
SENS:SWE:TIME:AUTO
sense2:sweep:time:auto off
```

**Query Syntax** SENSe<cnum>:SWEep:TIME:AUTO?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

## SENSe<cnum>:SWEep:TRIGger:DELaY <num>

(Read-Write) Sets and reads the trigger delay for all measurements in the specified CHANNEL. This delay is only applied while [TRIG:SOURce EXTernal](#) and [TRIG:SCOP CURRent](#). After an external trigger is applied, the start of the sweep is delayed for the specified delay value plus any inherent latency.

To apply a trigger delay for all channels (Global), use [TRIG:DEL](#)

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Trigger delay value in seconds. Range is from 0 to 107 seconds

### Examples

```
SENS:SWE:TRIG:DELaY .003
sense2:sweep:trigger:delay 1
```

**Query Syntax** SENSe<cnum>:SWEep:TRIGger:DELaY?

**Return Type** Numeric

**Default** 0

---

## SENSe<cnum>:SWEep:TRIGger:MODE <char>

(Read-Write) Sets and reads the trigger mode for the specified channel. This determines what EACH signal will trigger. [Learn more.](#)

**Note:** Setting Point and Sweep mode forces [Trigger:SCOPE](#) = CURRent

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<char> Trigger mode. choose from:

- **CHANNEL** - Each trigger signal causes **ALL traces** in that channel to be swept.
- **SWEEP** - Each Manual or External trigger signal causes **ALL traces that share a source port** to be swept.
- **POINT** - Each Manual or External trigger signal causes one data point to be measured.
- **TRACE** - Allowed ONLY when [SENS:SWE:GEN:POIN](#) is enabled. Each trigger signal causes two identical measurements to be triggered separately - one trigger signal is required for each measurement. Other trigger mode settings cause two identical parameters to be measured simultaneously.

#### Examples

```
SENS:SWE:TRIG:MODE SWEEP  
sense2:sweep:trigger:mode point
```

**Query Syntax** SENSE<num>:SWEep:TRIGger:MODE?

**Return Type** Character

**Default** Channel

---

SENSe<num>:SWEep:TRIGger:POINT <ON | OFF> **Superseded**



This command is replaced with [SENS:SWE:TRIG:MODE POINT](#)

**(Read-Write)** Specifies whether the specified channel will measure one point for each trigger or all of the measurements in the channel. Setting any channel to POINT mode will automatically set the [TRIGger:SCOPE](#) = CURRENT.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - Channel measures one data point per trigger.
- OFF** (or 0) - All measurements in the channel made per trigger.

**Examples**

```
SENS:SWE:TRIG:POIN ON  
sense2:sweep:trigger:point off
```

**Query Syntax** SENSE<num>:SWEep:TRIGger:POINT?

**Return Type** Boolean (1 = Point, 0 = Measurement)

**Default** 0 - Measurement

---

**SENSe<num>:SWEep:TYPE <char>**

**(Read-Write)** Sets the type of analyzer sweep mode. First set sweep type, then set sweep parameters such as frequency or power settings.

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <char> Choose from:

**LINEar | LOGarithmic | POWer | CW | SEGMENT**

**Note:** SWEep TYPE cannot be set to SEGMENT if there are no segments turned ON. A segment is automatically turned ON when the analyzer is started.

**Examples**

```
SENS:SWE:TYPE LIN  
sense2:sweep:type segment
```

**Query Syntax** SENSE<num>:SWEep:TYPE?

**Return Type** Character

**Default** LINEar

---

**SENSe<num>:SWEep:TYPE:FACW <num>**

**(Read-Write)** Enables Fast CW sweep and sets the number of data points for the channel. [Sweep Type](#) must already be set to CW and FIFO must already be enabled.

#### See Also

[FIFO commands](#)

[Example program](#)

[N5264A Measurement Receiver](#)

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> Number of data points to measure in Fast CW mode. This setting overwrites the standard number of points setting for the channel.
- Set to 0 to disable Fast CW.

#### Examples

```
SENS:SWE:TYPE:FACW 1e6
sense2:sweep:type facw 1e3
```

**Query Syntax** SENSE<num>:SWEep:TYPE:FACW?

**Return Type** Numeric

**Default** 0 - Disabled

---

#### Last Modified:

6-May-2010	Added sweep type note
1-Mar-2010	Added Sweep Delay
10-Oct-2008	Added FACW and point sweep
26-Aug-2008	Added sweep speed
21-Jun-2007	Increased max number of points
18-Jun-2007	Added Single to Mode
24-Apr-2007	Clarified Sweep mode

## X Values Command

---

### SENSe<cnum>:X[:VALues]?

(Read-only) Returns the stimulus values for the specified channel. If the channel is sweeping the source backwards, the values will be in descending order.

**Note:** For channels that can have different traces with different x-axis values (like GCA, SMC and VMC), this command (SENS:X?) returns only the channel's default x-axis values. To get x-axis values for specific traces, use [CALC:X?](#)

#### Parameters

<cnum> Any existing channel number; if unspecified, value is set to 1.

#### Examples

```
SENS:X?  
sense2:x:values?
```

**Return Type** Depends on [FORM:DATA](#) command

**Default** Not applicable

---

Last Modified:

9-Apr-2010    Added Note

## Source Commands

---

Controls the power delivered to the DUT.

**SOURce:**

[CATalog?](#)

**POWer**

| [ALC:MODE](#)

| [CATalog?](#)

| [RECeiver](#)

| [FAST](#)

| [IFBW](#)

| [ITERation](#)

| [OFFSet](#)

| [REFeRence](#)

| [SAFE](#)

| [MAX](#)

| [MIN](#)

| [\[:MODE\]](#)

| [STEP](#)

| [TOLerance](#)

| [ATTenuation](#)

| [AUTO](#)

| [CENTer](#)

| [CORRection \(More\)](#)

| [COUPle](#)

| [DETector](#)

| [\[LEVel\]](#)

| [\[IMMediate\]\[AMPLitude\]](#)

| [SLOPe](#)

| [STATe](#)

| [MODE](#)

| [PORT](#)

| [STARt](#)

| [STOP](#)

| [SPAN](#)

| [STARt](#)

| [STOP](#)

Click on a [blue](#) keyword to view the command details.

**See Also**

- [Example Programs](#)
- [Learn about Power Settings](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)
- [Remotely Specifying a Source Port](#)

## SOURce<cnum>:CATalog?

**(Read-Write)** Returns a list of valid port names that can be controlled. Some ports only have string names, NOT numbers. All commands that require a port argument have provisions for specifying either a port number OR a string name.

See also: [Remotely Specifying a Source Port](#).

### Parameters

#### Examples

```
SOUR:CAT?
source:catalog

'Some PNA-X models return
"Port 1,Port 2,Port 3,Port 4,Port 1 Src2"
```

**Return Type** Comma-separated list of strings.

**Default** Not applicable

## SOURce<cnum>:POWer<port>:ALC[:MODE] <char>, [src]

**(Read-Write)** Sets and returns the ALC mode for the specified channel and port. Use [SOUR:POW:ALC:MODE:CAT?](#) to return a list of valid ALC modes for the PNA.

[Learn more about ALC mode.](#)

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <char> ALC Mode.

For the PNA-X choose from:

- **INTernal** Standard ALC loop

- **OPENloop** No ALC loop

[src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC INT
source2:power2:alc:mode openloop
source:power:alc:mode openloop, "Port 1 Src2"
```

**Query Syntax** SOURce<cnum>:POWer<port>:ALC:MODE? [src]

**Return Type** Character

**Default** INTernal

### SOURce<cnum>:POWer<port>:ALC[:MODE]:CATalog?, [src]

**(Read-only)** Returns a list of valid ALC modes for the specified channel and port number. Use the returned values to set [SOUR:POW:ALC:MODE](#).

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC:MODE:CAT?
source2:power2:alc:mode:catalog?
source:power:alc:mode:catalog? "Port 1 Src2"
```

**Return Type** Comma-separated list of strings.

**Default** Not applicable

---

## **SOURce<cnum>:POWer<port>:ALC:MODE:RECeiver <bool>, [src]**

(Read-Write) Sets and returns the state of Receiver Leveling for the specified source port.

[Learn more about Receiver Leveling](#)

### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <bool> Receiver Leveling state.

**ON** or **1** - Receiver Leveling ON

**OFF** or **0** - Receiver Leveling OFF

- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

### **Examples**

```
SOUR:POW:ALC:MODE:REC 1
source2:power2:alc:mode:receiver on
source:power:alc:mode:receiver off,"Port 1 Src2"
See ReceiverLeveling example
```

**Query Syntax** SOURce<cnum>:POWer<port>:ALC:MODE:RECeiver? [src]

**Return Type** Boolean

**Default** OFF

---

## **SOURce<cnum>:POWer<port>:ALC:MODE:RECeiver:FAST <bool>, [src]**

**(Read-Write)** Sets and returns the state of a separate IFBW setting for leveling sweeps. ON allows a higher (faster) IFBW than the measurement sweep. It also causes leveling sweeps to be noisier.

### [Learn more about Receiver Leveling](#)

#### Parameters

- <cnun> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <bool> Separate IFBW setting state.

**ON** or **1** - Separate IFBW setting. Specify IFBW using [SOUR:POW:ALC:MODE:REC:IFBW](#)

**OFF** or **0** - Same IFBW as the measurement sweep. Specify IFBW using [Sens:BWID](#)

- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC:MODE:REC:FAST 1

source2:power2:alc:mode:receiver:fast off

source:power:alc:mode:receiver:fast off,"Port 1 Src2"

See ReceiverLeveling example
```

**Query Syntax** SOURce<cnun>:POWER<port>:ALC:MODE:RECEiver:FAST? [src]

**Return Type** Boolean

**Default** ON

---

**[SOURce<cnun>:POWER<port>:ALC:MODE:RECEiver:IFBW <value>, \[src\]](#)**



(Read-Write) Sets and returns the IFBW to be used for leveling sweeps. Enable separate IFBW for leveling sweeps using [SOUR:POW:ALC:MODE:REC:FAST 1](#)

### [Learn more about Receiver Leveling](#)

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> IFBW for leveling sweeps in Hz. The list of valid IF Bandwidths is different depending on the PNA model. ([Click to see the lists.](#)) If an invalid number is specified, the analyzer will round up to the closest valid number.

This parameter supports MIN and MAX as arguments. [Learn more.](#)

- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC:MODE:REC:IFBW 100E3
```

```
source2:power2:alc:mode:receiver:ifbw 100khz
```

```
source:power:alc:mode:receiver:ifbw 70e3,"Port 1 Src2"
```

[See ReceiverLeveling example](#)

**Query Syntax** SOURce<cnum>:POWER<port>:ALC:MODE:RECeiver:IFBW? [src]

**Return Type** Numeric

**Default** 100 kHz

---

**SOURce<cnum>:POWER<port>:ALC:MODE:RECeiver:ITERation <value>, [src]**

**(Read-Write)** Sets and returns the maximum iterations to be used in order to achieve the tolerance setting.

### [Learn more about Receiver Leveling](#)

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Max iterations. Choose a value between 1 and 25.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC:MODE:REC:ITER 5

source2:power2:alc:mode:receiver:iteration 10

source:power:alc:mode:receiver:iteration 7,"Port 1 Src2"

See ReceiverLeveling example
```

**Query Syntax** SOURce<cnum>:POWer<port>:ALC:MODE:RECeiver:ITErAtion? [src]

**Return Type** Numeric

**Default** 5

---

**SOURce<cnum>:POWer<port>:ALC:MODE:RECeiver:OFFSet <value>, [src]**

(Read-Write) Sets and returns the power level offset value.

[Learn more about Receiver Leveling](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Power level offset in dB. Choose a value between +200 and -200.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:OFFS 10  
  
source2:power2:alc:mode:receiver:offset 5  
  
source:power:alc:mode:receiver:offset 7,"Port 1 Src2"  
  
See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:OFFSet? [src]

**Return Type** Numeric

**Default** 0

---

**SOURce<num>:POWER<port>:ALC:MODE:RECeiver:REFerence <rec>, [src]**

(Read-Write) Sets and returns the reference receiver to be used with Receiver Leveling.

[Learn more about Receiver Leveling](#)

**Parameters**

- <enum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <rec> (String) PNA receiver. Choose the PNA physical receiver that works with the source port <port>.

For example:<port 1> = "R1"

- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:REF 'r1'

source2:power2:alc:mode:receiver:reference 'r2'

source:power1:alc:mode:receiver:reference "r1","Port 1 Src2"

See ReceiverLeveling example
```

**Query Syntax** SOURce<cnum>:POWER<port>:ALC:MODE? [src]

**Return Type** Boolean

**Default** OFF

---

**SOURce<cnum>:POWER<port>:ALC:MODE:RECeiver:SAFE[:MODE] <bool>, [src]**

(Read-Write) Sets and returns the state of Safe Mode.

[Learn more about Receiver Leveling](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <bool> Safe mode state.

**ON** or **1** - Safe mode ON

**OFF** or **0** - Safe mode OFF

- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:SAFE 1
source2:power2:alc:mode:receiver:safe:mode on
source:power:alc:mode:receiver:safe:mode off,"Port 1 Src2"
See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:MODE?  
[src]

**Return Type** Boolean

**Default** OFF

---

**SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:MAX <value>, [src]**

(Read-Write) Sets and returns the maximum power level for Safe Mode.

[Learn more about Receiver Leveling](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Maximum power level in dB.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:SAFE:MAX 10  
  
source2:power2:alc:mode:receiver:safe:max 20  
  
source:power:alc:mode:receiver:safe:max 15,"Port 1 Src2"  
  
See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:MAX?  
[src]

**Return Type** Numeric

**Default** 30 dB

---

**SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:MIN <value>, [src]**

(Read-Write) Sets and returns the minimum power level for Safe Mode.

[Learn more about Receiver Leveling](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Minimum power level in dB.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:SAFE:MIN -50

source2:power2:alc:mode:receiver:safe:min -80

source:power:alc:mode:receiver:safe:min -40,"Port 1 Src2"

See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:MIN?  
[src]

**Return Type** Numeric

**Default** -95 dB

---

**SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:STEP <value>, [src]**

(Read-Write) Sets and returns the maximum step power level for Safe Mode.

[Learn more about Receiver Leveling](#)

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Maximum Step power level in dB.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW:ALC:MODE:REC:SAFE:STEP 2  
  
source2:power2:alc:mode:receiver:safe:step 1.5  
  
source:power:alc:mode:receiver:safe:min 2,"Port 1 Src2"  
  
See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:SAFE:STEP?  
[src]

**Return Type** Numeric

**Default** 1 dB

---

**SOURce<num>:POWER<port>:ALC:MODE:RECeiver:TOLerance <value>, [src]**



(Read-Write) Sets and returns the tolerance value for leveling sweeps.

### [Learn more about Receiver Leveling](#)

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <value> Tolerance level in dB.
- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ALC:MODE:REC:TOL .01

source2:power2:alc:mode:receiver:tolerance .5

source:power:alc:mode:receiver:tolerance .2,"Port 1 Src2"

See ReceiverLeveling example
```

**Query Syntax** SOURce<num>:POWER<port>:ALC:MODE:RECeiver:TOLerance?  
[src]

**Return Type** Numeric

**Default** .1 dB

---

### **SOURce<num>:POWER<port>:ATTenuation <num>, [src]**

(Read-Write) Sets the attenuation level for the selected channel. Sending this command turns automatic attenuation control (SOUR:POW:ATT:AUTO) to OFF. If the ports are coupled, changing the attenuation on one port will also change the attenuation on all other ports. To turn port coupling OFF use [SOURce:POWER:COUPle OFF](#).

**Note:** Attenuation cannot be set with **Sweep Type** set to **Power**

See [Sens:Power:ATT](#) to change receiver attenuation.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <num> Attenuation value. The range of settable values depends on the PNA model. To determine the valid settings, do one of the following:

- See [PNA models and options](#) to see the range and step size for each model / option.
- Perform a query using MAX, then MIN, as an argument. Example:  
`SOURce:POWer:ATT? Max` However, this will not tell you the attenuation step size.

If an invalid attenuation setting is entered, the PNA will select the next lower valid value. For example, if 19 is entered, then for an E8361A, 10 dB attenuation will be selected.

**Note:** This command will accept **MIN** or **MAX** instead of a numeric parameter. See [SCPI Syntax](#) for more information.

[src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:ATT 10

source2:power2:attenuation maximum

source:power:att 20, "Port 1 Src2"
```

**Query Syntax** `SOURce<cnum>:POWer<port>:ATTenuation? [min/max] [src]`  
`[min/max,src]`

**Return Type** Numeric

**Default** 0

---

**`SOURce<cnum>:POWer<port>:ATTenuation:AUTO <bool>, [src]`**

**(Read-Write)** Turns automatic attenuation control ON or OFF. Setting an attenuation value (using SOURce:POWer:ATTenuation <num>) sets AUTO **OFF**.

#### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <bool> **ON** (or 1) - turns coupling ON. The analyzer automatically selects the appropriate attenuation level to meet the specified power level.

**OFF** (or 0) - turns coupling OFF. Attenuation level must be set using SOURce:POWer:ATTenuation <num>.

- [src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW2:ATT:Auto On
source2:power:attenuation:auto off
sour:pow:att:auto 1, "Port 1 Src2"
```

**Query Syntax** SOURce<cnum>:POWer:ATTenuation:Auto? [src]

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**SOURce<cnum>:POWer:CENTer <num>**

**(Read-Write)** Sets the power sweep center power. Must also set:  
[SENS:SWE:TYPE POWER](#) and [SOURce:POWer:SPAN <num>](#).

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <num> Center power. Actual achievable leveled power depends on frequency.

**Examples**

```
SOUR:POW:CENt -15  
source2:power:center -7
```

**Query Syntax** SOURce<cnum>:POWer:CENTer?

**Return Type** Numeric

**Default** 0 dBm

---

**SOURce<cnum>:POWer:COUPle <ON | OFF>**

**(Read-Write)** Turns Port Power Coupling ON or OFF.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <ON | OFF> **ON** (or 1) - turns coupling ON. The same power level is used for both source ports.
- OFF** (or 0) - turns coupling OFF. Power level can be set individually for each source port.

**Examples**

```
SOUR:POW:COUP ON  
source2:power:couple off
```

**Query Syntax** SOURce<cnum>:POWer:COUPle?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON

---

**SOURce<cnum>:POWer:DETector <char> **OBSOLETE****

**(Read-Write)** The PNA models with external leveling are now OBSOLETE.

Sets the source leveling loop as Internal or External.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <char> **INTernal** - Internal leveling is applied to the source
- EXTernal** - External leveling is applied to the source through a rear-panel connector. ONLY provided on 3 GHz, 6 GHz, and 9 GHz PNA models.

#### Examples

```
SOUR:POW:DET INT
source2:power:detector external
```

**Query Syntax** SOURce<num>:POWER:DETEctor?

**Return Type** Character

**Default** INTernal

---

**SOURce<num>:POWER<port>[:LEVel][:IMMediate]  
[:AMPLitude] <num>, [src]**

**(Read-Write)** Sets the RF power output level.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, value is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <num> Source power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, perform a query using MAX, then MIN, as an argument. ([SOUR:POW:ATT:AUTO](#) must be set to ON) Example: SOURce:POWER? Max

Actual achievable leveled power depends on frequency.

[src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both

arguments are specified, [src] takes priority.

**Examples**

```
SOUR:POW1 5
source2:power:level:immediate:amplitude maximum
sour:pow 5, "Port 1 Src2"
```

**Query Syntax** SOURce<cnum>:POWer[:LEVel][:IMMediate][:AMPLitude]? [src]

**Return Type** Numeric

**Default** 0 dBm

---

**SOURce<cnum>:POWer[:LEVel]:SLOPe <num>**

(Read-Write) Sets the RF power slope value.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Slope value in db/GHz. Choose any integer between -2 and 2 (0 is no slope).

**Examples**

```
SOUR:POW:SLOP 2
source2:power:slope -2
```

**Query Syntax** SOURce<cnum>:POWer[:LEVel]:SLOPe?

**Return Type** Numeric

**Default** 0

---

**SOURce<cnum>:POWer[:LEVel]:SLOPe:STaTe <ON | OFF>**

(Read-Write) Turns Power Slope ON or OFF.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<ON|OFF> **ON** (or 1) - turns slope ON.

**OFF** (or 0) - turns slope OFF.

**Examples**

```
SOUR:POW:SLOP:STAT ON
source2:power:slope:state off
```

**Query Syntax** SOURce<cnum>:POWer[:LEVel]:SLOPe:STaTe?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF

---

## SOURce<cnum>:POWer<port>:MODE <state>, [src]

(Read-Write) Sets the state of PNA source for the specified port.

[Learn more about the PNA internal second source](#)

### Parameters

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, <port> is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <state> Source state. Choose from:

- **AUTO** Source power is turned ON when required for a measurement.
- **ON** Source power is always ON regardless of the measurement.
- **OFF** Source power is always OFF regardless of the measurement.

**Note:** ON and OFF are valid only on PNA models with two sources.

- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

### Examples

```
SOUR:POW:MODE ON  
  
source2:power4:mode OFF  
  
sour:pow:mode on, "Port 1 Src2"
```

**Query Syntax** SOURce<cnum>:POWer<port>:MODE? [src]

**Return Type** Character

**Default** Auto

---

## SOURce<cnum>:POWer<port>:PORT:START <num>, [src]

**(Read-Write)** Sets and reads the power sweep start power value for a specific port. This allows uncoupled forward and reverse power sweep ranges. Must also set [SENS:SWE:TYPE POWer](#) and [SOUR:POW:COUPle OFF](#).

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, <port> is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <num> Start power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, perform a query using MAX, then MIN, as an argument. ([SOUR:POW:ATT:AUTO](#) must be set to ON) Example: SOURce:POWer:STARt? MIN

Actual achievable leveled power depends on frequency.

- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW1:PORT:STAR -15
source2:power:port:start 5, "bal port 1"
```

**Query Syntax** SOURce<num>:POWer<port>:PORT:STARt? [src]

**Return Type** Numeric

**Default** -10 dBm

---

**SOURce<num>:POWer<port>:PORT:STOP <num>, [src]**



**(Read-Write)** Sets and reads the power sweep stop power value for a specific port. This allows uncoupled forward and reverse power sweep ranges. Must also set [SENS:SWE:TYPE POWer](#) and [SOUR:POW:COUPLE OFF](#).

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Source port number of the PNA. If unspecified, <port> is set to 1. To make settings for ports that are not simple numbers, use the [src] argument.
- <num> Stop power in dBm.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, perform a query using MAX, then MIN, as an argument. ([SOUR:POW:ATT:AUTO](#) must be set to ON) Example: SOURce:POWer:STARt? MIN

Actual achievable leveled power depends on frequency.

- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports such as an [external source](#), [true mode balanced port](#), or one of the Source 2 outputs on the 2-port 2-source PNA-x model such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW1:PORT:STOP -15
source2:power:port:stop 5, "bal port 1"
```

**Query Syntax** SOURce<num>:POWer<port>:PORT:STOP? [src]

**Return Type** Numeric

**Default** 0 dBm

---

**SOURce<num>:POWer:SPAN <num>**

(Read-Write) Sets the power sweep span power. Must also set:

[SENS:SWE:TYPE POWER](#) and [SOURce:POWer:CENTer <num>](#).

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1.
- <num> Span power. Actual achievable leveled power depends on frequency.

**Examples**

```
SOUR:POW:SPAN -15  
source2:power:span -7
```

**Query Syntax** SOURce<num>:POWer:SPAN?

**Return Type** Numeric

**Default** 0 dBm

---

### **SOURce<num>:POWer:START <num>**

(Read-Write) Sets the power sweep start power. Must also set

[SENS:SWE:TYPE POWER](#) and [SOURce:POWer:STOP <num>](#).

**Parameters**

- <num> Any existing channel number. If unspecified, value is set to 1
- <num> Start power.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, perform a query using MAX, then MIN, as an argument. ([SOUR:POW:ATT:AUTO](#) must be set to ON) Example: SOURce:POWer:START? MIN

Actual achievable leveled power depends on frequency.

**Examples**

```
SOUR:POW:STAR -15  
source2:power:start -7
```

**Query Syntax** SOURce<num>:POWer:START?

**Return Type** Numeric

**Default** 0 dBm

---

### **SOURce<num>:POWer:STOP <num>**

(Read-Write) Sets the power sweep stop power. Must also set: [SENS:SWE:TYPE POWer](#) and [SOURce:POWer:START <num>](#).

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<num> Stop power.

**Note:** The range of settable power values depends on the PNA model and if source attenuators are installed. To determine the range of values, perform a query using MAX, then MIN, as an argument. ([SOUR:POW:ATT:AUTO](#) must be set to ON) Example: SOURce:POWer:STOP? MAX

Actual achievable leveled power depends on frequency.

#### Examples

```
SOUR:POW:STOP -15
```

```
source2:power:stop -7
```

**Query Syntax** SOURce<cnum>:POWer:STOP?

**Return Type** Numeric

**Default** 0 dBm

---

Last modified:

19-Oct-2009	Fixed many broken links
15-Apr-2009	Fixed sour:pow:mode example
12-Feb-2009	Added receiver leveling
31-Jul-2008	Fix sour:pow unit in example
23-May-2008	Added uncoupled power sweep commands
25-Oct-2007	Edit test for source and rec attenuators commands
27-Jun-2007	Edited wording on Source:Cat?
10/18/06	MQQ Added Mode command

## Source:Power:Correction Commands

---

Used to perform source power calibration on internal and external sources.

**Note:** Only ONE Source Power Cal can be performed at a time.

### **SOURce:POWer:CORRection**

#### **COLLect**

- | [ABORt](#)
- | [ACQuire](#)
- | AVERage
  - | [\[COUNT\]](#)
  - | [NTOLerance](#)
- | DISPlay
  - | [\[STATe\]](#)
- | FCHeck
  - | [\[STATe\]](#)
- | ITERation
  - | [\[COUNT\]](#)
  - | [NTOLerance](#)
- | **METHod**
- | SAVE
- | SENSor
  - | [RCFactor](#)
  - | [\[FRANge\]](#)
  - | [SElect](#)
- | TABLE
  - | [DATA](#)
  - | [FREQuency](#)
  - | LOSS
    - | [\[STATe\]](#)
  - | [POINTs?](#)
  - | [SElect](#)

#### [DATA](#)

- | [PRIor](#)

#### [LEVel](#)

#### **OFFSet**

- | [\[MAGNitude\]](#)

[\[STATe\]](#)

Click on a [blue](#) keyword to view the command details.

**Red** commands are superseded.

## See Also

- [Example program](#) using these commands.
- [Template](#) for creating your own Power Meter Driver
- [Learn about Source Power Cal](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

**Note:** The [SOURce:POWer:CORRection:COLLect:ACQuire](#) command, used to step the PNA and read a power meter, cannot be sent over the GPIB unless the power meter is connected to a different GPIB interface. See the alternative methods described in the command details.

---

## **SOURce<cnum>:POWer<port>:CORRection:COLLect:ABORt**

**(Write-only)** Aborts a source power calibration sweep that is in progress.

### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Port number to correct for source power. If unspecified, value is set to 1.

### **Examples**

```
SOUR:POW:CORR:COLL:ABOR
source1:power2:correction:collect:abort
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## **SOURce<cnum>:POWer<port>:CORRection:COLLect[:ACQuire] <char>,<id>[,src][,sync]**

**Note:** With PNA Rev. 6.2, a new "id" argument has been added to this command, replacing [SOUR:POW:CORR:COLL:METH](#).

**(Write-only)** Initiates a source power cal acquisition sweep using the power sensor attached to the specified channel (A or B) on the power meter, or using the specified PNA receiver.

**Note:** This command, when used with a power meter, can NOT be sent over the GPIB unless the power meter is connected to a different GPIB interface or USB. Use one of the following methods to perform this command or its equivalent:

- If present, use the [GPIB dedicated controller port](#).

- Connect the power meter to the PNA using a [USB / GPIB interface](#) (Agilent 82357A).
- SCPI programming of the PNA using a LAN Client interface ([see example](#)).
- Send SCPI commands through the COM interface using the [SCPI String Parser](#) object.
- Directly control the Power Meter and PNA to step frequency; then acquire and store the Power reading. ([see example](#)).

## Parameters

<enum> Any existing channel number. If unspecified, value is set to 1

<port> Port number to correct for source power. If unspecified, value is set to 1.

<char> The following choices are new beginning PNA Rev 6.2. Choose from:

- **PMETer** - Power Meter is used for all readings.
- **PMReceiver** - Power meter for the first iteration; then use the reference receiver for remaining readings if necessary (same as "fast iteration" box checked on [dialog box](#))
- **RECeiver** - Use PNA measurement receiver for all readings.

<id> **String** (Not case sensitive). The power sensor or PNA receiver to use for measuring power.

- For PMETer or PMRECeiver: select "**ASENSOR**" or "**BSENSOR**". For U series USB sensors, always specify "**ASENSOR**"
- For RECeiver: select a PNA receiver to acquire readings using physical or [logical receiver notation](#).

[src] Optional argument. **String**. (NOT case sensitive). Source port. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

[sync] If this argument is specified, must also specify [src].

Choose from:

**SYNChronous** - blocks SCPI commands during standard measurement (default behavior).

**ASYNchronous** - does NOT block SCPI commands during standard measurement.

[Learn more.](#)

**Examples**

```
SOUR:POW:CORR:COLL PMET,"ASENSOR","Port 1",ASYN 'acquires power
meter readings using the A sensor, source port 1, asynchronous.
source1:power2:correction:collect:acquire receiver,"a1"
'acquires source cal readings using the reference receiver for
port 1.
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## **SOURce:POWer:CORRection:COLLect:AVERage[:COUNT] <num>**

(Read-Write) This command, along with [SOUR:POW:CORR:COLL:AVER:NTOLerance](#), allows for settling of the power sensor READINGS.

Specifies the maximum number of power readings that are taken at each stimulus point to allow for measurement settling. Each reading is averaged with the previous readings at that stimulus point.

When this average meets the [Average:NTOLerance](#) value or this number of readings has been made, the average is returned as the valid reading.

This setting is not necessary when using a PNA receiver ([SOUR:POW:CORR:COLL REC](#)) to make the measurement.

[Learn more.](#)

**Parameters**

<num> Maximum number of readings to make to allow for settling. Choose any number between 3 and 25.

**Examples**

```
SOUR:POW:CORR:COLL:AVER 2
source:power:correction:collect:average:count 3
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:AVERage[:COUNT]?

**Return Type** Numeric

**Default** 3

---

## **SOURce:POWer:CORRection:COLLect:AVERage:NTOLerance <num>**

**(Read-Write)** This command, along with [SOUR:POW:CORR:COLL:AVER:COUNT](#), allows for settling of the power sensor READINGS.

Each power reading is averaged with the previous readings at each stimulus point. When the average meets this nominal tolerance value or the [max number of readings](#) has been made, the average is returned as the valid reading.

This setting is not necessary when using a PNA receiver ([SOUR:POW:CORR:COLL REC](#)) to make the measurement.

[Learn more.](#)

#### Parameters

<num> Power measurement settling tolerance value in dB. Choose any number between 0 and 5.

#### Examples

```
SOUR:POW:CORR:COLL:AVER:NTOL .05  
source1:power2:correction:collect:average:ntolerance .003
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:AVERage:NTOLerance?

**Return Type** Numeric

**Default** .050 dBm

---

### **SOURce<cnum>:POWer:CORRection:COLLect:DISPlay[:STATe] <ON | OFF>**

**(Read-Write)** Enables and disables the display of power readings on the PNA screen. Send this command BEFORE you begin a source power cal acquisition. After the source power cal data is acquired, this setting is reset to ON.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<ON|OFF> **ON (1)** Source power calibration dialog box is shown on the PNA screen. Power readings are plotted against the Tolerance value as limit lines.

**OFF (0)** - Source power calibration dialog box is NOT shown on the PNA screen.

#### Examples

```
SOUR:POW:CORR:COLL:DISP ON  
source1:power2:correction:collect:display:state off
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:DISPlay[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** ON (1)

---



## **SOURce<cnum>:POWer:CORRection:COLLect:FCHeck[:STATe] <ON | OFF>**

**(Read-Write)** Enables and disables frequency checking of source power cal acquisition sweeps. ONLY use when you have more than one power sensor.

### **Parameters**

- <cnum>** Any existing channel number. If unspecified, value is set to 1
- <ON|OFF>** **ON (1)** turns source power cal frequency checking ON. A requested acquisition will only succeed for those frequency points which fall within a frequency range specified for the power sensor being used. An acquisition will pause in mid-sweep if the frequency is about to exceed the maximum frequency limit specified for that sensor. When the sweep is paused in this manner, a sensor connected to the other channel input of the power meter can be connected to the measurement port in place of the previous sensor, and used to complete the sweep. However, the maximum frequency specified for the second sensor would need to be sufficient for the sweep to complete. Frequency limits are specified using the [SOUR:POW:CORR:COLL:SEN](#) command.
- OFF (0)** - turns source power cal frequency checking OFF. An acquisition will use just one power sensor for the entire sweep, regardless of frequency.

### **Examples**

```
SOUR:POW:CORR:COLL:FCH ON  
source1:power2:correction:collect:fcheck:state off
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:FCHeck[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF (0)

---

## **SOURce:POWer:CORRection:COLLect:ITERation[:COUNT] <num>**

**(Read-Write)** This command, along with [SOUR:POW:CORR:COLL:ITER:NTOL](#) describes the number of ADJUSTMENTS to make to the source power.

Sets the maximum number of readings to take at each data point for iterating the source power. Power READINGS (performed by [SOUR:POW:CORR:COLL:AVER:COUNT](#) and [SOUR:POW:CORR:COLL:AVER:NTOLerance](#)) will continue to be made, and source power adjusted, until the measurement is within the [iteration tolerance value](#) or this max number of measurements has been met. The last value is the valid measurement for that data point.

[Learn more.](#)

#### Parameters

<num> Maximum number of readings. Choose any number between 1 and 100.

#### Examples

```
SOUR:POW:CORR:COLL:ITER 2  
source:power:correction:collect:iteration 3
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:ITERation[:COUNT]?

**Return Type** Numeric

**Default** 1

---

### **SOURce:POWer:CORRection:COLLect:ITERation:NTOLerance <num>**

**(Read-Write)** This command, along with [SOUR:POW:CORR:COLL:ITER:COUNT](#) describes the number of ADJUSTMENTS to make to the source power.

Sets the maximum desired deviation from the sum of the [test port power](#) and the [offset](#) value. Power READINGS (performed by [SOUR:POW:CORR:COLL:AVER:COUNT](#) and [SOUR:POW:CORR:COLL:AVER:NTOLerance](#)) will continue to be made, and source power adjusted, until a measurement is within this tolerance value or the max number of measurements has been met. The last value is the valid measurement for that data point.

[Learn more.](#)

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<port> Port number to correct for source power. If unspecified, value is set to 1.

<num> Tolerance value in dBm. Choose any number between 0 and 5

#### Examples

```
SOUR:POW:CORR:COLL:ITER:NTOL .005  
source:power:correction:collect:iteration:ntolerance .1
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:ITERation:NTOLerance?

**Return Type** Numeric

**Default** .05

---

### **SOURce<cnum>:POWer<port>:CORRection:COLLect:METHod <char> Superseded**

This command is replaced with [SOUR:POW:CORR:COLLect\[:ACQuire\]](#) which now specifies the method **and** the device. The only parameter required by that command was either **ASENSor** or **BSENSor** which are still supported but not documented.

**(Read-Write)** Selects the calibration method to be used for the source power cal acquisition.

#### **Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<port> Port number to correct for source power. If unspecified, value is set to 1.

<char> Choose from:

**NONE** - No Cal method

**PMETer** - Power Meter is used for all readings. (same as "fast iteration" box not checked on [dialog box](#))

**PMReceiver** - Power meter for the first iteration; then use the reference receiver for remaining readings if necessary (same as "fast iteration" box checked on [dialog box](#))

#### **Examples**

```
SOUR:POW:CORR:COLL:METH PMET
source1:power2:correction:collect:method pmreceiver
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:METHod?

**Return Type** Character

**Default** NONE

---

### **SOURce:POWer:CORRection:COLLect:SAVE [<RREC>]**

**(Write-only)** Applies the array of correction values after a source power calibration sweep has completed. The source power correction will then be active on the specified source port for channel <cnum>. This command does NOT save the correction values. To save correction values, [save an instrument / calibration state](#) (\*.cst file) after performing a source power cal.

#### Parameters

<RREC> Optional argument.

**RRECeiver** In addition to a source Power Cal, perform a calibration of the reference receiver used in the measurement. ONLY the Reference Receiver calibration is then saved to a Cal Set or Cal Register as specified by the current setting of [SENS:CORR:PREF:CSET:SAVE](#).

This argument only applies to standard S-parameter channels.

#### Examples

```
SOUR:POW:CORR:COLL:SAVE
source:power:correction:collect:save rreceiver
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### **SOURce<cnum>:POWer:CORRection:COLLect:<pmChan>SENSor[:FRANge] <num1>,<num2>**

**(Read-Write)** Specifies the frequency range over which the power sensors connected to the specified channels (A and B) of the power meter can be used (minimum frequency, maximum frequency). If the power meter has only a single channel, that channel is considered channel A.

#### Parameters

<cnum> Any existing channel number. If unspecified, value is set to 1

<pmChan> Power Meter channel. Choose from:

**A** - Channel A

**B** - Channel B

<num1> Minimum frequency for the sensor. If a frequency unit is not specified, Hz is assumed. No limits are placed on this value.

<num2> Maximum frequency for the sensor. If a frequency unit is not specified, Hz is assumed. No limits are placed on this value.

#### Examples

```
SOUR:POW:CORR:COLL:ASEN 100E3, 3E9
source1:power2:correction:collect:bsensor:frange 10 MHz, 18 GHz
```

---

<b>Query Syntax</b>	SOURce:POWer:CORRection:COLLect:ASENsor[:FRANge]?  SOURce:POWer:CORRection:COLLect:BSENsor[:FRANge]?
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	0,0

---

### **SOURce<cnum>:POWer:CORRection:COLLect:<pmChan>SENsor:RCFactor <num>**

**(Read-Write)** ) Specifies the reference cal factor for the power sensor connected to channel A or B of the power meter. If the power meter has only a single channel, that channel is considered channel A.

**Note:** If the sensor connected to the specified channel of the power meter contains cal factors in EPROM (such as the Agilent E-series power sensors), those will be the cal factors used during the calibration sweep. The reference cal factor value associated with this command, and any cal factors entered into the PNA for that sensor channel, will not be used.

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <pmChan> Power Meter channel. Choose from:
  - A - Channel A
  - B - Channel B
- <num> Reference cal factor in percent. Choose any number between 1 and 150.

#### **Examples**

```
SOUR:POW:CORR:COLL:ASEN:RCF 98.7
source1:power2:correction:collect:bsensor:rcfactor 105
```

<b>Query Syntax</b>	SOURce:POWer:CORRection:COLLect:ASENsor:RCFactor?  SOURce:POWer:CORRection:COLLect:BSENsor:RCFactor?
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	100

---

### **SOURce<cnum>:POWer:CORRection:COLLect:<pmChan>SENsor:SElect**

**(Read-Write)** Sets and returns the power sensor channel (A or B) to be used. This performs the same function as the **Use this sensor only** checkbox in the [Power Sensor Settings dialog](#).

**Note:** This write portion of this command is only necessary when performing an SMC calibration.

**Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<pmChan> Power Meter channel. Choose from:

A - Channel A

B - Channel B

**Examples**

```
SOUR:POW:CORR:COLL:<pmChan>SEN:SEL 'Write
```

```
source1:power2:correction:collect:bsensor:select? 1e9 'Read
```

**Query Syntax**

SOURce:POWer:CORRection:COLLect:ASENsor:SELEct? <Frequency>

SOURce:POWer:CORRection:COLLect:BSEnSor:SELEct? <Frequency>

Returns a boolean 1 or 0 (ON or OFF) indicating whether the sensor is to be used at the specified frequency.

If [frequency checking](#) is OFF, then the <Frequency> parameter is ignored. The query returns if the sensor is selected for ALL frequencies.

**Return Type** Numeric

**Default** Not Applicable

---

**SOURce<cnum>:POWer:CORRection:COLLect:TABLE:DATA <data>**

**(Read-Write)** Read or write data into the selected table. Each table can contain up to 100 segments. Use [SOUR:POW:CORR:COLL:TABL:SElect](#) to select a table.

- When the power sensor table is selected, the data is interpreted as cal factors in **percent**.
- When the loss table is selected, POSITIVE values in dB are interpreted as LOSS. To compensate for gain, use negative values. Learn more about [Power Loss Compensation](#).

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1  
<data> Data to write into the selected table.

#### Examples

```
SOURce:POWer:CORRection:COLLect:TABLE:DATA 0.12, 0.34, 0.56
```

#### Query Syntax

SOURce<num>:POWer:CORRection:COLLect:TABLE:DATA?

If the selected table is currently empty, no data is returned.

#### Return Type

Numeric - one number per table segment.

#### Default

Not Applicable

---

### **SOURce<num>:POWer:CORRection:COLLect:TABLE:FREQuency <data>**

**(Read-Write)** Read or write frequency values for the selected table (cal factor table for a power sensor, or the loss compensation table). Use [SOUR:POW:CORR:COLL:TABL:SElect](#) to select a table.

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1  
<data> Frequency data to write into the selected table.

#### Examples

```
SOURce:POWer:CORRection:COLLect:TABLE:FREQuency 10E6, 1.5E9, 9E9
```

#### Query Syntax

SOURce<num>:POWer:CORRection:COLLect:TABLE:FREQuency?

If the selected table is currently empty, no data is returned.

#### Return Type

Numeric - one number per table segment

#### Default

Not Applicable

---

### **SOURce<num>:POWer:CORRection:COLLect:TABLE:LOSS[:STATe] <ON | OFF>**

**(Read-Write)** Indicates whether or not to adjust the power readings using the values in the loss table during a source power cal sweep. Learn more about [Power Loss Compensation](#).

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <ON|OFF> **ON (or 1)** - turns use of the loss table ON.
- OFF (or 0)** - turns use of the loss table OFF.

**Examples**

```
SOUR:POW:CORR:COLL:TABL:LOSS ON
source1:power2:correction:collect:table:loss:state off
```

**Query Syntax** SOURce:POWer:CORRection:COLLect:TABLE:LOSS[:STATe]?

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF (0)

---

**SOURce<cnum>:POWer:CORRection:COLLect:TABLE:POINts?**

**(Read-only)** Returns the number of segments that are currently in the selected table.

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1

**Examples**

```
SOUR:POW:CORR:COLL:TABL:POIN?
source1:power2:correction:collect:table:points?
```

**Return Type** Numeric

**Default** 0

---

**SOURce<cnum>:POWer:CORRection:COLLect:TABLE[:SELe] <char>**



(Read-Write) Selects which table you want to write to or read from. Read or write using [SOURCE:POWER:CORREction:COLLect:TABLE:FREQuency](#) and [SOURCE:POWER:CORREction:COLLect:TABLE:DATA](#)

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<char> Choose from:

**NONE** - No table selected

**ASENSor** - Cal Factor table for Power Sensor A

**BSENSor** - Cal Factor table for Power Sensor B

**LOSS** - Loss compensation table

#### Examples

```
SOUR:POW:CORR:COLL:TABL ASEN
source1:power2:correction:collect:table:select bsensor
```

**Query Syntax** SOURCE:POWER:CORREction:COLLect:TABLE[:SElect]?

**Return Type** Character

**Default** NONE

---

### SOURCE<num>:POWER<port>:CORREction:DATA <data>[,src]

(Read-Write) Writes and reads source power calibration data.

When querying source power calibration data, if no source power cal data exists for the specified channel and source port, then no data is returned.

If a change in the instrument state causes interpolation and/or extrapolation of the source power cal, the correction data associated with this command correspond to the new instrument state (interpolated and/or extrapolated data).

If the channel is sweeping the source backwards, then the first data point is the highest frequency value; the last data point is the lowest. Use the [SENS:X:VALues?](#) command to return the X-axis values in the displayed order.

#### Parameters

<num> Any existing channel number. If unspecified, value is set to 1

<port> Port number to correct for source power. If unspecified, value is set to 1.

<data> Correction Data

[src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples** `SOURce1:POWer2:CORRection:DATA 0.12, -0.34, 0.56`

**Query Syntax** `SOURce<cnum>:POWer<port>:CORRection:DATA? [src]`

**Return Type** Depends on [FORMat:DATA](#) command

**Default** Not Applicable

---

### **SOURce<cnum>:POWer<port>:CORRection:DATA:PRIor <data>[,src]**

**(Read-Write)** Writes and reads power correction values from the previous iteration of the source power cal. Data for which the first power meter reading were within the tolerance limit, the 'prior' correction value is 0.

In all other respects, this command is the same as [SOUR:POW:CORR:DATA](#).

This command can be used to determine the final power reading at each point of the power cal, for a cal that did not pass tolerance limits. The formula for determining the power reading (in dB):

Power reading = Target power at the source port + specified power cal offset value + 'prior' iteration corr value – actual power corr value.

The "actual" value in this equation is returned with [SOUR:POW:CORR:DATA?](#)

#### **Parameters**

<cnum> Any existing channel number. If unspecified, value is set to 1

<port> Port number to correct for source power. If unspecified, value is set to 1.

<data> Correction Data

[src] **String**. (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

**Examples** `SOURce1:POWer2:CORRection:DATA:PRIor 0.12, -0.34, 0.56`

---

<b>Query Syntax</b>	SOURce<cnum>:POWer<port>:CORRection:DATA:PRIor? [src]
<b>Return Type</b>	Depends on <a href="#">FORMat:DATA</a> command
<b><u>Default</u></b>	Not Applicable

---

### **SOURce<cnum>:POWer<port>:CORRection:LEVel <num>[,src]**

**(Read-Write)** Specifies the power level that is expected at the desired reference plane (DUT input or output). This is not used for [segment sweep with independent power levels](#) or [power sweeps](#).

**Note:** Although this command still works, it is recommended that you specify cal power by setting the [test port power](#) and [offset](#) value.

#### **Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1
- <port> Port number to correct for source power. If unspecified, value is set to 1.
- <num> Cal power level in dBm. Because this could potentially be at the output of a device-under-test, no limits are placed on this value here. It is realistically limited by the specifications of the device (power sensor) that will be used for measuring the power. The power delivered to the PNA receiver must never exceed PNA specifications for the receiver!
- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### **Examples**

```
SOUR:POW:CORR:LEV 10
source1:power2:correction:level 0 dbm
```

---

<b>Query Syntax</b>	SOURce:POWer:CORRection:LEVel? [src]
<b>Return Type</b>	Numeric
<b><u>Default</u></b>	0 dBm

---

### **SOURce<cnum>:POWer<port>:CORRection:OFFSet[:MAGNitude] <num>[,src]**

**(Read-Write)** Sets or returns a power level offset from the PNA test port power. This can be a gain or loss value (in dB) to account for components you connect between the source and the reference plane of your measurement. For example, specify 10 dB to account for a 10 dB amplifier at the input of your DUT.

Cal power is the sum of the test port power setting and this offset value. Following the calibration, the PNA power readouts are adjusted to the cal power.

#### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Port number to correct for source power. If unspecified, value is set to 1.
- <num> Gain or loss value in dB. Choose a value between -200 and 200
- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.

While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

#### Examples

```
SOUR:POW:CORR:OFFS 10
source1:power2:correction:offset:magnitude -3
```

**Query Syntax** SOURce:POWer:CORRection:OFFSet[:MAGNitude]? [src]

**Return Type** Numeric

**Default** 0 dB

---

**SOURce<num>:POWer<port>:CORRection[:STATe] <bool>[,src]**

**(Read-Write)** Enables and disables source power correction for the specified port on the specified channel.

### Parameters

- <num> Any existing channel number. If unspecified, value is set to 1
- <port> Port number to correct for source power. If unspecified, value is set to 1.
- <bool> ON (or 1) turns source power correction ON.  
OFF (or 0) - turns source power correction OFF.
- [src] **String.** (NOT case sensitive). Source port. Optional. Use [SOUR:CAT?](#) to return a list of valid port names.
- While this argument can be used to make settings for ALL ports, it is designed to access ports that are not simple numbers, such as "Port 1 Src2". Otherwise, the <port> argument performs the same function. If both arguments are specified, [src] takes priority.

### Examples

```
SOUR:POW:CORR ON
source1:power2:correction:state off
```

**Query Syntax** SOURce:POWer:CORRection[:STATe]? [src]

**Return Type** Boolean (1 = ON, 0 = OFF)

**Default** OFF (0)

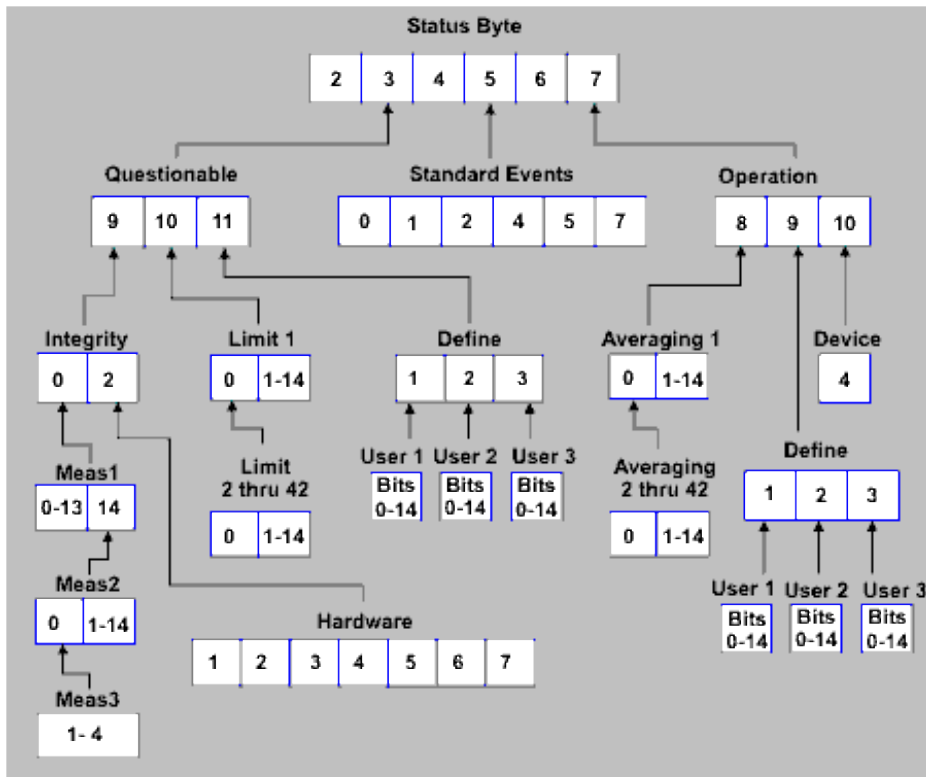
---

Last modified:

- 17-Mar-2010 Added PRIor command (9.2)
- 24-Feb-2009 Added sync to ACQ, removed old arguments
- 2-Jun-2008 Clarified Loss compensation data
- 17-Apr-2007 Removed ch and port arguments for 4 settling and accuracy commands.
- 12-Sept-2006 MQ Modified for receiver only SPC

## Status Register Commands

The status registers enable you to query the state of selected events that occur in the analyzer.



**Note:** This documentation requires familiarity with the "Standard Status Data Structure - Register Model" as defined in IEEE Std 488.2-1992. Also, first read [Learn about Status Registers](#)

Click on a [blue](#) keyword to view the command details.

### See Also

- [Example Programs](#)
- [Learn about Status Registers](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

### Notes:

- Any bit not shown in the registers is not used but may be reserved for future use.
- The [SCPIStringParser](#) can NOT be used with SCPI Status Reporting. However, the \*OPC? will work.

## Status Byte Register

Summarizes the states of the other registers and monitors the PNA output queue. It also generates **service requests**. The Enable register is called the Service Request Enable Register.

Commands	Description
----------	-------------

*CLS	Clears ALL "event" registers and the SCPI Error / Event queue. The corresponding ENABLE registers are unaffected.
------	---

*STB?	Reads the value of the analyzer's status byte. The byte remains after being read.
-------	---

*SRE?	Reads the current state of the Service Request <b>Enable</b> Register.
-------	--

*SRE <num>	Sets bits in the Service Request <b>Enable</b> register. The current setting of the SRE register is stored in non-volatile memory. Use *SRE 0 to clear the enable.
------------	--

<num> Combined value of the weights for bits to be set.

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
2	4	Error / Event queue Summary (EAV)	the Error / Event queue is not empty. To read the error message, use <a href="#">SYST:ERR?</a>
3	8	Questionable Register Summary	any enabled bit in the <b>questionable</b> event status register is set to 1
4	16	Message Available	the output queue is not empty
5	32	Standard Event Register Summary	any enabled bit in the <b>standard</b> event status register is set to 1
6	64	Request Service	any of the other bits in the status byte register is set to 1 (used to alert the controller of a service request within the analyzer). This bit cannot be disabled.
7	128	Operation Register Summary	any enabled bit in the standard <b>operation</b> event status register is set to 1

---

## STATus:QUEStionable:<keyword>

Summarizes conditions that monitor the quality of measurement data.

<keyword>    **Example**

:CONDition?    STAT:QUES:COND?

:ENABle <bits>    STAT:QUES:ENAB 1024

[:EVENT]?    STAT:QUES?

:NTRansition <bits>    STAT:QUES:NTR 1024

:PTRansition <bits>    STAT:QUES:PTR 0

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
9	512	Integrity Reg summary	any enabled bit in the <b>Integrity</b> event register is set to 1
10	1024	Limit Registers summary	any enabled bit in the <b>Limit</b> event registers is set to 1
11	2048	Define Registers summary	any enabled bit in the <b>Define</b> event registers is set to 1

---

## STATus:QUEStionable:INTegrity <keyword>

Summarizes conditions in the Measurement Integrity register.

<keyword>    **Example**

:CONDition?    STAT:QUES:INT:COND?

:ENABle <bits>    STAT:QUES:INT:ENAB 1024

[:EVENT]?    STAT:QUES:INT?

:NTRansition <bits>    STAT:QUES:INT:NTR 1024

:PTRansition <bits>    STAT:QUES:INT:PTR 0

---



Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	Measurement Summary	any bit in the <b>Measurement Integrity</b> event register is set to 1
2	4	Hardware Summary	any bit in the <b>Hardware</b> event register is set to 1

## STATus:QUESTionable:INTegrity:HARDware<keyword>

Monitors the status of hardware failures.

<keyword> **Example**

:CONDition? **STAT:QUES:INT:HARD:COND?**

:ENABle <bits> **STAT:QUES:INT:HARD:ENAB 1024**

[:EVENTt]? **STAT:QUES:INT:HARD?**

:NTRansition <bits> **STAT:QUES:INT:HARD:NTR 1024**

:PTRansition <bits> **STAT:QUES:INT:HARD:PTR 0**

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
1	2	Phase Unlock	the source has lost phaselock, possibly caused by a reference channel open or a hardware failure.
2	4	Unleveled	the source power is unleveled. This could be caused by a source set for more power than it can deliver at the tuned frequency. Or it could be caused by a hardware failure.
3	8	Not used	N/A
4	16	EE Write Failed	an attempted write to the EEPROM has failed, possibly caused by a hardware failure.
5	32	Not used	N/A
6	64	Ramp Cal Failed	the analyzer was unable to calibrate the analog ramp generator due to a possible hardware failure.
7	128	Not used	N/A

## STATus:QUESTionable:INTegrity:MEASurement<n> <keyword>

**Note:** This register can be used ONLY with standard S-parameter measurements.

Monitors the lag between changing a channel setting and when the data is ready to query.

When you change the channel state (start/stop freq, bandwidth, and so forth), then the questionable bit for that channel is set. This indicates that your desired channel state does not yet match the data you would get if querying a data trace. When the next sweep is complete (without aborting in the middle), and the data trace matches the channel state that produced it, the bit is cleared for that channel.

<n> Measurement register number. Choose from 1 to 3

<keyword> **Example**

:CONDition? STAT:QUES:INT:MEAS1:COND?

:ENABle <bits> STAT:QUES:INT:MEAS2:ENAB 1024

[:EVENTt]? STAT:QUES:INT:MEAS3?

:NTRansition <bits> STAT:QUES:INT:MEAS2:NTR 1024

:PTRansition <bits> STAT:QUES:INT:MEAS1:PTR 0

		Measurement Register <n>			Bit is set to 1 when the following conditions exist:
Bit	Weight	1	2	3	
0	1	1	Summary from Meas Reg 3		a setting change on this channel has occurred and the data does not yet reflect that change.
1	2	2	15	29	a setting change on this channel has occurred and the data does not yet reflect that change.
2	4	3	16	30	a setting change on this channel has occurred and the data does not yet reflect that change.
3	8	4	17	31	a setting change on this channel has occurred and the data does not yet reflect that change.
4	16	5	18	32	a setting change on this channel has occurred and the data does not yet reflect that change.
5	32	6	19		a setting change on this channel has occurred and the data does not yet reflect that change.
6	64	7	20		a setting change on this channel has occurred and the data does not yet reflect that change.

7	128	8	21		a setting change on this channel has occurred and the data does not yet reflect that change.
8	256	9	22		a setting change on this channel has occurred and the data does not yet reflect that change.
9	512	10	23		a setting change on this channel has occurred and the data does not yet reflect that change.
10	1024	11	24		a setting change on this channel has occurred and the data does not yet reflect that change.
11	2048	12	25		a setting change on this channel has occurred and the data does not yet reflect that change.
12	4096	13	26		a setting change on this channel has occurred and the data does not yet reflect that change.
13	8192	14	27		a setting change on this channel has occurred and the data does not yet reflect that change.
14	16384	Summary from Meas Reg 2	28		a setting change on this channel has occurred and the data does not yet reflect that change.

### **STATus:QUEStionable:LIMit<n> <keyword>**

Monitors and summarizes the status of limit line failures. When a trace fails, the representative bit is set to 1.

Bit 0 is used to summarize failures in the registers that follow. For example, Limit Register 3, bit 0, summarizes the failures from registers 4 through 42.

All enable bits are set to 1 by default.

To find the measurement number, use [Calc:Par:Mnum](#)

<n> Limit register: Choose from 1 to 42.

<keyword> **Example**

```

:CONDition? STAT:QUES:LIM4:COND?
:ENABLE <bits> STAT:QUES:LIM1:ENAB 1024
[:EVENT]? STAT:QUES:LIM3?
:NTRansition <bits> STAT:QUES:LIM2:NTR 1024
:NTRansition? STAT:QUES:LIM1:NTR?

```

:PTRansition <bits> **STAT:QUES:LIM5:PTR 0**

:PTRansition? **STAT:QUES:LIM1:PTR?**

		Limit Register <n>											Bit is set to 1 when the following conditions exist:
Bit	Weight	1	2	3	4	5	6	7	8	...	41	42	
0	1	2-42	3-42	4-42	5-42	6-42	7-42	8-42	9-42	...	42	--	Summary Bit - If any bit from that register fails, it propagates to the previous register, bit 0.
		Trace Numbers											
1	2	1	15	29	43	57	71	85	99	...	561	575	any point on trace fails the limit test
2	4	2	16	30	44	58	72	86	100	...	562	576	any point on trace fails the limit test
3	8	3	17	31	45	59	73	87	101	...	563	577	any point on trace fails the limit test
4	16	4	18	32	46	60	74	88	102	...	564	578	any point on trace fails the limit test
5	32	5	19	33	47	61	75	89	103	...	565	579	any point on trace fails the limit test
6	64	6	20	34	48	62	76	90	104	...	566	580	any point on trace fails the limit test
7	128	7	21	35	49	63	77	91	105	...	567	--	any point on trace fails the limit test
8	256	8	22	36	50	64	78	92	106	...	568	--	any point on trace fails the limit test
9	512	9	23	37	51	65	79	93	107	...	569	--	any point on trace fails the limit test
10	1024	10	24	38	52	66	80	94	108	...	570	--	any point on trace fails the limit test
11	2048	11	25	39	53	67	81	95	109	...	571	--	any point on trace fails the limit test

12	4096	12	26	40	54	68	82	96	110	...	572	--	any point on trace fails the limit test
13	8192	13	27	41	55	69	83	97	111	...	573	--	any point on trace fails the limit test
14	16384	14	28	42	56	70	84	98	112	...	574	--	any point on trace fails the limit test

To determine Register, Bit number, and Weight for trace numbers between 113 and 560 (not shown in the above table) use the following calculations.

The limit status for trace numbers higher than 580 can NOT be tracked.

The following example calculates the Register, Bit number, and Bit Weight for trace # 400:

- To determine Limit **Register** number, use  $((\text{Trace \#} - 1) / 14) + 1$ .
- To determine Limit **Bit Number**, use the **remainder** +1 of the above calculation.
- $((400-1)/14) + 1 = \text{Register\#} + 1 \text{ Bit}$ 
  - $399/14 = 28 \text{ r}7$
  - $28+1 = \text{Register } 29$
  - $7+1 = \text{Bit number } 8$
- To determine Limit **Bit Weight**: Use above table. For example: Bit 8 = **256**

## STATus:QUESTionable:DEFine<keyword>

Summarizes conditions in the Questionable:Define:User<1|2|3> event registers.

<keyword>	Example
:CONDition?	STAT:QUES:DEF:COND?
:ENABle <bits>	STAT:QUES:DEF:ENAB 1024
[:EVENTt]?	STAT:QUES:DEF?
:NTRansition <bits>	STAT:QUES:DEF:NTR 1024
:PTRansition <bits>	STAT:QUES:DEF:PTR 0

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
1	2	USER1	any bit in the <b>USER1</b> event register is set to 1
2	4	USER2	any bit in the <b>USER2</b> event register is set to 1
3	8	USER3	any bit in the <b>USER3</b> event register is set to 1

### STATus:QUEStionable:DEFine:USER<1|2|3><keyword>

Monitors conditions that you define and map in any of the three QUES:DEF:USER event registers.

<keyword> **Example**

:ENABLE <bits> **STAT:QUES:DEF:USER1:ENABLE 1024**

[ :EVENT ]? **STAT:QUES:DEF:USER1?**

:MAP <bit>,<error> **STAT:QUES:DEF:USER1:MAP 0,-113 'when error -113 occurs, bit 0 in USER1 will set to 1.**

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	for user	user defined
1	2	for user	user defined
2	4	for user	user defined
3	8	for user	user defined
4	16	for user	user defined
5	32	for user	user defined
6	64	for user	user defined
7	128	for user	user defined
8	256	for user	user defined
9	512	for user	user defined
10	1024	for user	user defined
11	2048	for user	user defined
12	4096	for user	user defined
13	8192	for user	user defined
14	16384	for user	user defined

## Standard Event Status Register

Monitors "standard" events that occur in the analyzer. This register can only be cleared by:

- a Clear Command (\*CLS).
- reading the Standard Enable Status Register (\*ESE?).
- a power-on transition. The analyzer clears the register and then records any transitions that occur, including setting the Power On bit (7).

### Commands Description

\*ESE? Reads the settings of the standard event **ENABLE** register.

\*ESE <*bits*> Sets bits in the standard event **ENABLE** register. The current setting is saved in non-volatile memory.

<*bits*> The sum of weighted bits in the register. Use \*ESE 0 to clear the enable register.

\*ESR? Reads and clears the **EVENT** settings in the Standard Event Status register.

\*OPC Sets bit 0 when the overlapped command is complete. (see Understanding Command Synchronization / OPC).

\*OPC? Operation complete query - read the Operation Complete bit (0).

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	Operation Complete	the two following events occur <b>in order</b> :  1. the *OPC command is sent to the analyzer  2. the analyzer completes all pending overlapped commands
1	NA	Request Control	Not Supported - the analyzer application is not configured to control GPIB operation
2	4	Query Error	a query error is detected indicating: - an attempt to read data from the output queue when no data was present <b>OR</b> - data in the output queue was lost, as in an overflow
4	16	Execution Error	an execution error is detected indicating: - a <PROGRAM DATA> element was outside the legal range or inconsistent with the operation of the analyzer <b>OR</b>

			- the analyzer could not execute a valid command due to some internal condition
5	32	Command Error	<p>a command error is detected indicating that the analyzer received a command that:</p> <ul style="list-style-type: none"> <li>• did not follow proper syntax</li> <li>• was misspelled</li> <li>• was an optional command it does not implement</li> </ul>
7	128	Power ON	Power to the analyzer has been turned OFF and then ON since the last time this register was read.

### STATus:OPERation<keyword>

Summarizes conditions in the Averaging and Operation:Define:User<1|2|3> event registers.

[<keyword>](#) **Example**

:CONDition? STAT:OPER:COND?

:ENABle <bits> STAT:OPER:ENAB 1024

[:EVENTt]? STAT:OPER?

:NTRansition <bits> STAT:OPER:NTR 1024

:PTRansition <bits> STAT:OPER:PTR 0

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
8	256	Averaging summary	either enabled bit in the <b>Averaging summary</b> event register is set to 1
9	512	User Defined summary	
10	1024	Device summary	either enabled bit in the <b>Device summary</b> event register is set to 1



## STATus:OPERation:AVERaging<n> <keyword>

Monitors and summarizes the status of Averaging on traces 1 to 580. When averaging for a trace is complete, the representative bit is set to 1.

Bit 0 is used to summarize the status in the registers that follow. For example, Average Register 3, bit 0, summarizes the status from registers 4 through 42.

All enable bits are set to 1 by default.

To find the measurement number, use [Calc:Par:Mnum](#).

<n> Averaging Register. Choose from 1 to 42

<keyword> **Example**

:CONDition? STAT:OPER:AVER1:COND?

:ENABle <bits> STAT:OPER:AVER1:ENAB 1024

[:EVENTt]? STAT:OPER:AVER1?

:NTRansition <bits> STAT:OPER:AVER1:NTR 1024

:PTRansition <bits> STAT:OPER:AVER1:PTR 0

Averaging Register <n>													
Bit	Weight	1	2	3	4	5	6	7	8	...	41	42	Bit is set to 1 when the following conditions exist:
0	1	2-42	3-42	4-42	5-42	6-42	7-42	8-42	9-42	...	42	--	Summary Bit - If any bit from that register fails, it propagates to the previous register, bit 0.
Trace Numbers													
1	2	1	15	29	43	57	71	85	99	...	561	575	Averaging on this trace is complete
2	4	2	16	30	44	58	72	86	100	...	562	576	Averaging on this trace is complete
3	8	3	17	31	45	59	73	87	101	...	563	577	Averaging on this trace is complete
4	16	4	18	32	46	60	74	88	102	...	564	578	Averaging on this trace is complete

5	32	5	19	33	47	61	75	89	103	...	565	579	Averaging on this trace is complete
6	64	6	20	34	48	62	76	90	104	...	566	580	Averaging on this trace is complete
7	128	7	21	35	49	63	77	91	105	...	567	--	Averaging on this trace is complete
8	256	8	22	36	50	64	78	92	106	...	568	--	Averaging on this trace is complete
9	512	9	23	37	51	65	79	93	107	...	569	--	Averaging on this trace is complete
10	1024	10	24	38	52	66	80	94	108	...	570	--	Averaging on this trace is complete
11	2048	11	25	39	53	67	81	95	109	...	571	--	Averaging on this trace is complete
12	4096	12	26	40	54	68	82	96	110	...	572	--	Averaging on this trace is complete
13	8192	13	27	41	55	69	83	97	111	...	573	--	Averaging on this trace is complete
14	16384	14	28	42	56	70	84	98	112	...	574	--	Averaging on this trace is complete

To determine Register, Bit number, and Weight for trace numbers between 113 and 560 (not shown in the above table) use the following calculations.

The averaging status for trace numbers higher than 580 can NOT be tracked.

The following example calculates the Register, Bit number, and Bit Weight for trace # 400:

- To determine **Register** number, use  $((\text{Trace \#} - 1) / 14) + 1$ .
- To determine **Bit Number**, use the **remainder** +1 of the above calculation.
- $((400-1)/14) + 1 = \text{Register\# } r+1\text{Bit}$ 
  - $399/14 = 28 \text{ r}7$
  - $28+1= \text{Register } 29$
  - $7+1= \text{Bit number } 8$

- To determine **Bit Weight**: Use above table. For example: Bit 8 = 256

### STATus:OPERation:DEFine<keyword>

Summarizes conditions in the OPERation:Define:User<1|2|3> event registers.

<keyword>	Example
:CONDition?	STAT:OPER:DEF:COND?
:ENABle <bits>	STAT:OPER:DEF:ENAB 12
[:EVENTt]?	STAT:OPER:DEF?
:NTRansition <bits>	STAT:OPER:DEF:NTR 12
:PTRansition <bits>	STAT:OPER:DEF:PTR 0

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
1	2	USER1	any bit in the <b>USER1</b> event register is set to 1
2	4	USER2	any bit in the <b>USER2</b> event register is set to 1
3	8	USER3	any bit in the <b>USER3</b> event register is set to 1

### STATus:OPERation:DEFine:USER<1|2|3><keyword>

Monitors conditions that you define and map in any of the three OPER:DEF:USER event registers.

<keyword>	Example
:ENABle <bits>	STAT:OPER:DEF:USER1:ENAB 1024
[:EVENTt]?	STAT:OPER:DEF:USER1?
:MAP <bit>,<error>	STAT:OPER:DEF:USER1:MAP 0,-113 'when error -113 occurs, bit 0 in USER1 will set to 1.

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	for user	user defined
1	2	for user	user defined
2	4	for user	user defined
3	8	for user	user defined
4	16	for user	user defined
5	32	for user	user defined
6	64	for user	user defined
7	128	for user	user defined
8	256	for user	user defined
9	512	for user	user defined
10	1024	for user	user defined
11	2048	for user	user defined
12	4096	for user	user defined
13	8192	for user	user defined
14	16384	for user	user defined

### STATus:OPERation:DEvice<keyword>

Summarizes conditions in the OPERation:DEvice event registers.

<keyword> **Example**

:CONDition? STAT:OPER:DEV:COND?

:ENABle <bits> STAT:OPER:DEV:ENAB 16

[:EVENTt]? STAT:OPER:DEV?

:NTRansition <bits> STAT:OPER:DEV:NTR 16

:PTRansition <bits> STAT:OPER:DEV:PTR 0

Bit	Weight	Description	Bit is set to 1 when the following conditions exist:
0	1	Unused	
1	2	Unused	
2	4	Unused	
3	8	Unused	
4	16	Sweep Completed	When sweep is complete
5	32	Unused	
6	64	Unused	
7	128	Unused	
8	256	Unused	
9	512	Unused	
10	1024	Unused	
11	2048	Unused	
12	4096	Unused	
13	8192	Unused	
14	16384	Unused	

Last modified:

13-Jan-2010 Changes to Limit and Average register  
16-Jul-2009 Added note to Stat:Ques:Int:Meas register  
16-Jul-2009 Fixed typo in STAT:QUES:INT register  
9/19/06 MQ Modified for unlimited windows.

## System Commands

---

Controls and queries settings that affect the PNA system.

### **SYSTem:**

#### **ACTive**

- | **CHANnel**
- | **MEASurement**

#### **CHANnels**

- | **CATalog?**
- | **COUPle**
- | **HOLD**
- | **RESume**

#### **COMMunicate**

- | **GPIB**
  - | **PMETer**
    - | **ADDRess**
  - | **RDEVice**
    - | **CLOSe**
    - | **OPEN**
    - | **READ?**
    - | **RESet**
    - | **WBINary**
    - | **WBLock**
    - | **WRITe**
- | **LAN:HOSTname**
- | **PSEnsor**
- | **TCPIP:CONTRol?**
- | **USB:PMETer:CAT?**

#### **CONFigure**

- | **REVision**
  - | **CPU?**
  - | **DSP?**
  - | **DSPFpga?**

#### **CONFigure:EDEVice More commands**

#### **CORRection**

- | **WIZard**

#### **ERRor?**

- | **COUNt?**
- | **REPort**
  - | **SUNLeveled**

#### **FIFO More commands**

#### **FPReset**

#### **MACRO:COPY**

- | **CHANnel**
- | **SOURce**

#### **MEASurement**

```
|CATalog?  
|NAME?  
|TRACe?  
|WINDow?  
POWer:  
|LIMit  
|LOCK  
|STATe  
PRESet  
PREferences More commands  
SECurity  
|[LEVel]  
SET?  
SHORtcut  
|ARGuments  
|DELeTe  
|EXECute  
|PATH  
|TITLe  
TOUCHscreen  
UPReset  
|FPANel[:STATe]  
|LOAD  
|SAVE[:STATe]  
WINDows  
|CATalog?
```

Click on a [blue](#) keyword to view the command details.

#### See Also

- [Learn about PNA Preferences](#)
- [Example Programs](#)
- [Synchronizing the PNA and Controller](#)
- [SCPI Command Tree](#)

---

## SYSTem:ACTive:CHANnel?

**(Read-only)** Returns the number of the active channel or an error message if there is no active channel. The active channel is the channel number that contains the active measurement.

**Examples**

```
SYST:PRES  
SYST:ACT:CHAN?
```

'Returns 1

**Return Type** Integer

**Default** Not Applicable

---

### SYSTem:ACTive:MEASurement?

**(Read-only)** Returns the name of the active measurement or an error message if there is no active measurement. While looking at the PNA display, the active measurement is the trace that has an indented [Trace Status button](#) and a label in the upper-left corner of the display. Only displayed measurements can be active.

**Examples**

```
SYST:PRES  
SYST:ACT:MEAS?
```

'Returns "CH1\_S11\_1"

**Return Type** String

**Default** Not Applicable

---

### SYSTem:CHANnels:CATalog?

**(Read-only)** Returns the channel numbers currently in use.

**Examples**

```
SYST:CHAN:CAT?  
  
system:channels:catalog?
```

'Returns:

"1, 2, 3"

**Return Type** String of comma-separated numbers

**Default** Not Applicable

---

### SYSTem:CHANnels:COUPle <bool>



**(Read-Write)** Sets and reads the state of channel coupling. This causes the PNA to emulate Agilent 8720 channel coupling.

When set to ON, all existing S-parameter channels receive the stimulus settings of the active channel. Subsequent changes made to any coupled channel are changed on all coupled channels.

Channels with applications such as SMC, VMC, GCA, Noise, IMD are not affected.

Coupling is primarily aimed at stimulus settings (such as start, stop, points, power) but also applies to many trigger settings and to Cal Set pointers.

#### Parameters

<bool> **ON** (or 1) Channels are coupled  
**OFF** (or 0) Channels are NOT coupled

#### Examples

```
SYST:CHAN:COUP 1  
system:channels:coupled OFF
```

**Query Syntax** SYSTem:CHANnels:COUPle?

**Default** OFF

---

### SYSTem:CHANnels:HOLD

**(Write-only)** Places all channels in hold mode. To place a single channel in hold mode, use [SENS:SWE:MODE](#).

#### Examples

```
SYST:CHAN:HOLD
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### SYSTem:CHANnels:RESume

**(Write-only)** Resumes the trigger mode of all channels that was in effect before sending [SYSTem:CHANnels:HOLD](#) (must be sent before SYST:CHAN:RESume).

#### Examples

```
SYST:CHAN:RES
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:COMMunicate:GPIB:PMETer:ADDRes <num>** **Superseded**

**Note:** This command is replaced with [SYST:COMM:PSENsor](#)

**(Read-Write)** Specifies the GPIB address of the power meter to be used in a source power calibration. When performing a source power cal, the PNA will search VISA interfaces that are configured in the Agilent IO Libraries on the PNA.

Use

**Parameters**

<num> GPIB address of the power meter. Choose any integer between 0 and 30.

**Examples**

```
SYST:COMM:GPIB:PMET:ADDR 13
```

```
system:communicate:gpiib:pmeter:address 14
```

**Query Syntax** SYSTem:COMMunicate:GPIB:PMETer:ADDRess?

**Return Type** Numeric

**Default** 13

---

**SYSTem:COMMunicate:GPIB:RDEvice:CLOSE <ID>**

**(Write only)** Closes the remote GPIB session. This command should be sent when ending every successful OPEN session.

**Parameters**

<ID> Session identification number that was returned with the [OPEN?](#) command.

**Examples**

[See an example program](#)

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:COMMunicate:GPIB:RDEvice:OPEN <bus>, <addr>, <timeout>**

**(Read-Write)** Initiates a GPIB pass-through session. First send this OPEN command, then send the OPEN query to read the session ID number. An existing GPIB pass-through session remains open after an instrument preset.

To learn more about GPIB pass-through capability, see the [example program](#).

**Parameters**

<bus> Bus ID number.

You can find the USB-GPIB adapter bus number by looking at the dialog that appears when the USB-GPIB device is connected. Error 1073 indicates the bus or address number is incorrect.

Use 0 (zero) when connected using a GPIB cable to the PNA controller port.

<addr> GPIB Address of the device to be controlled

<timeout> The amount of time (in milliseconds) to wait for a response from the remote device after sending a command. A "timeout" error is displayed after this time has passed without a response.

**Examples** [See an example program](#)

**Query Syntax** SYSTem:COMMunicate:GPIB:RDEvice:OPEN?

Returns the session identification number that is used when communicating with this device.

**Return Type** Numeric

**Default** Not Applicable

---

**SYSTem:COMMunicate:GPIB:RDEvice:READ? <ID>**

**(Read-only)** Returns data from the GPIB pass-through device.

**Parameters**

<ID> Session identification number that was returned with the [OPEN?](#) command.

**Examples** [See an example program](#)

**Return Type** String

**Default** Not Applicable

---

**SYSTem:COMMunicate:GPIB:RDEvice:RESet**

**(Write-only)** Performs the same function as [SYST:COMM:PIB:RDEV:CLOS](#) except that ALL pass-through sessions are closed.

**Examples** `SYST:COMM:PIB:RDEV:RES`

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## **SYSTem:COMMunicate:PIB:RDEvice:WBINary <ID>,<data>**

**(Write-only)** Sends data to a GPIB pass-through device. This command requires a header that specifies the size of the data to be written. The header (described below) is not passed along to the device.

Use this command if too many embedded quotes prevent you from using [SYST:COMM:PIB:RDEV:WRIT](#).

Use [SYST:COMM:PIB:RDEV:OPEN](#) to open the pass through session.

### **Parameters**

<ID> Session identification number that was returned with the [OPEN?](#) command.

<data> Data to be sent to the GPIB pass-through device. Use the following syntax:

```
#<num digits><byte count><data bytes><NL><END>
```

<num\_digits> specifies how many digits are contained in <byte\_count>

<byte\_count> specifies how many data bytes will follow in <data bytes>

**Examples** `SYSTem:COMMunicate:PIB:RDEvice:WBINary 101,#17ABC+XYZ<nl><end>`

# - always sent before data.

1 - specifies that the byte count is one digit (7).

7 - specifies the number of data bytes that will follow, not counting <NL><END>.

ABC+XYZ - Data block

<nl><end> - always sent at the end of block data.

The following example sends a line feed at the end.

```
SYST:COMM:GPIB:RDEV:WBIN 1, #210SYST:PRES<EOL>
```

The <EOL> represents your linefeed character.

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### **SYSTem:COMMunicate:GPIB:RDEvice:WBLock <ID>,<data>**

(Write-only) Same as [SYSTem:COMM:GPIB:RDEV:WBIN](#) (above) but the header **IS** passed along to the device.

Use this command if too many embedded quotes prevent you from using [SYST:COMM:GPIB:RDEV:WRIT](#).

#### **Parameters**

<ID> Session identification number that was returned with the [OPEN?](#) command.

<data> Data to be sent to the GPIB pass-through device. [See previous command.](#)

**Examples** See previous example.

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### **SYSTem:COMMunicate:GPIB:RDEvice:WRITe <ID>,<string>**

(Write-only) Sends ASCII string data to the GPIB pass-through device.

A line feed is NOT appended to the string data. To send a line feed, see the example in [SYST:COMM:GPIB:RDEV:WBIN](#).

#### **Parameters**

<ID> Session identification number that was returned with the [OPEN?](#) command.

<string> Commands to be sent to the GPIB pass-through device.

**Examples** [See an example program](#)

**Query Syntax** Not Applicable

**Default** Not Applicable

---

### **SYSTem:COMMunicate:LAN:HOSTname?**

**(Read-only)** Returns the LAN hostname that is visible in the Help, About Network Analyzer dialog box. [Learn more](#). This is the same information that is visible on the [LXI compliance dialog](#).

**Parameters** None

**Example** `SYST:CONF:LAN:HOSTname?`

**Return Type** String

**Default** Not applicable

---

## **SYSTem:COMMunicate:PSENSor <char>, <string>**

This command replaces [SYST:COMM:PIB:PMET:ADDR](#).

**(Read-Write)** Specifies the type and location of the power meter to be used in a source power calibration.

### **Parameters**

<char> Type of power meter/ sensor. Choose from:

- **GPIB** GPIB power meter
- **USB** USB power sensor or USB power sensor
- **LAN** LAN enabled power meter

<string> For **GPIB**, address of the power meter. Choose any integer between 0 and 30.

For **USB**, the ID string of the power meter or power sensor. Use [SYST:COMM:USB:PMET:CAT?](#) to see a list of ID strings of connected power meters and sensors.

For **LAN**, the hostname or IP address of the power meter.

### **Examples**

```
SYST:COMM:PSEN gpib, "14"  
  
system:communicate:psensor usb, "Agilent  
Technologies,U2000A,MY12345678"  
  
syst:comm:psen lan, "mymeter.agilent.com"
```

**Query Syntax** SYSTem:COMMunicate:PSENSor?

**Return Type** Character / String

**Default** GPIB

---

## SYSTem:COMMunicate:USB:PMETer:CATalog?

(Read-only) Returns the ID string of power meters / sensors that are connected to the PNA USB. Use the list to select a power sensor for a [source power cal.](#)

### Parameters

### Examples

```
SYST:COMM:USB:PMET:CAT?
```

```
system:communicate:usb:pmet:catalog?
```

**Return Type** Comma-delimited strings. Two power sensor strings are separate by a semicolon.

**Default** Not applicable

---

## SYSTem:COMMunicate:TCPip:CONTRol?

(Read-only) Queries the TCP/IP port number to use for opening a TCP/IP socket control connection to the PNA. The control connection is used for two purposes:

1. To perform a Device Clear operation on the PNA
2. To detect when a Service Request (SRQ) event occurs on the PNA.

The port number can range from 5000 to 5099. The PNA will skip over 5025 as it is being used for the primary socket connection.

To detect an SRQ, your program sends the appropriate commands via the regular socket connection to set up for a SRQ event to occur – the same sequence of commands as if you were sending them via GPIB. You write your program so that while your program is doing SCPI transactions on the standard socket connection, a second thread of execution in your program detects the SRQ on the control connection and responds to the event. When the SRQ event occurs, the PNA sends a “SRQ +xxx/n” message on the control connection (where ‘/n’ is linefeed character, ASCII value 10 decimal). The xxx value in the “SRQ +xxx/n” string is the IEEE 488.2 status byte at the time the SRQ was generated. So ‘listening’ for that on the control connection is how your program detects the event. If for your socket communication you’re using a software API that provides for asynchronous communication via a callback mechanism (for example, if you’re using Microsoft’s ‘winsock’ API, or their .NET ‘Socket’ class as in the example program below), in that case your ‘listener’ execution thread is created implicitly for you so your program doesn’t have to create one explicitly.

**Note:** If this SCPI query is sent to the PNA via a SCPI parser other than a TCP/IP socket connection (for example, if sent via GPIB), the query is not applicable in that case and will return value of 0.

**Parameters** None

### Example

[See example program](#)

---

<b>Return Type</b>	Integer
<b><u>Default</u></b>	Not applicable

---

## SYSTem:CONFigure <model>,<address>

**(Write-only)** Restarts as an "N-port" PNA using the specified multiport test set.

[Learn more about PNA Multiport capability.](#)

[See other commands to configure multiport test sets.](#)

### Parameters

- <model> String - Model of the test set with which to restart.
- Use "Native" to restart without a test set.
- To see a list of supported test sets, use [SENS:MULT:CAT?](#)
- <address> Numeric - GPIB Address of the test set. Ignored when model = "Native".

### Examples

```
SYST:CONF "NATIVE", 0
system:configure "N44xx", 18
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SYSTem:CONFigure:REVision:CPU?

**(Read-only)** Returns a number that corresponds to the PNA CPU speed that is visible in the Help, About Network Analyzer dialog box. [Learn more.](#)

Use the following table to learn the clock speed using the returned value.

Reported CPU version - Clock speed

**1.0** - 266 MHz

**2.0** - 500 MHz

**3.0** - 1100 MHz

**4.0** - 1600 MHz

**5.0** - 2000 MHz



**Parameters** None  
**Example**  
**Return Type** String  
**Default** Not applicable

---

### SYSTem:CONFigure:REVision:DSP?

(Read-only) Returns the DSP Revision number that is visible in the Help, About Network Analyzer dialog box. [Learn more.](#)

**Parameters** None  
**Example**  
**Return Type** String  
**Default** Not applicable

---

### SYSTem:CONFigure:REVision:DSPFpga?

(Read-only) Returns the DSP FPGA Revision number that is visible in the Help, About Network Analyzer dialog box. [Learn more.](#)

**Parameters** None  
**Example**  
**Return Type** String  
**Default** Not applicable

---

### SYSTem:CORRection:WIZard <char>

**(Write-only)** Launches either the Calibration Wizard or the Version 2 Calibration Kit File Manager dialog box.

Remote operation returns immediately after the dialog is launched. This is done to avoid timeout issues with I/O protocols such as VISA. Although it is possible to send commands to the PNA while the dialog is open, this is not encouraged. Application programs should wait until the dialog is closed before resuming remote operations.

#### Parameters

<char> Choose from:

**MAIN** - Launches the Calibration Wizard which matches the current channel, such as standard S-params, NoiseFigure, GCA, and so forth.

**CKIT** - Launches the Version 2 Calibration Kit File Manager dialog box.

Both display on the PNA screen.

#### Examples

```
SYST:CORR:WIZ MAIN
system:correction:wizard ckit
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SYSTem:ERRor?

**(Read-only)** Returns the next error in the error queue. Each time the analyzer detects an error, it places a message in the error queue. When the `SYSTEM:ERROR?` query is sent, one message is moved from the error queue to the output queue so it can be read by the controller. Error messages are delivered to the output queue in the order they were received. The error queue is cleared when any of the following conditions occur:

- When the analyzer is switched ON.
- When the `*CLS` command is sent to the analyzer.
- When all of the errors are read.

If the error queue overflows, the last error is replaced with a "Queue Overflow" error. The oldest errors remain in the queue and the most recent error is discarded.

[See list of all SCPI Errors.](#)

#### Examples

```
SYST:ERR?
system:error?
```

---

**Default** Not Applicable

---

## SYSTem:ERRor:COUNT?

(Read-only) Returns the number of errors in the error queue. Use SYST:ERR? to read an error.

[See list of all SCPI Errors.](#)

**Examples**

```
SYST:ERR:COUN?  
system:error:count?
```

**Default** Not Applicable

---

## SYSTem:ERRor:REPort:SUNLeveled <bool>

(Read-Write) Specifies whether or not to report [Source Unleveled](#) errors to the SCPI system error buffer.

This setting will revert to the default (OFF) setting on Instrument Preset.

### Parameters

<bool> **ON** (or 1) Report Source Unleveled Errors. Read errors from the system error buffer using [SYST:ERR?](#)

**OFF** (or 0) Do NOT report Source Unleveled Errors.

**Examples**

```
SYST:ERR:REP:SUNL 1  
system:error:report:sunleveled ON
```

**Query Syntax** SYSTem:ERRor:REPort:UNLeveled?

**Default** OFF

---

## SYSTem:FPReset

(Write-only) Performs a standard [Preset](#), then deletes the default trace, measurement, and window. The PNA screen becomes blank.

**Examples**

```
SYST:FPR  
system:fpreset
```

**Default** Not applicable

---

## SYSTem:MACRo:CoPY:CHANnel<cnum>[:TO] <num>

(Write-only) Sets up channel <num> as a copy of channel <cnum>. Learn more about [copy channels](#).

**Parameters**

- <cnum> Any existing channel number. If unspecified, value is set to 1.
- <num> Number of the channel which is to become a copy of channel <cnum>.

**Examples**

```
SYST:MACR:COPY:CHAN1 2  
system:macro:copy:channel2:to 3
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:MACRo:COPIY:CHANnel<fromChan>:SOURce <fromPort>,<toChan>,<toPort>**

(Write-only) Copies and applies an existing Source Power Calibration to another channel. Learn more about [source power calibration](#).

**Parameters**

- <fromChan> Channel number of the existing source power correction.
- <fromPort> Port number of the existing source power correction.
- <toChan> Channel number to which the source power correction will be copied.
- <toPort> Port number to which the source power correction will be applied.

**Examples**

```
SYST:MACR:COPIY:CHAN1:SOUR 1,2,1  
system:macro:copy:channel2:source 2,1,2
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:MEASurement:CATalog? [chan]**

**(Read-only)** Returns ALL measurement numbers, or measurement numbers from a specified channel.

**Parameters**

[chan] Optional. Channel number to catalog. If not specified, all measurement numbers are returned.

**Examples**

```
'Returns all measurement names  
SYST:MEAS:CAT?  
  
'Returns the measurement names on channel 2  
system:measurement:catalog? 2
```

**Return Type** String of comma-separated numbers

For example: "1,2"

**Default** Not Applicable

---

**SYSTem:MEASurement<n>:NAME?**

**(Read-only)** Returns the name of the specified measurement.

**Parameters**

<n> Measurement number for which to return the measurement name. If unspecified, value is set to 1.

**Examples**

```
'Returns the name of measurement 2  
SYST:MEAS2:NAME?
```

**Return Type** String

**Default** Not Applicable

---

**SYSTem:MEASurement<n>:TRACe?**

**(Read-only)** Returns the trace number of the specified measurement number. Trace numbers restart for each window while measurement numbers are always unique.

**Parameters**

<n> Measurement number for which to return the trace number. If unspecified, value is set to 1.

**Examples**

```
'Returns the trace number of measurement 1  
SYST:MEAS1:TRAC?
```

**Return Type** Numeric

**Default** Not Applicable

---

**SYSTem:MEASurement<n>:WINDow?**

**(Read-only)** Returns the window number for the specified measurement number.

**Parameters**

<n> Measurement number for which to return the window number. If unspecified, value is set to 1.

**Examples**

```
'Returns the window number of measurement 2  
SYST:MEAS2:WIND?
```

**Return Type** Numeric

**Default** Not Applicable

---

**SYSTem:POWer<pnum>:LIMit <value>**

**(Read-Write)** Sets and returns the power limit for the specified port. [Learn more about Power Limit.](#)

**Parameters**

<pnum> Port number. Choose any PNA port.

<value> Power limit in dBm

**Examples**

```
SYST:POW1:LIM 5  
system:power2:limit 0
```

**Query Syntax** SYSTem:POWer<pnum>:LIMit?

**Return Type** Numeric

**Default** 100 dBm

---

## SYSTem:POWer:LIMit:LOCK <bool>

(Read-Write) Enables or disables the ability to change the power limit values through the user interface. [Learn more about Power Limit.](#)

### Parameters

<bool> Power limit lock. Choose from:

**ON or 1** - Enables the ability to change the power limit values from the user interface.

**OFF or 0** - Disables the ability to change the power limit values from the user interface.

### Examples

```
SYST:POW:LIM:LOCK 1
```

```
system:power:limit:lock OFF
```

**Query Syntax** SYSTem:POWer:LIMit:LOCK?

**Return Type** Boolean

**Default** OFF

---

## SYSTem:POWer<pnum>:LIMit:STATe <bool>

(Read-Write) Enables or disables the power limit for the specified port. [Learn more about Power Limit.](#)

### Parameters

<pnum> Port number. Choose any PNA port.

<value> Power limit state. Choose from:

**ON or 1** Enables the power limit for the port<pnum>.

**OFF or 0** Disables the power limit for the port<pnum>.

### Examples

```
SYST:POW1:LIM:STAT ON
```

```
system:power2:limit:state 0
```

**Query Syntax** SYSTem:POWer<pnum>:LIMit:STATe?

**Return Type** Boolean

**Default** OFF

---

## SYSTem:PRESet

(Write-only) Deletes all traces, measurements, and windows. In addition, resets the analyzer to factory defined default settings and creates a S11 measurement named "CH1\_S11\_1". For a list of default settings, see [Preset](#).

Regardless of the state of the User Preset Enable checkbox, the SYST:PRESet command will always preset the PNA to the factory preset settings, and [SYST:UPReset](#) will always perform a User Preset.

If the PNA display is disabled with [DISP:ENAB OFF](#) then SYST:PRES will NOT enable the display.

This command performs the same function as [\\*RST](#) with one exception: Syst:Preset does NOT reset [Calc:FORMAT](#) to ASCII as does \*RST.

### Examples

```
SYST:PRES  
system:preset
```

[Default](#) Not applicable

---

## SYSTem:SECurity[:LEVel] <char>

(Read-Write) Sets and returns the display of frequency information on the PNA screen and printouts.

[Learn more about security level.](#)

### Parameters

<char> Choose from:

**NONE** - ALL frequency information is displayed.

**LOW** - NO frequency information is displayed. Frequency information can be redisplayed using the Security Setting dialog box or this command.

**HIGH** - LOW setting plus [GPIB console](#) is disabled. Frequency information can be redisplayed ONLY by performing a Preset, recalling an instrument state with None or Low security settings, or using this command.

**EXTRa** - HIGH setting plus [ASCII data saving](#) is disabled. Same method to redisplay frequency information as HIGH setting.

### Examples

```
SYST:SEC LOW  
system:security:level high
```

**Query Syntax** SYSTem:SECurity[:LEVel]?

**Return Type** Character

[Default](#) None

---



## SYSTem:SET <block>

(Read-Write) Sends a definite-length binary block Instrument state and sets the PNA with those settings. This command does the same as saving a \*.sta file to the PNA ([MMEM:STOR STATE](#)) and then [MMEM:TRAN](#) to transfer the file to the computer.

### Parameters

<block> The Instrument state file as definite-length arbitrary binary block.

### Examples

```
SYST:SET <block>
```

**Query Syntax** SYSTem:SET? (This saves the instrument state file to the remote computer.)

**Return Type** Definite-length arbitrary binary block.

**Default** Not Applicable

---

## SYSTem:SHORTcut<n>:ARGuments<string>

(Read-Write) Reads and writes the arguments for the specified macro. On the [Edit Macro Dialog](#), this is called the "Macro run string parameters".

### Parameters

<n> Numeric. Number of the macro that is stored in the PNA.

To find the number of a macro, either open the [Macro Setup dialog](#) and count the line number of the desired macro, or query the titles of all 12 macros for the desired macro title.

<string> Arguments for the specified macro.

### Examples

```
SYST:SHOR1:ARG  
"http://na.tm.agilent.com/pna/help/PNAWebHelp/help.htm"
```

**Query Syntax** SYSTem:SHORTcut<n>:ARGuments?

**Default** Not Applicable

---

## SYSTem:SHORTcut<n>:DELeTe

**(Write-only)** Removes the specified macro from the list of macros in the PNA. Does not delete the macro executable file.

**Parameters**

<n> Numeric. Number of the macro that is stored in the PNA.

To find the number of a macro, either open the [Macro Setup dialog](#) and count the line number of the desired macro, or query the titles of all 12 macros for the desired macro title.

**Examples**

```
SYST:SHOR1:DEL
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:SHORtcut<n>:EXECute**

**(Write-only)** Executes (runs) the specified Macro (shortcut) that is stored in the PNA.

**Parameters**

<n> Numeric. Number of the macro that is stored in the PNA.

To find the number of a macro, either open the [Macro Setup dialog](#) and count the line number of the desired macro, or query the titles of all 12 macros for the desired macro title.

**Examples**

```
SYST:SHOR1:EXEC
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:SHORtcut<n>:PATH <string>**

**(Read-Write)** Defines a Macro (shortcut) by linking a path and file name to the Macro number. To be executed, the executable file must be put in the PNA at the location indicated by this command.

**Parameters**

<n> Numeric. Number of the macro to be stored in the analyzer. If the index number already exists, the existing macro is replaced with the new macro.

<string> Full path, file name, and extension, of the existing macro "executable" file.

To find the number of a macro, either open the [Macro Setup dialog](#) and count the line number of the desired macro, or query the titles of all 12 macros for the desired macro title.

**Examples**

```
SYST:SHOR1:PATH "C:/Program Files/Agilent/Network Analyzer/Documents/unguideMultiple.vbs"
```

**Query Syntax** SYSTem:SHORTcut<n>:PATH?

**Default** Not Applicable

---

**SYSTem:SHORTcut<n>:TITLE<string>**

**(Read-Write)** Reads and writes the name of the specified macro.

**Parameters**

<n> Numeric. Number of the macro that is stored in the PNA.

To find the number of a macro, either open the [Macro Setup dialog](#) and count the line number of the desired macro, or query the titles of all 12 macros for the desired macro title.

<string> The name to be assigned to the macro.

**Examples**

```
SYST:SHOR1:TITL "Guided 4-Port Cal"
```

**Query Syntax** SYSTem:SHORTcut<n>:TITLE?

**Default** Not Applicable

---

**SYSTem:TOUCHscreen[:STATE] <bool>**

**(Read-Write)** Enables and disables the **PNA-X** touchscreen.

This setting remains until changed again from the front-panel or remotely, or until the hard drive is changed or reformatted.

**Parameters**

<bool> Choose from:

**ON (1)** Enables the touchscreen.

**OFF (0)** Disables the touchscreen.

**Examples**

```
SYST:TOUC 1
system:touchscreen:state OFF
```

**Query Syntax** SYSTem:TOUCHscreen[:STATe]?

**Return Type** Boolean

**Default** ON when shipped from factory.

---

## **SYSTem:UPReset**

**(Write-only)** Performs a User Preset. There must be an active User Preset state file (see [Load](#) and [Save](#)) or an error will be returned. [Learn more about User Preset.](#)

Regardless of the state of the User Preset Enable checkbox, the SYST:PRESet command will always preset the PNA to the factory preset settings, and [SYST:UPReset](#) will always perform a User Preset.

**Examples**

```
SYST:UPReset
system:upreset
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

**SYSTem:UPReset:FPANel[:STATe] <bool>**

(Read-Write) 'Checks' and 'clears' the enable box on the [User Preset dialog box](#). This only affects subsequent Presets from the front panel user interface.

Regardless of the state of the User Preset Enable checkbox, the [SYST:PRESet](#) command will always preset the PNA to the factory preset settings, and [SYST:UPReset](#) will always perform a User Preset.

#### Parameters

<bool> Front Panel User Preset State. Choose from:

0 User Preset OFF

1 User Preset ON

#### Examples

```
SYST:UPR:FPAN 1
```

```
system:upreset:fpanel:state 0
```

Query Syntax SYSTem:UPREset:FPANel[:STATe]?

Return Type Boolean

Default 0

---

### SYSTem:UPReset:LOAD <file>

(Read-Write) Loads an existing instrument state file (.sta or .cst) to be used for User Preset. Subsequent execution of [SYSTem:UPReset](#) will cause the PNA to assume this instrument state.

Regardless of the state of the User Preset Enable checkbox, the [SYST:PRESet](#) command will always preset the PNA to the factory preset settings, and [SYST:UPReset](#) will always perform a User Preset.

[Learn more about User Preset.](#)

#### Parameters

<file> String - Name of the file to be loaded. The default folder "C:/Program Files/Agilent/Network Analyzer/Documents" is used if unspecified. Change the default folder name using [MMEMory:CDIRectory](#).

#### Examples

```
SYST:UPR:LOAD '1MHzto20GHzUserPreset.cst'
```

```
system:upreset:load 'C:/Documents and Settings/Administrator/My Documents/NewUserPreset.cst'
```

Query Syntax Not Applicable

Default Not Applicable

## SYSTem:UPReset:SAVE[:STATe]

(Write-only) Saves the current instrument settings as UserPreset.sta. Subsequent execution of [SYSTem:UPReset](#) will cause the PNA to assume this instrument state.

Regardless of the state of the User Preset Enable checkbox, the [SYST:PRESet](#) command will always preset the PNA to the factory preset settings, and [SYST:UPReset](#) will always perform a User Preset.

[Learn more about User Preset.](#)

### Examples

```
SYST:UPR:SAVE
```

```
system:upreset:save:state
```

**Query Syntax** Not Applicable

**Default** Not Applicable

---

## SYSTem:WINDows:CATalog?

(Read-only) Returns the numbers of window that are currently being used.

### Examples

```
SYST:WIND:CAT?
```

```
system:windows:catalog?
```

**Return Type** String of comma-separated numbers.

For example: "1,2"

**Default** Not Applicable

---

Last modified:

9-Apr-2010	Added Preset note to 'unleveled' command
30-Nov-2009	Added Help About read commands(9.1)
30-Jul-2009	Added syst:conf:edev and RTOF (9.0)
24-Feb-2009	Added Chan:Coup; Replaced True/False
4-Nov-2008	Added FIFO (8.33)
30-Oct-2008	Added several meas/trace/window query commands (8.33.x)
29-Sep-2008	Removed rev from Psensor example
17-Sep-2008	Added Syst:Pres vs *RST note
28-Aug-2008	Updated Launch Cal Wiz command
11-Feb-2008	Added Noise switch preference (8.2)
5-Feb-2007	Added Extra security and USB power meter commands
23-Feb-2007	Added touchscreen command

15-Nov-2006 Added Unleveled Error reporting  
31-Oct-2006 Added PSRTrace command

## Trigger Commands

---

Controls the starts or ends of data acquisition.

### TRIGger:

AUX

| COUNT

CHANnel:AUX

| DELay

| DURATION

| ENABLE

| HANDshake

| INTERVAL

| I POLarity

| O POLarity

| POSition

| TYPE

CHANnel:ROUTe

| INPut

| READy

DELay

PREFerence

| AIGLobal

READy:POLarity

[SEQUence]

| LEVel

| SCOPe

| SOURce

SLOPe



## TYPE

Click on a [blue](#) keyword to view the command details.

Red commands are superseded.

### See Also

- **Example program** [Triggering the PNA](#)
  - [See other SCPI Triggering commands](#)
  - [Learn about External / Aux Triggering](#)
  - [Synchronizing the PNA and Controller](#)
  - [SCPI Command Tree](#)
- 

## TRIGger:AUX:COUNt?

**(Read-only)** Returns the number of AUX trigger input / output connector pairs in the instrument.

- PNA-X = 2
- E836xB and PNA-L models = 1

### Parameters

#### Examples

```
TRIG:CHAN:AUX:COUN?
```

```
trigger:channel2:aux:count?
```

**Return Type** Numeric

**Default** Not Applicable

---

## TRIGger:CHANnel<ch>:AUXiliary<n>:DELay <num>

**(Read-Write)** Specifies the delay that should be applied by the PNA after the Aux trigger input is received and before the acquisition is made.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connectors used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1 IN](#)) or **2** ([AUX TRIG 2 IN](#))
- All other models: choose **1**.

If unspecified, value is set to 1.

<num> Delay value in seconds. Choose a value between 0 and 3.0 seconds.

**Examples**

```
TRIG:CHAN:AUX:DEL .5  
trigger:channel2:aux2:delay 1.5
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:DELay?

**Return Type** Numeric

**Default** 0

---

**TRIGger:CHANnel<ch>:AUXiliary<n>:DURation <num>**

**(Read-Write)** Specifies the width of the output pulse, which is the time that the Aux trigger output will be asserted.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<num> Duration value in seconds. Choose a value between 1us (1E-6) and 1

**Examples**

```
TRIG:CHAN:AUX:DUR .1  
trigger:channel2:aux2:duration .01
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:DURation?

**Return Type** Numeric

**Default** 1E-6

---

### **TRIGger:CHANnel<ch>:AUXiliary<n>[:ENABLE] <bool>**

**(Read-Write)** Turns ON / OFF the trigger output.

#### **Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<bool> **ON** (or 1) - turns trigger output ON.

**OFF** (or 0) - turns trigger output OFF.

#### **Examples**

```
TRIG:CHAN:AUX 1
```

```
trigger:channel2:aux2:enable off
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>[:ENABLE]?

**Return Type** Boolean

**Default** OFF

---

### **TRIGger:CHANnel<ch>:AUXiliary<n>:HANDshake <bool>**

**(Read-Write)** Turns handshake ON / OFF.

To enable handshake, the main trigger enable must also be set using [TRIG:CHAN:AUX:ENAB](#).

When ON, PNA waits indefinitely for the input line to be asserted before continuing with the acquisition. When OFF, the PNA acquires data without waiting.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<bool> **ON** (or 1) - turns handshaking ON.

**OFF** (or 0) - turns handshaking OFF.

#### Examples

```
TRIG:CHAN:AUX:HAND 1
```

```
trigger:channel2:aux2:handshake off
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:HANDshake?

**Return Type** Boolean

**Default** OFF

---

**TRIGger:CHANnel<ch>:AUXiliary<n>:INTerval <char>**

**(Read-Write)** Specifies how often a trigger output signal is sent.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<char> Choose from:

- **POINT** Trigger signal is sent every data point. (effectively the same as [Point sweep](#))
- **SWEep** Trigger signal is sent once every sweep.

#### Examples

```
TRIG:CHAN:AUX:INT POI
```

```
trigger:channel2:aux2:interval sweep
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:INTerval?

**Return Type** Character

**Default** SWEep

---

### TRIGger:CHANnel<ch>:AUXiliary<n>:IPOLarity <char>

**(Read-Write)** Specifies the polarity of the trigger IN signal to which the PNA will respond.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<char> Choose from:

- **POSitive** PNA responds to leading edge or HIGH level

- **NEGative** PNA responds to trailing edge or LOW level.

Set Edge or Level triggering using [TRIG:CHAN:AUX:TYPE](#)

#### Examples

```
TRIG:CHAN:AUX:IPOL POS
```

```
trigger:channel2:aux2:ipolarity negative
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:IPOLarity?

**Return Type** Character

**Default** NEGative

### TRIGger:CHANnel<ch>:AUXiliary<n>:OPOLarity <char>

(Read-Write) Specifies the polarity of the Aux Output signal being supplied by the PNA.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<char> Choose from:

- **POSitive** PNA sends positive going pulse.
- **NEGative** PNA sends negative going pulse.

#### Examples

```
TRIG:CHAN:AUX:OPOL NEG
```

```
trigger:channel2:aux2:opolarity positive
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:OPOLarity?

**Return Type** Character

**Default** NEGative

### TRIGger:CHANnel<ch>:AUXiliary<n>:POSition <char>

**(Read-Write)** Specifies whether the aux trigger out signal is sent BEFore or AFTer the acquisition.

### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<char> Choose from:

- **BEFore** Use if the external device needs to be triggered before the data is acquired, such as a power meter.
- **AFTer** Use if the external device needs to be triggered just after data has been acquired, such as an external source. This could be more efficient since it allows the external device to get ready for the next acquisition at the same time as the PNA.

### Examples

```
TRIG:CHAN:AUX:POS BEF
```

```
trigger:channel2:aux2:position after
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:POSition?

**Return Type** Character

**Default** AFTer

---

**TRIGger:CHANnel<ch>:AUXiliary<n>:TYPE <char>**

**(Read-Write)** Specifies the type of Aux input detection that the PNA will employ.

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<n> Rear panel connector used to send or receive signals.

- PNA-X - choose from **1** ([AUX TRIG 1](#)) or **2** ([AUX TRIG 2](#))
- All other models: choose **1**. ([BNC Trig I/O](#))

If unspecified, value is set to 1.

<char> Choose from:

**EDGE** PNA responds to the leading edge of a signal

**LEVel** PNA responds to the level (HIGH or LOW) of a signal

**Examples**

```
TRIG:CHAN:AUX:TYPE EDGE
```

```
trigger:channel2:aux2:type level
```

**Query Syntax** TRIGger:CHANnel<ch>:AUXiliary<n>:TYPE?

**Return Type** Character

**Default** EDGE

---

**TRIGger:CHANnel<ch>:ROUTE:INPut <char>**

**(Read-Write)** Specifies the connector to use for the external trigger input.(Similar to [CONTROL:SIGNal <conn>](#))

**Parameters**

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Choose from:

**MAIN** - meas trig ready

**MATH** - material handler

**AUXT** - auxio (PNA-L and E836xB ONLY)

**Examples**

```
TRIG:CHAN:ROUT:INP MAIN
```

```
trigger:channel2:route:input main
```

**Query Syntax** TRIGger:CHANnel<ch>:ROUTE:INPut?



**Return Type** Character

**Default** PNA model dependent

---

### TRIGger:CHANnel<ch>:ROUTe:READy <char>

(Read-Write) Specifies the connector to use for the external trigger OUT ready line.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Choose from:

MAIN - [Meas trig ready](#)

MATH - [Material handler pin 21](#)

AUXT - [AUX I/O pin 18](#) (PNA-L and E836xB ONLY)

#### Examples

```
TRIG:CHAN:ROUT:READ main
trigger:channel2:route:ready auxt
```

**Query Syntax** TRIGger:CHANnel<ch>:ROUTe:READy?

**Return Type** Character

**Default** PNA model dependent

---

### TRIGger:DELay <num>

(Read-Write) Sets and reads the trigger delay for ALL channels (globally). This delay is only applied while [TRIG:SOURce](#) = EXTERNAL and [TRIG:SCOP](#) = ALL. After an external trigger is applied, the start of the sweep is held off for an amount of time equal to the delay setting plus any inherent latency.

To apply a trigger delay for the specified channel ONLY, use [SENS:SWE:TRIG:DELay](#)

#### Parameters

<num> Delay value in seconds. Choose from 0 to 107.

#### Examples

```
TRIG:DEL .0003
```

Sets the trigger delay to 300 microseconds. The sweep will not start until approximately 300 microseconds after an external trigger is applied.

**Query Syntax** TRIGger:DELay?

**Return Type** Numeric

## TRIGger:PREFerence:AIGLobal <bool>

(Read-Write) Sets the Trigger OUT behavior to either Global or Channel. [Learn more about this setting.](#)

This command will cause the PNA to Preset.

This setting remains until changed again using this command, or until the hard drive is changed or reformatted.

To send this command using the PNA front panel, open the [GPIB Command Processor Console](#), then type either of the following examples at the command prompt. Then type the Query Syntax and press enter to be sure the PNA took the command.

### Parameters

<bool> Choose from:

- **ON** (or 1) - Trigger properties apply to ALL channels (Global).
  - Allows use of [CONT:SIGNAL](#) command to configure the external trigger properties.
  - "Per Point" trigger property is not settable. Use the channel's [Point trigger](#) setting.
  - Default setting for E836xB and PNA-L models.
- **OFF** (or 0) - External Trigger properties apply to each channel independently.
  - Must use [TRIG:CHAN:AUX](#) commands to configure the external trigger properties. [CONT:SIGNAL](#) will NOT work.
  - "Per Point" trigger output property is set using the channel's [Point trigger](#) setting **AND** [TRIG:CHAN:AUX:INTERval](#).
  - Default setting for PNA-X models.

### Examples

```
TRIG:PREF:AIGL 1  
trigger:preference:aiglobal 0
```

**Query Syntax** TRIGger:PREFerence:AIGLobal?

**Return Type** Boolean

**Default** 1 - E836xB and PNA-L models

0 - PNA-X models

---

### TRIGger:READy:POLarity <char>

(Read-Write) Specifies the polarity of Ready for Trigger output.

All existing Ready for Trigger outputs for PNA-X and PNA-L models are configured simultaneously with this command. [See Capabilities Summary](#).

The Ready for Trigger polarity can NOT be configured for E836x models.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> **LOW** - Outputs a TTL low when the PNA is ready for trigger.

**HIGH** - Outputs a TTL high when the PNA is ready for trigger.

#### Examples

```
TRIG:READ:POL HIGH
```

```
trigger:ready:polarity low
```

**Query Syntax** TRIGger:READy:POLarity?

**Return Type** Character

**Default** Low

---

### TRIGger[:SEQuence]:LEVel <char> - Superseded

This command is replaced with [CONTrol:SIGNal](#)

(Read-Write) Triggers either on a **High or Low** level trigger signal. This setting only has an effect when [TRIG:SOURce EXTernal](#) is selected.

#### Parameters

<char> Choose from:

- **HIGH** - analyzer triggers on TTL **High**
- **LOW** - analyzer triggers on TTL **Low**

#### Examples

```
TRIG:LEV HIGH
```

```
trigger:sequence:level low
```

**Query Syntax** TRIGger[:SEQuence]:LEVel?

**Return Type** Character

**Default** LOW

---

### TRIGger[:SEquence]:SCOPE <char>

(Read-Write) Specifies whether a trigger signal is sent to all channels or only the current channel.

See [Triggering the PNA using SCPI](#).

#### Parameters

<char> Choose from:

- **ALL** - trigger signal is sent to all channels. Also sets [SENS:SWEp:TRIG:POINT OFF](#) on **ALL** channels.
- **CURRent** - trigger signal is sent to only one channel at a time. With each trigger signal, the channel is incremented to the next triggerable channel.

#### Examples

```
TRIG:SCOP ALL
trigger:sequence:slope current
```

**Query Syntax** TRIGger[:SEquence]:SCOPE?

**Return Type** Character

**Default** ALL

---

### TRIGger[:SEquence]:SLOPE <char>

(Read-Write) Specifies the polarity expected by the external trigger input circuitry. Also specify [TRIG:TYPE](#) (Level |Edge).

See [Triggering the PNA using SCPI](#).

#### Parameters

<char> Choose from:

- **POSitive** (rising Edge) or High Level
- **NEGative** (falling Edge) or Low Level

#### Examples

```
TRIG:SLOP NEG
trigger:sequence:slope positive
```

**Query Syntax** TRIGger[:SEquence]:SLOPE?

**Return Type** Character

**Default** POSitive

---

### TRIGger[:SEQuence]:SOURce <char>

(Read-Write) Sets the source of the sweep trigger signal. This command is a super-set of [INITiate:CONTinuous](#) which can NOT set the source to External. To configure external triggering, see [CONTrol:SIGNal](#).

See [Triggering the PNA using SCPI](#).

#### Parameters

<char> Choose from:

- **EXTernal** - external (rear panel) source.
- **IMMediate** - internal source sends continuous trigger signals
- **MANual** - sends one trigger signal when manually triggered from the front panel or [INIT:IMM](#) is sent.

#### Examples

```
TRIG:SOUR EXT
trigger:sequence:source immediate
```

**Query Syntax** TRIGger[:SEQuence]:SOURce?

**Return Type** Character

**Default** IMMediate

---

### TRIGger:TYPE <char>

(Read-Write) Specifies the type of EXTERNAL trigger input detection used to listen for signals on the Ext I/O Trig and Meas Trig IN connectors. Edge triggers are most commonly used.

#### Parameters

<ch> Any existing channel number. If unspecified, value is set to 1.

<char> Choose from:

**EDGE** PNA responds to the rising and falling edge of a signal.

**LEVel** PNA responds to a level (HIGH or LOW).

Use [TRIG:SLOPe](#) to specify Rising or falling - High or Low.

**Examples**`TRIG:TYPE EDGE``trigger:type level`**Query Syntax**`TRIGger:TYPE?`**Return Type**

Character

**Default**EDGE

---

## Last modified:

28-Sep-2009	Fixed CHAN:AUXiliary commands
11-Feb-2009	Added TRIG:SLOPe
14-Mar-2008	Added READy:POL command
22-Feb-2008	Clarified AIGL command
24-Apr-2007	Clarified trigger source and scope
15-Feb-2007	MX Updated for AUX triggering

## Catalog Measurements using SCPI

---

This Visual Basic Program does the following:

- Catalogs the currently defined measurements, windows, and traces
- Selects a measurement for further definition
- Adds a Title to the window

To run this program, you need:

- An established [GPIB interface connection](#)

---

[See Other SCPI Example Programs](#)

---

```
Dim Meas as String
Dim Win as String
Dim Trace as String

'Read the current measurements in Channel 1
GPIB.Write "<a href='\"#\"'>CALCulate1:PARAmeter:CATalog?</a>"
Meas = GPIB.Read
MsgBox ("Ch1 Measurments: " & Meas)

'Read the current windows
GPIB.Write "<a href='\"#\"'>DISPlay:CATalog?</a>"
Win = GPIB.Read
MsgBox ("Windows: " & Win)

'Read current traces in window 1
GPIB.Write "<a href='\"#\"'>DISPlay:WINDow1:CATalog?</a>"
Trace = GPIB.Read
MsgBox ("Traces in Window1: " & Win)
```

---

## Create an FOM Measurement

---

All three VBScript examples in this topic create a FOM measurement with the following attributes:

- Sweep the Source (input) from 1 GHz to 2 GHz
- Sweep the Receivers (output) from 2 GHz to 3 GHz
- You provide an LO at 1 GHz

### [Learn more about Frequency Offset Mode](#)

These programs can be run as a macro in the PNA. To do this, copy the code into a text editor file such as Notepad and save on the PNA hard drive as FOM.vbs. [Learn how to setup and run the macro.](#)

### [See Other SCPI Example Programs](#)

The following example will run on any PNA model with FOM (opt 080). However, these commands have no provisions for internal second source. It uses [Sens:Offset commands](#) introduced before 'enhanced FOM' was released for the A.07.10 release.

```
' This section gets the PNA application
' starts the scpi parser, and presets the PNA
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
scpi.Execute("SYST:FPRESET")
' Create and turn on window 1
scpi.Execute ("DISPlay:WINDow1:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate:PARAmeter:DEFine 'MyMeas',S21")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (1)
'and give the new TRACe a number (1).
scpi.Execute ("DISPlay:WINDow1:TRACe1:FEED 'MyMeas'")

scpi.Execute ("SENS:FREQ:START 1e9")
scpi.Execute ("SENS:FREQ:STOP 2e9")
'set the receivers to be 2e9 -> 3e9
scpi.Execute ("SENS:OFFS:OFFS 1e9")
scpi.Execute ("SENS:OFFS ON")
```



The following example can be run ONLY on a PNA with revision A.07.10 or later and has FOM (opt 080). It uses new [Sens:FOM commands](#).

```
' This section gets the PNA application
' starts the scpi parser, and presets the PNA
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
scpi.Execute("SYST:FPRESET")
' Create and turn on window 1
scpi.Execute ("DISPlay:WINDow1:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate:PARAmeter:DEFine 'MyMeas',S21")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (1), and give the
new TRACe a number (1) .
scpi.Execute ("DISPlay:WINDow1:TRACe1:FEED 'MyMeas'")

scpi.Execute("SENS:FREQ:START 1e9")
scpi.Execute("SENS:FREQ:STOP 2e9")
'set the receivers to be 2e9 -> 3e9
scpi.Execute("SENS:FOM:RANG3:FREQ:OFFS 1e9")
scpi.Execute("SENS:OFFS ON")
```

The following example can be run ONLY on a PNA with a second internal source, has revision A.07.10 or later, and has FOM (opt 080). It uses the internal 2nd source for the fixed LO frequency.

```
' This section gets the PNA application
' starts the scpi parser, and presets the PNA
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
scpi.Execute("SYST:FPRESET")
' Create and turn on window 1
scpi.Execute ("DISPlay:WINDow1:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate:PARAmeter:DEFine 'MyMeas',S21")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (1)
'and give the new TRACe a number (1) .
scpi.Execute ("DISPlay:WINDow1:TRACe1:FEED 'MyMeas'")
```

```
scpi.Execute ("SENS:FREQ:START 1e9")
scpi.Execute ("SENS:FREQ:STOP 2e9")
'set the receivers to be 2e9 -> 3e9
scpi.Execute ("SENS:FOM:RANG3:FREQ:OFFS 1e9")
'setup the 2nd source frequencies
scpi.Execute ("SENS:FOM:RANG4:COUP 0")
scpi.Execute ("SENS:FOM:RANG4:FREQ:START 1e9")
scpi.Execute ("SENS:FOM:RANG4:FREQ:STOP 1e9")
'turn off coupling
scpi.Execute ("SOUR:POW:COUP 0")
'set LO power to 10dBm
scpi.Execute ("SOUR:POW3 10")
'turn ON port 3, our LO signal
scpi.Execute ("SOUR:POW3:MODE ON")
scpi.Execute ("SENS:FOM:STAT ON")
```

---

Last Modified:

9-Oct-2007    MX New topic

## Create a Measurement using SCPI

---

This VBScript program creates a new S21 measurement and displays it on the PNA screen.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as NewMeas.vbs. [Learn how to setup and run the macro.](#)

---

### [See Other SCPI Example Programs](#)

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpIStringParser

' A comment
'Preset the analyzer
scpi.Execute ("SYST:FPRreset")
' Create and turn on window 1
scpi.Execute ("DISPlay:WINDow1:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate:PARameter:DEFine 'MyMeas',S21")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (1), and give the new
TRACe a number (1).
scpi.Execute ("DISPlay:WINDow1:TRACe1:FEED 'MyMeas'")
```

## Create a Balanced Measurement using SCPI

---

This example program does the following:

- creates several Balanced measurements in separate windows
- generates markers
- calculates statistics
- sets limit lines and queries results
- queries a measurement to determine if we have a balanced parameter and what type it is.

**Note:** By their nature, balanced measurements are extremely sensitive to phase differences between the two RF paths that make up the balanced port, especially at higher frequencies. A good calibration (not performed in this example) is critical to achieving good balanced measurement results.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Balanced.vbs. [Learn how to setup and run the macro.](#)

### [See Other SCPI Example Programs](#)

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpIStringParser
' A comment
scpi.Parse("SYST:FPRESET")
' Put in "Single" trigger mode.
scpi.Parse("SENS:SWE:GROups:COUNT 1")
scpi.Parse("SENS:SWE:MODE GROUPS")

' This example uses DUT topology Bal-Bal -
' a DUT with a balanced input and balanced output.
'
' Port mapping for our DUT:
' logical port 1 = physical ports 1 and 4
' logical port 2 = physical ports 2 and 3
' The default is:
' logical port 1 = physical ports 1 and 2
' logical port 2 = physical ports 3 and 4
'
'      logical 1          logical 2
'      +-----+
' 1 -----|             |----- 2 +
'          |             |
'          |      DUT    |
'          |             |
' 4 -----|             |----- 3 -
```

```

scpi.Parse("CALC:FSIM:BAL:DEVIce BBALanced")
scpi.Parse("CALC:FSIM:BAL:TOPology:BBAL:PPORTs 1,4,2,3")
' Set up stimulus
scpi.Parse("SENS:SWE:POINTs 801")
scpi.Parse("SENS:FREQ:START 10e6")
scpi.Parse("SENS:FREQ:STOP 1e9")
' Turn on Four windows
scpi.Parse("DISP:WIND1:STATe ON")
scpi.Parse("DISP:WIND2:STATe ON")
scpi.Parse("DISP:WIND3:STATe ON")
scpi.Parse("DISP:WIND4:STATe ON")
' Create a trace called "sdd21", and for that trace turn on the balanced
' transformation and set the balanced transformation to BBAL SDD21.
scpi.Parse("CALC:PAR:DEF ""sdd21"",S11")
scpi.Parse("CALC:PAR:SEL ""sdd21""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDD21")
' Feed the sdd21 trace to window 1, trace 1
scpi.Parse("DISP:WIND1:TRAC1:FEED ""sdd21""")
' Similarly create 3 more balanced transmission/conversion parameters
' Create Scd21
scpi.Parse("CALC:PAR:DEF ""scd21"",S11")
scpi.Parse("CALC:PAR:SEL ""scd21""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCD21")
scpi.Parse("DISP:WIND1:TRAC2:FEED ""scd21""")
' Create Sdc21
scpi.Parse("CALC:PAR:DEF ""sdc21"",S11")
scpi.Parse("CALC:PAR:SEL ""sdc21""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDC21")
scpi.Parse("DISP:WIND1:TRAC3:FEED ""sdc21""")
' Create Scc21
scpi.Parse("CALC:PAR:DEF ""scc21"",S11")
scpi.Parse("CALC:PAR:SEL ""scc21""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCC21")
scpi.Parse("DISP:WIND1:TRAC4:FEED ""scc21""")
' Now create logical port 1 reflection parameters, and place them in window 2
scpi.Parse("CALC:PAR:DEF ""sdd11"",S11")
scpi.Parse("CALC:PAR:SEL ""sdd11""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDD11")
' Feed the sdd11 trace to window 2, trace 1
scpi.Parse("DISP:WIND2:TRAC1:FEED ""sdd11""")
' Similarly create 3 more balanced reflection/conversion parameters
scpi.Parse("CALC:PAR:DEF ""scd11"",S11")
scpi.Parse("CALC:PAR:SEL ""scd11""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATe ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCD11")
scpi.Parse("DISP:WIND2:TRAC2:FEED ""scd11""")
scpi.Parse("CALC:PAR:DEF ""sdc11"",S11")

```

```

scpi.Parse("CALC:PAR:SEL ""sdc11""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDC11")
scpi.Parse("DISP:WIND2:TRAC3:FEED ""sdc11""")
scpi.Parse("CALC:PAR:DEF ""scc11"",S11")
scpi.Parse("CALC:PAR:SEL ""scc11""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCC11")
scpi.Parse("DISP:WIND2:TRAC4:FEED ""scc11""")
' Now create reverse transmission parameters, and place them in window 3
scpi.Parse("CALC:PAR:DEF ""sdd12"",S11")
scpi.Parse("CALC:PAR:SEL ""sdd12""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDD12")
' Feed the sdd11 trace to window 3, trace 1
scpi.Parse("DISP:WIND3:TRAC1:FEED ""sdd12""")
' Similarly create 3 more balanced reverse transmission/conversion parameters
scpi.Parse("CALC:PAR:DEF ""scd12"",S11")
scpi.Parse("CALC:PAR:SEL ""scd12""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCD12")
scpi.Parse("DISP:WIND3:TRAC2:FEED ""scd12""")
scpi.Parse("CALC:PAR:DEF ""sdc12"",S11")
scpi.Parse("CALC:PAR:SEL ""sdc12""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDC12")
scpi.Parse("DISP:WIND3:TRAC3:FEED ""sdc12""")
scpi.Parse("CALC:PAR:DEF ""scc12"",S11")
scpi.Parse("CALC:PAR:SEL ""scc12""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCC12")
scpi.Parse("DISP:WIND3:TRAC4:FEED ""scc12""")
' Now create reverse reflection parameters, and place them in window 4
scpi.Parse("CALC:PAR:DEF ""sdd22"",S11")
scpi.Parse("CALC:PAR:SEL ""sdd22""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDD22")
' Feed the sdd11 trace to window 3, trace 1
scpi.Parse("DISP:WIND4:TRAC1:FEED ""sdd22""")
' Similarly create 3 more balanced reverse reflection parameters
scpi.Parse("CALC:PAR:DEF ""scd22"",S11")
scpi.Parse("CALC:PAR:SEL ""scd22""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCD22")
scpi.Parse("DISP:WIND4:TRAC2:FEED ""scd22""")
scpi.Parse("CALC:PAR:DEF ""sdc22"",S11")
scpi.Parse("CALC:PAR:SEL ""sdc22""")
scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SDC22")
scpi.Parse("DISP:WIND4:TRAC3:FEED ""sdc22""")
scpi.Parse("CALC:PAR:DEF ""scc22"",S11")
scpi.Parse("CALC:PAR:SEL ""scc22""")

```

```

scpi.Parse("CALC:FSIM:BAL:PAR:STATE ON")
scpi.Parse("CALC:FSIM:BAL:PAR:BBAL:DEF SCC22")
scpi.Parse("DISP:WIND4:TRAC4:FEED ""scc22""")
' Show statistics for forward differential transmission
scpi.Parse("CALC:PAR:SEL ""sdd21""")
scpi.Parse("CALC:FUNC:STATISTICS:STATE ON")
' Set up test range for markers. Set up range 1 to be between 100MHz and 900MHz
scpi.Parse("CALC:FUNC:DOMain:USER:START 1,100e6")
scpi.Parse("CALC:FUNC:DOMain:USER:STOP 1,900e6")
' Set up a marker on Sdd21, find minimum differential transmission
' Set this marker to use user-range 1 (previously defined)
scpi.Parse("CALC:PAR:SEL ""sdd21""")
scpi.Parse("CALC:MARK ON")
scpi.Parse("CALC:MARK:FUNC:DOM:USER:RANGE 1")
scpi.Parse("CALC:MARK:FUNC:EXEC MIN")
' Find Maximum common mode to differential mode conversion within user-range 1.
scpi.Parse("CALC:PAR:SEL ""sdc21""")
scpi.Parse("CALC:MARK ON")
scpi.Parse("CALC:MARK:FUNC:DOM:USER:RANGE 1")
scpi.Parse("CALC:MARK:FUNC:EXEC MAX")
' Set up limit lines
' 'MIN' Limit on Sdd21 between 100MHz and 900MHz at -2dB
scpi.Parse("CALC:PAR:SEL ""sdd21""")
scpi.Parse("CALC:LIM:SEGM:TYPE LMIN")
scpi.Parse("CALC:LIM:SEGM:STIM:START 100e6")
scpi.Parse("CALC:LIM:SEGM:STIM:STOP 900e6")
scpi.Parse("CALC:LIM:SEGM:AMPL:START -2")
scpi.Parse("CALC:LIM:SEGM:AMPL:STOP -2")
scpi.Parse("CALC:LIM:DISP:STATE ON")
scpi.Parse("CALC:LIM:STATE ON")
' 'MAX' Limit on Sdc21 between 100MHz and 900MHz at -20dB
scpi.Parse("CALC:PAR:SEL ""sdc21""")
scpi.Parse("CALC:LIM:SEGM:TYPE LMAX")
scpi.Parse("CALC:LIM:SEGM:STIM:START 100e6")
scpi.Parse("CALC:LIM:SEGM:STIM:STOP 900e6")
scpi.Parse("CALC:LIM:SEGM:AMPL:START -20")
scpi.Parse("CALC:LIM:SEGM:AMPL:STOP -20")
scpi.Parse("CALC:LIM:DISP:STATE ON")
scpi.Parse("CALC:LIM:STATE ON")
' "Single" trigger to take data.
scpi.Parse("SENS:SWE:MODE GROUPS")
response = scpi.Parse("*OPC?")
' Read the limit status byte to determine pass/fail.
' If bit 1 is set (+2), then our first measurement failed (Sdd21).
' If bit 3 is set (+8), then our third measurement failed (Sdc21).
' Both measurements failing would return +10.
response = scpi.Parse("STAT:QUES:LIM1:COND?")
If response <> 0 Then
Wscript.Echo "Failure of limit lines: " & response
End If

' Here we demonstrate how to determine if we have

```

```
' a balanced parameter and what type it is.
' Read back one parameter to verify its type
scpi.Parse("CALC:PAR:SEL ""sdd21""")
' Is this a balanced parameter?
isbal = scpi.Parse("CALC:FSIM:BAL:PAR?")
' Which topology/device is set?
device = scpi.Parse("CALC:FSIM:BAL:DEV?")
device = Left( device, Len(device)-1 ) ' strip off newline
' Which parameter are we measuring within that topology?
balparam = scpi.Parse("CALC:FSIM:BAL:PAR:" & device & ":DEF?")
balparam = Left( balparam, Len(balparam)-1 ) ' strip off newline
If isbal Then
WScript.Echo "Balanced Parameter: " & balparam & " in topology: " & device & "."
Else
WScript.Echo "Parameter not balanced."
End If
```



## Channels, Windows, and Measurements using SCPI

**SOURCE** and most **SENSE** commands act on the **channel** that is specified in the command. Channel 1 is default if not specified.

Most **DISPLAY** commands act on the **window and trace** specified in the command. Window1 and Trace1 are default if not specified.

**CALCulate** commands act on the **selected measurement** in the specified channel. Select the measurement for each channel using CALCulate<channel number>:PARAmeter:SELEct <meas name>. You can select one measurement in each channel.

The following Visual Basic program does the following:

- Presets the analyzer
- Create 2 windows
- Create 2 Measurements
- Feed the measurements to windows / traces
- Change frequency ranges for channels
- Select both measurements
- Turn marker 1 ON for each measurement

To run this program, you need:

- An established [GPIB interface connection](#)

### [See Other SCPI Example Programs](#)

```
GPIB.Write "SYSTem:PRESet"

'Create Measurements
GPIB.Write "CALCulate1:PARAmeter:DEFine 'Meas1',S11"
GPIB.Write "CALCulate2:PARAmeter:DEFine 'Meas2',S21"

' Turn on windows - creates if new
GPIB.Write "DISPlay:WINDow1:STATE ON"
GPIB.Write "DISPlay:WINDow2:STATE ON"

'Associate ("FEED") the measurement name('Meas1') to WINDow(1), and give the new
TRACe a number(1).
GPIB.Write "DISPlay:WINDow1:TRACe1:FEED 'Meas1'"
GPIB.Write "DISPlay:WINDow2:TRACe2:FEED 'Meas2'"

'Change each channel's frequency range
GPIB.Write "SENSE1:FREQuency:SPAN 1e9"
GPIB.Write "SENSE2:FREQuency:SPAN 2e9"
```

'Select both measurements

GPIB.Write "CALCulate1:PARAmeter:SElect 'Meas1' "

GPIB.Write "CALCulate2:PARAmeter:SElect 'Meas2' "

'Turn marker 1 ON for each measurement

GPIB.Write "CALCulate1:MARKer:STATe ON"

GPIB.Write "CALCulate2:MARKer:STATe ON"

## Setup Sweep Parameters using SCPI

---

This Visual Basic program sets up sweep parameters on the Channel 1 measurement.  
To run this program, you need:

- An established [GPIB interface connection](#)

[See Other SCPI Example Programs](#)

```
GPIB.Write "SYSTem:PRESet"
'Select the measurement
GPIB.Write "CALCulate:PARAMeter:SElect 'CH1_S11_1'"
'Set sweep type to linear
GPIB.Write "SENSe1:SWEep:TYPE LIN"

'Set IF Bandwidth to 700 Hz
GPIB.Write "SENSe1:BANDwidth 700"

'Set Center and Span Freq's to 4 GHz
GPIB.Write "SENSe1:FREQuency:CENTer 4ghz"
GPIB.Write "SENSe1:FREQuency:SPAN 4ghz"

'Set number of points to 801
GPIB.Write "SENSe1:SWEep:POINTs 801"

'Set sweep generation mode to Analog
GPIB.Write "SENSe1:SWEep:GENeration ANAL"

'Set sweep time to Automatic
GPIB.Write "SENSe1:SWEep:TIME:AUTO ON"

'Query the sweep time
GPIB.Write "SENSe1:SWEep:TIME?"
SweepTime = GPIB.Read
```

## Setup the Display using SCPI

---

This Visual Basic program:

- Sets data formatting
- Turns ON the Trace, Title, and Frequency Annotation
- Autoscales the Trace
- Queries Per Division, Reference Level, and Reference Position
- Turn ON and set averaging
- Turn ON and set smoothing

To run this program, you need:

- An established [GPIB interface connection](#)

### [See Other SCPI Example Programs](#)

```
GPIB.Write "SYSTem:PRESet"

'Select the measurement
GPIB.Write "CALCulate:PARAmeter:SElect 'CH1_S11_1'"

'Set the Data Format to Log Mag
GPIB.Write ":CALCulate1:FORMat MLOG"

'Turn ON the Trace, Title, and Frequency Annotation
GPIB.Write "Display:WINDow1:TRACe1:STATe ON"
GPIB.Write "DISPlay:WINDow1:TITLe:STATe ON"
GPIB.Write "DISPlay:ANNotation:FREQuency ON"

'Autoscale the Trace
GPIB.Write "Display:WINDow1:TRACe1:Y:Scale:AUTO"

'Query back the Per Division, Reference Level, and Reference Position
GPIB.Write "DISPlay:WINDow1:TRACe1:Y:SCALE:PDIVision?"
Pdiv = GPIB.Read
GPIB.Write "DISPlay:WINDow1:TRACe1:Y:SCALE:RLEVel?"
Rlev = GPIB.Read
GPIB.Write "DISPlay:WINDow1:TRACe1:Y:SCALE:RPOSition?"
Ppos = GPIB.Read

'Turn ON, and average five sweeps
GPIB.Write "SENSe1:AVERAge:STATe ON"
GPIB.Write "SENSe1:AVERAge:Count 5"

'Turn ON, and set 20% smoothing aperture
GPIB.Write "CALCulate1:SMOothing:STATe ON"
```

```
GPIB.Write "CALCulate1:SMOothing:APERture 20"
```

## Triggering the PNA using SCPI

---

To understand how to trigger the PNA using SCPI, it is very important to understand the [PNA trigger model](#). Here is a very simple explanation. These three separate functions control PNA triggering:

1. [Trigger:Source](#) - Where the trigger signals originate:
  - Internal Continuous
  - Internal Manual (Single)
  - External - a trigger source that is connected to the PNA rear panel.
2. [Trigger:Scope](#) - what gets triggered:
  - Global - each signal triggers all channels in turn.
  - Channel - each signal triggers ONE channel.
3. Channel settings ([Sense<ch>:Sweep:Mode](#)) How many triggers will each channel accept before going into hold.
  - HOLD - channel will not trigger.
  - CONTinuous - channel triggers indefinitely.
  - GROups - channel accepts the number of triggers specified with the last [SENS:SWE:GRO:COUN<num>](#).
  - **SINGLE** - channel accepts ONE trigger, then goes to HOLD.
  - Point trigger [SENS1:SWE:TRIG:POINT](#)

When controlling the PNA using SCPI, a SINGLE trigger is used to ensure that a complete sweep is taken. This example demonstrates how to Single trigger the PNA using the following two methods:

- **Simplest Triggering** This method sets the Trigger Source to Internal Continuous - a stream of trigger signals. Each channel is configured to ACCEPT only a single trigger signal, then HOLD. This method can also be used when an External trigger source sends a continuous stream of trigger signals.
- **Advanced Triggering** This method SENDS a single trigger from the Source, which can be from either Internal (using INIT:IMM) or External triggering. Each channel is configured to accept an unlimited number of triggers. This method is the only way to perform point triggering.  
When you require some channels to accept continuous triggers and other channels to accept single triggers, see [INIT:IMM Advanced](#) to learn how.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text

editor file such as Notepad and save it on the PNA hard drive as Trigger.vbs. [Learn how to setup and run the macro.](#)

**Measurement setup example:** This section of code can be used at the start of both methods. It sets up:

- S11 traces on two channels
- 10 data points
- Sweep time of 2 seconds - this is slow enough to allow us to watch as each trace is triggered.

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
'=====
'Setup the PNA
'Preset the analyzer
scpi.Execute ("SYST:FPRreset")
' Create and turn on window/channel 1
scpi.Execute ("DISPlay:WINDow1:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate1:PARAMeter:DEFine 'MyMeas1',S11")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (1)
scpi.Execute ("DISPlay:WINDow1:TRACe1:FEED 'MyMeas1'")
' Create and turn on window/channel 2
scpi.Execute ("DISPlay:WINDow2:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate2:PARAMeter:DEFine 'MyMeas2',S11")
'Associate ("FEED") the measurement name ('MyMeas') to WINDow (2)
scpi.Execute ("DISPlay:WINDow2:TRACe2:FEED 'MyMeas2'")
'Set slow sweep so we can see
scpi.Execute ("SENS1:SWE:TIME 2")
scpi.Execute ("SENS2:SWE:TIME 2")
'set number of points to 10
scpi.Execute ("SENS1:SWE:POIN 10")
scpi.Execute ("SENS2:SWE:POIN 10")
```

```

'=====
' Put both channels in Hold
scpi.Execute ("SENS1:SWE:MODE HOLD")
scpi.Execute ("SENS2:SWE:MODE HOLD")
'=====
'Pick Single Send or Single Accept
resp=Msgbox ("Single Send? - Click No for Single Accept", 4, "PNA Trigger Demo")
If resp=6 Then
SingleSend()
Else
SingleAccept()
End If

```

**Simple Triggering** The following example sends a continuous stream of trigger signals and each PNA channel is set to ACCEPT only a signal trigger signal, then HOLD.

- This example can be used to configure External triggering where the trigger source sends a continuous stream of trigger signals. Configure the type of trigger signal that the PNA responds to using the [CONTRol:SIGNal](#) command. The command in this example sets the PNA to respond to HIGH TTL signals at the rear-panel BNC1 trigger IN connector. This command also automatically sets Trigger Source to External Trigger.
- The [TRIG SCOPE](#) (Global or Channel) setting is NOT necessary with a continuous stream of trigger signals. The example program directly controls when each channel is triggered.
- Point triggering can NOT be used with a continuous stream of trigger signals because in point triggering the channel will accept as many triggers as necessary to complete ONE full sweep. Use the single SEND example for point triggering.



```

Sub SingleAccept()
'PNA sends continuous trigger signals
scpi.Execute ("TRIG:SOUR IMMediate")
'Uncomment the following to set External triggering
'scpi.Execute ("CONT:SIGN BNC1,TILHIGH")
AcceptOne()
End Sub

Sub AcceptOne()
'The following command makes the channel immediately sweep
'*OPC? allows the measurement to complete before the controller sends another
command
scpi.Execute ("SENS1:SWE:MODE SINGLE;*OPC?")
' You could do something to ch2 here before sweeping it
scpi.Execute ("SENS2:SWE:MODE SINGLE;*OPC?")
resp=Msgbox ("Another trigger?", 1, "PNA Trigger Demo")
If resp=1 Then
AcceptOne()
End If
End Sub

```

**Advanced Trigger** This example section performs Single Send triggering. Here, single triggering is accomplished by SENDING one trigger signal from the Trigger source and each channel is setup to accept unlimited trigger signals. See the [INIT:IMM](#) command for more details.

- Using this method, it is possible to change [Trigger:Scope](#) to Global or Channel. Set trigger scope to channel if there is some code to execute between channel measurements. Similarly, this method can be used to set [Point triggering](#). Use this method if there is some code to execute between data point measurements.
- In addition, this method can also be used to perform External triggering if the external trigger source is capable of SENDING single triggers. See the [CONTrol:SIGNal](#) command to set the type of signal to which the PNA will respond.
- If the external source can only send a continuous stream of trigger signals, then the [Single Accept](#) section must be used.

```

Sub SingleSend()
'Set Source Internal - Manual Triggering
scpi.Execute ("TRIG:SOUR MANual")

```

```

'If using an External trigger source that is capable of
'sending SINGLE trigger signals, then uncomment the following.
'This command automatically sets trigger source to External
'scpi.Execute ("CONT:SIGN BNC1,TILHIGH")

'Setup Trigger Scope
'WHAT gets triggered
'Pick one using comments
'Set Channel triggering
'scpi.Execute ("TRIG:SCOPE CURRENT")
'Set Global triggering (Default)
scpi.Execute ("TRIG:SCOPE ALL")

'Set Channel Settings
'The channels respond to UNLIMITED trigger signals (Default)
scpi.Execute ("SENS1:SWE:MODE CONTinuous")
scpi.Execute ("SENS2:SWE:MODE CONTinuous")

'To do Point trigger on one or more channels, uncomment the following.
'Point trigger automatically sets Trig:Scope to Current/Channel
'scpi.Execute ("SENS1:SWE:TRIG:POINT ON")
'scpi.Execute ("SENS2:SWE:TRIG:POINT ON")
IntTrig()
End Sub

Sub IntTrig()
'If External triggering, replace this Sub with code
'to single trigger the External Trig Source
Dim resp
'*OPC? allows the measurement to complete before the controller sends another
command
scpi.Execute ("INITiate:IMMediate;*OPC?")
resp=Msgbox ("Another trigger?", 1, "PNA Trigger Demo")
If resp=1 Then
IntTrig()
End If
End Sub

```

---

Last modified:

18-Jun-2007	Updated with Sens:Swe:Mode Single
June 6, 2007	Changed order and wording
April 24, 2007	Updated with links
Oct. 5, 2006	New topic

## GPIB using Visual C++

---

[See Other SCPI Example Programs](#)

```
/*
 * This example assumes the user's PC has a National Instruments GPIB board. The
 * example is comprised of three basic parts:
 *
 * 1. Initialization
 * 2. Main Body
 * 3. Cleanup
 *
 * The Initialization portion consists of getting a handle to the PNA and then doing
 * a GPIB clear of the PNA.
 *
 * The Main Body consists of the PNA SCPI example.
 *
 * The last step, Cleanup, releases the PNA for front panel control.
 */

#include <stdio.h>
#include <stdlib.h>

/*
 * Include the WINDOWS.H and DECL-32.H files. The standard Windows
 * header file, WINDOWS.H, contains definitions used by DECL-32.H and
 * DECL-32.H contains prototypes for the NI GPIB routines and constants.
 */
#include <windows.h>
#include "decl-32.h"

#define ERRMSG_SIZE 1024 // Maximum size of SCPI command string
#define ARRAY_SIZE 1024 // Size of read buffer

#define BDINDEX 0 // Board Index of GPIB board
#define PRIMARY_ADDR_OF_PNA 16 // GPIB address of PNA
#define NO_SECONDARY_ADDR 0 // PNA has no Secondary address
#define TIMEOUT T10s // Timeout value = 10 seconds
#define EOTMODE 1 // Enable the END message
#define EOSMODE 0 // Disable the EOS mode

int pna;
char ValueStr[ARRAY_SIZE + 1];
char ErrorMnemonic[21][5] = {"EDVR", "ECIC", "ENOL", "EADR", "EARG",
    "ESAC", "EABO", "ENEB", "EDMA", "",
    "EOIP", "ECAP", "EFSO", "", "EBUS",
    "ESTB", "ESRQ", "", "", "", "ETAB"};

void GPIBWrite(char* SCPIcmd);
char *GPIBRead(void);
```

```

void GPIBCleanup(int Dev, char* ErrorMsg);

int main()
{

char *opc;
char *result;
char *value;

/*
 * =====
 * INITIALIZATION SECTION
 * =====
 */

/*
 * The application brings the PNA online using ibdev. A device handle,pna, is
returned and is used in all subsequent calls to the PNA.
 */
pna = ibdev(BDINDEX, PRIMARY_ADDR_OF_PNA, NO_SECONDARY_ADDR,
TIMEOUT, EOTMODE, EOSMODE);
if (ibsta & ERR)
{
printf("Unable to open handle to PNA/nibsta = 0x%x iberr = %d/n",
ibsta, iberr);
return 1;
}

/*
 * Do a GPIB Clear of the PNA. If the error bit ERR is set in ibsta, call
GPIBCleanup with an error message.
 */
ibclr (pna);
if (ibsta & ERR)
{
GPIBCleanup(pna, "Unable to perform GPIB clear of the PNA");
return 1;
}

/*
 * =====
 * MAIN BODY SECTION
 * =====
 */

// Reset the analyzer to instrument preset
GPIBWrite("SYSTem:FPRESET");

// Create S11 measurement
GPIBWrite("CALCulatel:PARAMeter:DEFine 'My_S11',S11");

```

```

// Turn on Window #1
GPIBWrite("DISPlay:WINDow1:STATe ON");

// Put a trace (Trace #1) into Window #1 and 'feed' it from the measurement
GPIBWrite("DISPlay:WINDow1:TRACe1:FEED 'My_S11'");

// Setup the channel for single sweep trigger
GPIBWrite("INITiate1:CONTinuous OFF;*OPC?");
opc = GPIBRead();
GPIBWrite("SENSe1:SWEep:TRIGger:POINT OFF");

// Set channel parameters
GPIBWrite("SENSe1:SWEep:POINTs 11");
GPIBWrite("SENSe1:FREQuency:START 1000000000");
GPIBWrite("SENSe1:FREQuency:STOP 2000000000");

// Send a trigger to initiate a single sweep
GPIBWrite("INITiate1;*OPC?");
opc = GPIBRead();

// Must select the measurement before we can read the data
GPIBWrite("CALCulate1:PARAMeter:SElect 'My_S11'");

// Read the measurement data into the "result" string variable
GPIBWrite("FORMat ASCII");
GPIBWrite("CALCulate1:DATA? FDATA");
result = GPIBRead();

// Print the data to the display console window
printf("S11(dB) - Visual C++ SCPI Example for PNA/n/n");
value = strtok(result, ",");
while (value != NULL)
{
    printf("%s/n", value);
    value = strtok(NULL, ",");
}

/*
 * =====
 * CLEANUP SECTION
 * =====
 */

/* The PNA is returned to front panel control. */
ibonl(pna, 0);

return 0;
}

/*
 * Write to the PNA
 */

```

```

void GPIBWrite(char* SCPIcmd)
{
int length;
char ErrorMessage[ERRMSGSIZE + 1];
length = strlen(SCPIcmd) ;

    ibwrt (pna, SCPIcmd, length);
    if (ibsta & ERR)
    {
        strcpy(ErrorMessage, "Unable to write this command to PNA:/n");
        strcat(ErrorMessage, SCPIcmd);

        GPIBCleanup(pna, ErrorMessage);
        exit(1);
    }
}

/*
 * Read from the PNA
 */
char* GPIBRead(void)
{
    ibrd (pna, ValueStr, ARRAYSIZE);
    if (ibsta & ERR)
    {
        GPIBCleanup(pna, "Unable to read from the PNA");
        exit(1);
    }
else
    return ValueStr;
}

/*
 * After each GPIB call, the application checks whether the call succeeded. If an
NI-488.2 call fails, the GPIB driver sets the corresponding bit in the global status
variable. If the call failed, this procedure prints an error message, takes the PNA
offline and exits.
 */
void GPIBCleanup(int Dev, char* ErrorMessage)
{
    printf("Error : %s/nibsta = 0x%x iberr = %d (%s)/n",
        ErrorMessage, ibsta, iberr, ErrorMnemonic[iberr]);
    if (Dev != -1)
    {
        printf("Cleanup: Returning PNA to front panel control/n");
        ibonl (Dev, 0);
    }
}

```

## Perform a Guided 2-Port or 4-Port Cal using SCPI

---

This example performs a Guided 2-Port or 4-port Calibration using ONE set of calibration standards or an ECAL module.

A measurement must first be set up with desired frequency range, power, and so forth, ready to be calibrated.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file, such as Notepad, and save it on the PNA hard drive as \*.vbs.

[Learn how to setup and run the macro.](#)

[See Guided Cal SCPI commands](#)

---

### [See Other SCPI Example Programs](#)

---

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpStringParser

' To perform 2-port cal, Uncomment TwoPortGuidedCal()
' Then comment FourPortGuidedCal()

'Do 2-port Cal
'TwoPortGuidedCal()

'Do 4-port Cal
FourPortGuidedCal

Sub TwoPortGuidedCal()
' Select the connectors
scpi.Execute("sens:corr:coll:guid:conn:port1 ""APC 3.5 female"" ")
scpi.Execute("sens:corr:coll:guid:conn:port2 ""APC 3.5 male"" ")
scpi.Execute("sens:corr:coll:guid:conn:port3 ""Not used"" ")
scpi.Execute("sens:corr:coll:guid:conn:port4 ""Not used"" ")
MsgBox("Connectors defined for Ports 1 and 2")
' Select the Cal Kit for each port being calibrated.
scpi.Execute("sens:corr:coll:guid:ckit:port1 ""85052D"" ")
scpi.Execute("sens:corr:coll:guid:ckit:port2 ""85052D"" ")
' To use an ECal module instead, comment out the above two lines
' and uncomment these two lines and use the part number printed
' on your module (which in our case was N4691-60004), followed
' by the word 'ECAL'. Your ECal module must already be connected
' via USB to the PNA.
'scpi.Execute("sens:corr:coll:guid:ckit:port1 ""N4691-60004 ECal"" ")
'scpi.Execute("sens:corr:coll:guid:ckit:port2 ""N4691-60004 ECal"" ")
```



```

MsgBox("Cal kits defined for Ports 1 and 2")
' Initiate the calibration and query the number of steps
numSteps = GenerateSteps()
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub

Sub FourPortGuidedCal()
' Select the connectors
scpi.Execute("sens:corr:coll:guid:conn:port1 ""APC 3.5 female"" ")
scpi.Execute("sens:corr:coll:guid:conn:port2 ""APC 3.5 female"" ")
scpi.Execute("sens:corr:coll:guid:conn:port3 ""APC 3.5 female"" ")
scpi.Execute("sens:corr:coll:guid:conn:port4 ""APC 3.5 female"" ")
MsgBox("Connectors defined for Ports 1 to 4")
' Select the Cal Kit for each port being calibrated.
scpi.Execute("sens:corr:coll:guid:ckit:port1 ""85052D"" ")
scpi.Execute("sens:corr:coll:guid:ckit:port2 ""85052D"" ")
scpi.Execute("sens:corr:coll:guid:ckit:port3 ""85052D"" ")
scpi.Execute("sens:corr:coll:guid:ckit:port4 ""85052D"" ")
' To use an ECal module instead, comment out the above four lines
' and uncomment these four lines and use the part number printed
' on your module (which in our case was N4431-60003), followed
' by the word 'ECal'. Your ECal module must already be connected
' via USB to the PNA.
'scpi.Execute("sens:corr:coll:guid:ckit:port1 ""N4431-60003 ECal"" ")
'scpi.Execute("sens:corr:coll:guid:ckit:port2 ""N4431-60003 ECal"" ")
'scpi.Execute("sens:corr:coll:guid:ckit:port3 ""N4431-60003 ECal"" ")
'scpi.Execute("sens:corr:coll:guid:ckit:port4 ""N4431-60003 ECal"" ")
MsgBox("Cal kits defined for Ports 1 to 4")
' Initiate the calibration and query the number of steps
numSteps = GenerateSteps()
' If your selected cal kit is not a 4-port ECal module which can
' mate to all 4 ports at once, then you may want to choose which
' thru connections to measure for the cal. You must measure at
' least 3 different thru paths for a 4-port cal (for greatest
' accuracy you can choose to measure a thru connection for all 6
' pairings of the 4 ports). If you omit this command, the default
' is to measure from port 1 to port 2, port 1 to port 3, and
' port 1 to port 4. For this example we select to measure
' from port 1 to port 2, port 2 to port 3, and port 2 to port 4.
scpi.Execute("sens:corr:coll:guid:thru:ports 1,2,2,3,2,4")
' Re-generate the connection steps to account for the thru changes
numSteps = GenerateSteps()
' Measure the standards, compute and apply the cal
MeasureAndComplete(numSteps)
End Sub

Function GenerateSteps()
' Initiate the calibration and query the number of steps
scpi.Execute("sens:corr:coll:guid:init")
GenerateSteps = scpi.Execute("sens:corr:coll:guid:steps?")
End Function

```

```

Sub MeasureAndComplete(numSteps)
MsgBox("Number of steps is " + CStr(numSteps))
' Measure the standards
For i = 1 To numSteps
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = scpi.Execute("sens:corr:coll:guid:desc? " + CStr(i))
MsgBox strPrompt, vbOKOnly, step
' Note: if you have set up a slow sweep speed (for example, if
' you're using a narrow IF bandwidth) or you're using ECal, and
' while a cal step is being measured you wish to have your program
' perform other operations (like checking for the click event of a
' Cancel button) and you're NOT using the COM ScpiStringParser,
' you can use the optional ASYNchronous argument with the ACQUIRE
' command as shown in this commented-out line below. The SCPI
' parser then will return immediately while the cal step measurement
' proceeds (i.e., the parser will NOT block-and-wait for the
' measurement step to finish, so you can send additional commands
' in the meantime). So you can do "*ESR?" or "*STB?" queries to
' monitor the status register bytes to see when the OPC bit gets set,
' which indicates the cal measurement step has finished. This OPC
' detection works for all of the PNA's SCPI parsers except the COM
' ScpiStringParser.
' "sens:corr:coll:guid:acq STAN" + CStr(i) + ",ASYN;*OPC"
scpi.Execute("sens:corr:coll:guid:acq STAN" + CStr(i))
Next
' Conclude the calibration
scpi.Execute("sens:corr:coll:guid:save")
MsgBox ("Cal is done!")
End Sub

```

## Perform a Guided Calibration using SCPI

---

This VBScript program performs a Guided Calibration using ECal **or** Mechanical standards. This example includes optional ECal orientation features.

- This example has been updated to include the setting of Unknown Thru or Adapter Removal adapter delay. (March 2006).
- This example has been updated to show the activation of a channel to be calibrated. (Aug. 2006).

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Guided.vbs. [Learn how to setup and run the macro.](#)

```
' Performing a Guided 2-port cal (Ports 1 and 2)
TwoPortGuidedCal
Sub TwoPortGuidedCal
Dim app
Dim scpi
Dim connList
Dim selectedConn1, selectedConn2
Dim kitList
Dim selectedKit
Dim message
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser

'The following demonstrates that the Active Channel is cal'd
'Preset the PNA
scpi.Execute "SYST:PRES"
'Create a new measurement on Chan 2
'Now there are two windows, channels and measurements
'This becomes the Active Measurement
scpi.Execute ("DISPlay:WINDow2:STATE ON")
'Define a measurement name, parameter
scpi.Execute ("CALCulate2:PARAmeter:DEFine 'MyMeas',S21")
'"FEED" the measurement
scpi.Execute ("DISPlay:WINDow2:TRACe1:FEED 'MyMeas'")
```

```

'This is the Active Measurement
'Activate the 'Preset' measurement to cal chan 1
scpi.Execute("CALC1:PAR:SEL 'CH1_S11_1'")

' Query the list of connectors that the PNA system recognizes
connList = scpi.Execute("sens:corr:coll:guid:conn:cat?")
' Format the list with linefeed characters in place of the commas
connList = FormatList(connList)
message = "Enter your DUT connector for Port 1. Choose from this list:"
message = message & Chr(10) & Chr(10) & connList
' Select the connector for Port 1
selectedConn1 = InputBox(message)
If selectedConn1 = "" Then Exit Sub
scpi.Execute "sens:corr:coll:guid:conn:port1 '" & selectedConn1 & "'"
message = "Enter your DUT connector for Port 2. Again, choose from this list:"
message = message & Chr(10) & Chr(10) & connList
' Select the connector for Port 2
selectedConn2 = InputBox(message)
If selectedConn2 = "" Then Exit Sub
scpi.Execute "sens:corr:coll:guid:conn:port2 '" & selectedConn2 & "'"
' Note: If your PNA has more than 2 ports, then uncomment
' one or both of these next two lines.
'scpi.Execute "sens:corr:coll:guid:conn:port3 '"Not used'" "
'scpi.Execute "sens:corr:coll:guid:conn:port4 '"Not used'" "

' This next block of commented code demonstrates how to specify an adapter
' and it's electrical delay, in situations where you are performing an
' Unknown Thru or Adapter Removal calibration. In most situations, the
' PNA is able to correctly determine an adapter's electrical length
' at the end of the calibration. However, there are scenarios where
' the PNA cannot correctly calculate the length -- such as when the channel
' has a relatively small number of measurement points (for example, 201 or less)
' and the adapter is significantly long (for example, a cable that is several feet).
' In these cases, the ADAP commands (below) enable you to explicitly specify
' the adapter you are using.
' Send these commands prior to the "sens:corr:coll:guid:init" command.

```

```

' Create adapter and return the adapter number
'adapterNum = scpi.Execute("sens:corr:coll:guid:adap:cre? " & selectedConn1 & ", "& selectedConn2 & " ")
' The adapterNum string contains a '+' character.
' Here we convert to integer to remove that.
'adapterNum = CStr( CInt(adapterNum) )
' Specify that this adapter has 10 nanoseconds electrical delay (coaxial).
'scpi.Execute "sens:corr:coll:guid:adap" & adapterNum & ":del 10E-9"
' Text description of adapter
'scpi.Execute "sens:corr:coll:guid:adap" & adapterNum & ":desc 'My adapter'"
' Select to use this adapter specifically between ports 1 and 2
'scpi.Execute "sens:corr:coll:guid:adap" & adapterNum & ":path 1,2"
' End of adapter block

' Query the list of acceptable cal kits and
' ECal module characterizations for Port 1.
kitList = scpi.Execute("sens:corr:coll:guid:ckit:port1:cat?")
' Format the list with linefeed
' characters in place of the commas
kitList = FormatList(kitList)
message = "Enter your cal kit or ECal module characterization for Port 1. "
message = message & "Choose from this list:"
message = message & Chr(10) & Chr(10) & kitList
' Select the Cal Kit or ECal module
' characterization to use for Port 1.
selectedKit = InputBox(message)
If selectedKit = "" Then Exit Sub
scpi.Execute "sens:corr:coll:guid:ckit:port1 " & selectedKit & " "

' Query the list of acceptable cal kits
' and ECal module characterizations for Port 2.
kitList = scpi.Execute("sens:corr:coll:guid:ckit:port2:cat?")
' Format the list with linefeed characters in place of the commas
kitList = FormatList(kitList)
message = "Enter your cal kit or ECal module characterization for Port 2. "
message = message & "Choose from this list:"
message = message & Chr(10) & Chr(10) & kitList
' Select the Cal Kit or ECal module

```

```

' characterization to use for Port 2.
selectedKit = InputBox(message)
If selectedKit = "" Then Exit Sub
scpi.Execute "sens:corr:coll:guid:ckit:port2 '" & selectedKit & "'"

' This next block of commented code
' shows optional functions when using ECal.
' Send these "sens:corr:pref" commands prior to the
' "sens:corr:coll:guid:init" command.
' Read ECal information from ECal module #1 on the USB bus
' about the Agilent factory characterization data
'module1Info = scpi.Execute("sens:corr:coll:ckit:inf? ECAL1,CHAR0")
'MsgBox "Description of ECal Module #1:" & Chr(10) & Chr(10) & module1Info

' The following command enables auto orientation of
' the ECal module (The PNA senses which port of the
' module is connected to which port of the PNA).
'scpi.Execute "sens:corr:pref:ecal:ori ON"
' However, if you are measuring at very low power levels where
' the PNA may fail to sense the module's orientation, then turn auto
' orientation OFF and specify how the module is connected.
' "A1,B2" indicates Port A of the module is connected
' to PNA Port 1 and Port B is connected to PNA Port 2).
'scpi.Execute "sens:corr:pref:ecal:ori OFF"
'scpi.Execute "sens:corr:pref:ecal:pmap ECAL1,'A1,B2'"
' End of optional ECal setup

' Select the thru method of "Default". This instructs the PNA to
' determine which thru standard measurement technique to use
' based upon the selected connectors and
' calibration kit(s) and the PNA model number.
' with new CMET and TMET 'default' is set by not sending the commands
'
' Initiate the calibration and query the number of steps
scpi.Execute "sens:corr:coll:guid:init"
numSteps = scpi.Execute("sens:corr:coll:guid:steps?")

```

```

MsgBox "Number of steps is " + CStr(numSteps)
' Measure the standards
For i = 1 To numSteps
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = scpi.Execute("sens:corr:coll:guid:desc? " + CStr(i))
MsgBox strPrompt, vbOKOnly, step
scpi.Execute "sens:corr:coll:guid:acq STAN" + CStr(i)
Next
' Conclude the calibration
scpi.Execute "sens:corr:coll:guid:save"
MsgBox "Cal is done!"
End Sub

Function FormatList(list)
Dim tokens
' Strip the leading and trailing quotation
' marks from the list string
list = Mid(list, 2, Len(list) - 3)
' Tokenize the comma-delimited list string
' into an array of the individual substrings
tokens = Split(list, ",")
' Rebuild the list string, placing linefeed
' characters where the commas were,
' using Trim to remove leading and trailing spaces.
list = ""
For i = 0 To UBound(tokens)
tokens(i) = Trim(tokens(i))
list = list & tokens(i) & Chr(9)
If i < UBound(tokens) Then
i = i + 1
tokens(i) = Trim(tokens(i))
list = list & tokens(i) & Chr(10)
End If
Next
FormatList = list
End Function

```

Last Modified:

14-May-2007    MX Updated for new CMET and TMET commands



## Perform Guided ECal using SCPI

---

This VBScript program performs a Guided ECal Calibration. While this example is good to use as a starting point for Guided ECal, the [Guided comprehensive cal example](#) has some advanced features that are not in this program.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Guided.vbs. [Learn how to setup and run the macro.](#)

```
' Performing a 2-port cal (Ports 1 and 2)
Dim app
Dim scpi

' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser

' Specify the DUT connectors
' (for each connector of your DUT, one of the ECal module's ports must have
' that same connector, or else you cannot achieve the cal using that module).
scpi.Execute "sens:corr:coll:guid:conn:port1 ""APC 3.5 female"" "
scpi.Execute "sens:corr:coll:guid:conn:port2 ""APC 3.5 male"" "

' Note: If your PNA has more than 2 ports, you would need to uncomment
' one or both of these next two lines, to explicitly specify this is
' just a 2-port cal.
'scpi.Execute "sens:corr:coll:guid:conn:port3 ""Not used"" "
'scpi.Execute "sens:corr:coll:guid:conn:port4 ""Not used"" "
MsgBox "Connectors defined for Ports 1 and 2"

' Select the ECal module for each port being calibrated.
' Replace N4691-60004 with the part number printed on your module
' followed by User x if using a User characterization, then the word 'ECal'
' Your ECal module must already be connected via USB to the PNA.
scpi.Execute "sens:corr:coll:guid:ckit:port1 ""N4691-60004 ECal"" "
scpi.Execute "sens:corr:coll:guid:ckit:port2 ""N4691-60004 User 1 ECal"" "
MsgBox "Cal kits defined for Ports 1 and 2"

' Initiate the calibration and query the number of steps
scpi.Execute "sens:corr:coll:guid:init"
numSteps = scpi.Execute("sens:corr:coll:guid:steps?")
MsgBox "Number of steps is " + CStr(numSteps)

' Measure the standards
For i = 1 To numSteps
    step = "Step " + CStr(i) + " of " + CStr(numSteps)
    strPrompt = scpi.Execute("sens:corr:coll:guid:desc? " + CStr(i))
    MsgBox strPrompt, vbOKOnly, step
    scpi.Execute "sens:corr:coll:guid:acq STAN" + CStr(i)
```

Next

```
' Conclude the calibration  
scpi.Execute "sens:corr:coll:guid:save"  
MsgBox "Cal is done!"
```

## Perform Guided Mechanical Cal using SCPI

---

This VBScript program performs a Guided Calibration using Mechanical standards. While this example is good to use as a starting point for guided mechanical cal, the [Guided comprehensive cal example](#) has some advanced features that are not in this program.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Guided.vbs. [Learn how to setup and run the macro.](#)

' Performing a 2-port cal (Ports 1 and 2)

```
Dim app
Dim scpi

' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser

' Specify the DUT connectors
scpi.Execute "sens:corr:coll:guid:conn:port1 ""APC 3.5 female"" "
scpi.Execute "sens:corr:coll:guid:conn:port2 ""APC 3.5 male"" "

' Note: If your PNA has more than 2 ports, you would need to uncomment
' one or both of these next two lines, to explicitly specify this is
' just a 2-port cal.
'scpi.Execute "sens:corr:coll:guid:conn:port3 ""Not used"" "
'scpi.Execute "sens:corr:coll:guid:conn:port4 ""Not used"" "
MsgBox "Connectors defined for Ports 1 and 2"

' Select the Cal Kit for each port being calibrated.
scpi.Execute "sens:corr:coll:guid:ckit:port1 ""85052D"" "
scpi.Execute "sens:corr:coll:guid:ckit:port2 ""85052D"" "
MsgBox "Cal kits defined for Ports 1 and 2"

' Initiate the calibration and query the number of steps
scpi.Execute "sens:corr:coll:guid:init"
numSteps = scpi.Execute("sens:corr:coll:guid:steps?")
MsgBox "Number of steps is " + CStr(numSteps)

' Measure the standards
'The following series of commands shows that standards
'can be measured in any order. These steps acquire
'measurement of standards in reverse order.
'It is easiest to iterate through standards using
'a For-Next Loop.
For i = numSteps To 1
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = scpi.Execute("sens:corr:coll:guid:desc? " + CStr(i))
MsgBox strPrompt, vbOKOnly, step
```

```
scpi.Execute "sens:corr:coll:guid:acq STAN" + CStr(i)
Next

' Conclude the calibration
scpi.Execute "sens:corr:coll:guid:save"
MsgBox "Cal is done!"
```

---

Last Modified:

20-Jan-2007    Added note about any order for steps.

## Perform a Guided 1-Port Cal on Port 2

---

This VBScript program does the following:

1. Clear measurements from the PNA
2. Create a new S22 measurement
3. Set an instrument state
4. Select the connector types
5. Select a cal kit
6. Initiate a Guided calibration
7. Display a prompt to connect each standard
8. Save the calibration to a newly created cal set

**Note:** This example illustrates an important step when calibrating a reflection measurement in the reverse direction. You MUST create a reverse (S22) measurement and have it be the active (selected) measurement on the channel that is being calibrated. This is not necessary for any calibrating any other measurement parameter.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Guided.vbs. [Learn how to setup and run the macro.](#)

```
Dim App
Set App = CreateObject("AgilentPNA835x.Application")
App.Preset

Dim step
Dim Parser
Dim prompt
Dim txtDat
Dim Chan

Rem Clear old measurements
App.Reset

Rem Create a new Measurement
Set Parser = App.SCPIStringParser
Parser.Parse "DISPlay:WINDow1:STATE ON"
Parser.Parse "CALCulate:PARAMeter:DEFine 'MyMeas', S22"
Parser.Parse "DISPlay:WINDow1:TRACe1:FEED 'MyMeas' "

Rem Initialize state
Set Chan = App.ActiveChannel
Chan.StartFrequency = 200e6
```

```

Chan.StopFrequency = 1.5e9
Chan.IFBandwidth = 1000
step = 3

Rem Begin a guided calibration
Parser.Parse "SENS:CORR:COLL:GUID:CONN:PORT1 'Not used'"
Parser.Parse "SENS:CORR:COLL:GUID:CONN:PORT2 'Type N (50) male'"
Parser.Parse "SENS:CORR:COLL:GUID:CKIT:PORT1 '"
Parser.Parse "SENS:CORR:COLL:GUID:CKIT:PORT2 '85054D'"
Parser.Parse "SENS:CORR:COLL:GUID:INIT"

Rem Query the number of steps
txtDat = Parser.Parse("SENS:CORR:COLL:GUID:STEP?")

Rem Display the number of steps
MsgBox("Number of steps is " + txtDat)

Rem Set the loop counter limit
step = txtDat

Rem Measure the standards
For i = 1 To step
If i= 1 Then
prompt = Parser.Parse("sens:corr:coll:guid:desc? 1")
MsgBox(prompt)
Parser.Parse ("sens:corr:coll:guid:acq STAN1")
ElseIf i = 2 then
prompt = Parser.Parse("sens:corr:coll:guid:desc? 2")
MsgBox(prompt)
Parser.Parse ("sens:corr:coll:guid:acq STAN2")
ElseIf i = 3 then
prompt = Parser.Parse("sens:corr:coll:guid:desc? 3")
MsgBox(prompt)
Parser.Parse ("sens:corr:coll:guid:acq STAN3")
End If
Next

Rem All standards have been measured. Save the result
Parser.Parse "SENS:CORR:COLL:GUID:SAVE"
MsgBox("The calibration has been completed")

```

## Perform Guided TRL Calibration

---

This VBScript file performs a 2-Port Guided TRL calibration on **2-port PNA analyzers**. ([See an example of TRL cal on a 4-port PNA.](#)) This program does the following:

- Clear old measurements from the PNA
- Create a new S22 measurement
- Set an instrument state
- Select the connectors and cal kit
- Initiate a Guided calibration
- Display a prompt as each new standard must be connected
- Save the calibration to a newly created cal set.

**Note:** This program runs without error on all PNA code revisions 7.21 and higher.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as TRL.vbs. [Learn how to setup and run the macro.](#)

```
Dim App
Dim Parser
Dim Chan
Dim txtDat
Dim step
Dim parserTxt
Dim prompt
Set App = CreateObject("AgilentPNA835x.Application")
' Clear old measurements
App.Reset
' Create a new Measurement
Set Parser = App.SCPIStringParser
Parser.Parse "DISPlay:WINDow1:STATE ON"
Parser.Parse "CALCulate:PARAMeter:DEFine 'MyMeas',S12"
Parser.Parse "DISPlay:WINDow1:TRACe1:FEED 'MyMeas'"
' Initialize state
Set Chan = App.ActiveChannel
```

```

Chan.StartFrequency = 18.0e9
Chan.StopFrequency = 20.0e9
Chan.IFBandwidth = 1000
' Begin a guided calibrations
Parser.Parse "SENS:CORR:COLL:GUID:CONN:PORT1 'APC 3.5 male'"
Parser.Parse "SENS:CORR:COLL:GUID:CONN:PORT2 'APC 3.5 female'"
Parser.Parse "SENS:CORR:COLL:GUID:CKIT:PORT1 '85052C'"
Parser.Parse "SENS:CORR:COLL:GUID:CKIT:PORT2 '85052C'"
' Select TRL cal method.
Parser.Parse "SENS:CORR:COLL:GUID:PATH:CMET 1,2,'TRL'"
txtDat = Parser.Parse("SENS:CORR:COLL:GUID:PATH:CMET? 1,2")
MsgBox("Method " + txtDat)
Parser.Parse "SENS:CORR:COLL:GUID:INIT"
' Query the number of steps
txtDat = Parser.Parse("SENS:CORR:COLL:GUID:STEP?")
' Display the number of steps
MsgBox("Number of steps is " + txtDat)
' Set the loop counter limit
step = CInt(txtDat)
' Measure the standards
For i = 1 To step
parserTxt = "sens:corr:coll:guid:desc? " + CStr(i)
prompt = Parser.Parse(parserTxt)
MsgBox(prompt)
parserTxt = "sens:corr:coll:guid:acq STAN" + CStr(i)
Parser.Parse (parserTxt)
Next
' All standards have been measured. Save the result
Parser.Parse "SENS:CORR:COLL:GUID:SAVE"
MsgBox("The TRL calibration has been completed")

```

Last Modified:

9-Apr-2007    MX Updated for new CMethod command



## Perform Unknown Thru or TRL Cal

---

The following program performs either a 2-port SOLT Unknown Thru Cal or a 2-port TRL Cal. The 85052C Cal Kit used in this program contains both types of standards. This program can be run on 2-port or 4-port PNAs. When run on a multiport (4 or more ports) PNA, which does not have [a reference receiver per port](#), a [Delta Match Cal](#) is required.

See [example of Delta Match Cal](#).

[See the Guided Cal commands](#)

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unknown.vbs. [Learn how to setup and run the macro](#).

```
Sub PerformUnknownThruOrTRLCal()
Set pna = CreateObject("AgilentPNA835x.Application")
Set scpi = pna.ScpiStringParser

' Specify connectors for Ports 1 and 2
scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT1 'APC 3.5 female'"
scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT2 'APC 3.5 male'"

'If your PNA has 3 or 4 ports, uncomment one or both of
'these next two lines, to explicitly specify this is a 2-port cal.
'scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT3 'Not used'"
'scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT4 'Not used'"

' Specify cal kit for Ports 1 and 2
scpi.Parse "SENS:CORR:COLL:GUID:CKIT:PORT1 '85052C'"
scpi.Parse "SENS:CORR:COLL:GUID:CKIT:PORT2 '85052C'"
' Since the 85052C cal kit contains SOLT standards and also TRL
' standards, these next two lines set cal and thru method
scpi.Execute "SENS:CORR:COLL:GUID:PATH:CMETHOD 1,2,"SOLT"
scpi.Execute "SENS:CORR:COLL:GUID:PATH:TMETHOD 1,2,"UNKN"

' To set up the cal as TRL, comment the previous line and uncomment
' this next line. The TMETHOD is set by default
'scpi.Execute "SENS:CORR:COLL:GUID:PATH:CMETHOD 1,2,"TRL"

' Initiate the calibration
scpi.Parse "SENS:CORR:COLL:GUID:INIT"

' Query the list of ports that need delta match
retStr = scpi.Parse("SENS:CORR:COLL:GUID:DMAT:APPL:PORT?")
portList = Split(retStr, ",")

' If portList contains just one element and it's value is 0, then that indicates
' none of the ports being calibrated require delta match data.
' Note: if each testport on the PNA has it's own reference receiver (R channel),
' then delta match is never needed, so portList will always be just 0.
lowerBound = LBound(portList)
If (UBound(portList) <> lowerBound) Or (CInt( portList(lowerBound) ) <> 0) Then
```

```

' Delta match data is required for at least one port.
' For this example, we assume a Global Delta Match Cal has previously been
' performed so the Global Delta Match CalSet exists.
' The Global Delta Match CalSet is used when the APPL command is invoked
' without a specific calset ID (GUID).
    scpi.Parse "SENS:CORR:COLL:GUID:DMAT:APPL"
End If

' Query the number of calibration steps
retStr = scpi.Parse("SENS:CORR:COLL:GUID:STEP?")
numSteps = CInt(retStr)

' Measure the cal standards
For i = 1 To numSteps
    prompt = scpi.Parse("SENS:CORR:COLL:GUID:DESC? " & CStr(i))
    retVal = MsgBox(prompt, vbOKCancel)
    If retVal = vbCancel Then Exit Sub
    retStr = scpi.Parse("SENS:CORR:COLL:GUID:ACQ STAN" & CStr(i) & ";*OPC?")
Next

' Compute the error coefficients and save the cal to CalSet, and turn it on
scpi.Parse "SENS:CORR:COLL:GUID:SAVE"
MsgBox "Cal is done!"
End Sub

```

---

Last Modified:

14-May-2007    MX Updated for new CMET and TMET commands

## Perform Global Delta Match Cal

---

The following program performs a [Global Delta Match Calibration](#). This is required when performing an Unknown Thru Cal or TRL Cal on PNAs without a [reference receiver for each test port](#). [See example of Unknown Thru or TRL Cal](#).

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Delta.vbs. [Learn how to setup and run the macro](#).

```
Sub PerformGlobalDeltaMatchCal()  
Set pna = CreateObject("AgilentPNA835x.Application")  
Set scpi = pna.ScpiStringParser  
  
' Initiate a Global Delta Match calibration, choosing connector and cal kit  
scpi.Parse "SENS:CORR:COLL:GUID:DMAT 'APC 3.5 female', '85033D/E'"  
  
' Query the number of calibration steps  
retStr = scpi.Parse("SENS:CORR:COLL:GUID:STEP?")  
numSteps = CInt(retStr)  
  
' Measure the cal standards  
For i = 1 To numSteps  
    prompt = scpi.Parse("SENS:CORR:COLL:GUID:DESC? " & CStr(i))  
    retVal = MsgBox(prompt, vbOKCancel)  
    If retVal = vbCancel Then Exit Sub  
    retStr = scpi.Parse("SENS:CORR:COLL:GUID:ACQ STAN" & CStr(i) & ";*OPC?")  
Next  
  
' Compute the error coefficients and save the cal to Global Delta Match CalSet  
scpi.Parse "SENS:CORR:COLL:GUID:SAVE"  
MsgBox "Cal is done!"  
End Sub
```

## Perform an Unguided ECal

---

This VBScript program performs an Unguided Full 2-Port ECal.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unguided.vbs. [Learn how to setup and run the macro.](#)

[See Sense:Correction commands.](#)

### [See other SCPI Examples](#)

```
Set pna = CreateObject("AgilentPNA835x.Application")
Set scpi = pna.ScpStringParser
' Preset the analyzer
scpi.Execute "SYSTem:PRESet"

' Start frequency of 10 MHz
scpi.Execute "SENSe:FREQuency:START 10E6"

' Stop frequency of 9 GHz
scpi.Execute "SENSe:FREQuency:STOP 9E9"

' Select the preset S11 measurement
scpi.Execute "CALCulate:PARAMeter:SElect 'CH1_S11_1'"
' Read the information about the Agilent factory
' characterization data of ECal module #1 on the USB bus
module1Info = scpi.Execute("SENSe:CORRection:COLlect:CKIT:INFormation? ECAL1,CHAR0")

' Prompt for the ECal module
MsgBox "Description of ECal Module #1:" & Chr(10) & Chr(10) & module1Info & _Chr(10)
& Chr(10) & "Make port connections to the ECal module, then press enter"
' ECal full 1 port and 2 port
' Choose a Calibration Type (comment out one of these)
scpi.Execute "SENSe:CORRection:COLlect:METHod refl3"
scpi.Execute "SENSe:CORRection:COLlect:METHod SPARSOLT"
' Specify to have the PNA automatically determine which port of the
' ECal module is connected to which port of the PNA.
scpi.Execute "SENSe:CORRection:PREference:ECAL:ORientation ON"
' Alternatively, if you are measuring at very low power levels where
' the PNA fails to sense the module's orientation, you may need to turn
' off the auto orientation and specify how the module is connected (as in
' these next two commented lines of code -- "A1,B2" would indicate Port A
' of the module is connected to Port 1 and Port B is connected to Port 2).
'scpi.Execute "SENSe:CORRection:PREference:ECAL:ORientation OFF"
'scpi.Execute "SENSe:CORRection:PREference:ECAL:PMAP ECAL1,'A1,B2'"
' Acquire and store the calibration terms. *OPC? causes a "+1" to be
```

```

' returned when finished.  CHAR0 indicates to use the Agilent factory
' characterized data within the ECal module (as opposed to a user characterization).
x = scpi.Execute("SENSe:CORRection:COLLect:ACQuire ECAL1,CHAR0;*OPC?")
' Note: if you have set up a slow sweep speed (for example, if
' you're using a narrow IF bandwidth), and while this calibration is
' being acquired you wish to have your program perform other operations
' (like checking for the click event of a Cancel button) and you're
' NOT using the COM ScpiStringParser, you can use the optional
' ASYNchronous argument with the ACQuire command as shown here below
' instead of sending that command in the way shown above. The SCPI
' parser then will return immediately while the cal acquisition
' proceeds (i.e., the parser will NOT block-and-wait for the
' cal to finish, so you can send additional commands in the meantime).
' So you can do "*ESR?" or "*STB?" queries to monitor the status register
' bytes to see when the OPC bit gets set, which indicates the cal has
' finished. That type of OPC detection works for all of the PNA's SCPI
' parsers except the COM ScpiStringParser.
' An alternative to querying the status register, is to setup an SRQ handler
' if your IO Libraries supports that.
' When an SRQ event occurs, a call back will automatically
' "SENSe:CORRection:COLLect:ACQuire ECAL1,CHAR0,ASYNchronous;*OPC"
MsgBox "Done with calibration."

```

## Perform an Unguided 2-Port Mechanical Cal

---

This VBScript program performs an Unguided, Full 2-Port, calibration using ONE set of mechanical calibration standards.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unguided.vbs. [Learn how to setup and run the macro.](#)

```
Set App = CreateObject("AgilentPNA835x.Application")
Set Scpi = App.SCPIStringParser

'Initialize state
Scpi.Execute ("SYSTem:PRESet")

'Select the Preset measurement
Scpi.Execute ("CALCulate:PARAmeter:SElect 'CH1_S11_1'")

'Set the calibration method
Scpi.Execute ("SENSe:CORRection:COLLect:METHod SPARSOLT")

'Select a cal kit
Scpi.Execute ("SENSe:CORRection:COLLect:CKIT:SElect 1")

'Set one set of standards
Scpi.Execute ("SENSe:CORRection:TSTandards OFF")

'Set acquisition to FORWARD
Scpi.Execute ("SENSe:CORRection:SFORward ON")

'Measure the standards in forward direction
MsgBox "Connect OPEN to Port 1; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan1")

MsgBox "Connect SHORT to Port 1; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan2")

MsgBox "Connect LOAD to Port 1; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan3")

'Set acquisition to REVERSE
Scpi.Execute ("SENSe:CORRection:SFORward OFF")

'Measure the standards in reverse direction
MsgBox "Connect OPEN to Port 2; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan1")

MsgBox "Connect SHORT to Port 2; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan2")
```

```
MsgBox "Connect LOAD to Port 2; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan3")

'Measure the thru standard
MsgBox "Connect THRU between Ports 1 and 2; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan4")

'OPTIONAL Measure Isolation
MsgBox "Connect LOADS to Port 1 AND Port 2; then press OK"
Scpi.Execute ("SENSe:CORRection:COLLect:ACQuire stan5")

'All standards have been measured. Save the result
Scpi.Execute ("SENS:CORR:COLL:SAVE")
MsgBox "The calibration has been completed"
```

## Perform an Unguided 1-Port Cal on Port 2

---

This VBScript program does the following:

1. Clear measurements from the PNA
2. Create a new S22 measurement
3. Set an instrument state
4. Select a cal kit
5. Initiate an Unguided calibration
6. Display a prompt to connect each standard
7. Save the calibration to a newly created cal set

**Note:** This example illustrates an important step when calibrating a reflection measurement in the reverse direction. You MUST create a reverse (S22) measurement and have it be the active (selected) measurement on the channel that is being calibrated. This is not necessary for any calibrating any other measurement parameter.

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unguided.vbs. [Learn how to setup and run the macro.](#)

```
Dim App
Set App = CreateObject("AgilentPNA835x.Application")
App.Preset

Dim Parser
Dim Chan

Rem Clear old measurements
App.Reset

Rem Create a new Measurement
Set Parser = App.SCPIStringParser
Parser.Parse "DISPlay:WINDow1:STATE ON"
Parser.Parse "CALCulate:PARAMeter:DEFine 'MyMeas',S22"
Parser.Parse "DISPlay:WINDow1:TRACe1:FEED 'MyMeas' "

Rem Initialize state
Set Chan = App.ActiveChannel
Chan.StartFrequency = 200e6
Chan.StopFrequency = 1.5e9
Chan.IFBandwidth = 1000

Rem Begin an unguided calibration
```



```
Rem Set the calibration method
Parser.Parse "SENSe:CORRection:COLLect:METhod REFL3"

Rem Turn off continuous sweep
Parser.Parse "INITiate:CONTinuous OFF"

Rem Select a cal kit
Parser.Parse "SENSe:CORRection:COLLect:CKIT:SElect 1"

Rem Measure the standards
MsgBox("Connect OPEN to port 2. Then press OK")
Parser.Parse ("sens:corr:coll:acq STAN1")

MsgBox("Connect SHORT to port 2. Then press OK")
Parser.Parse ("sens:corr:coll:acq STAN2")

MsgBox("Connect LOAD to port 2. Then press OK")
Parser.Parse ("sens:corr:coll:acq STAN3")

Rem All standards have been measured. Save the result
Parser.Parse "SENS:CORR:COLL:SAVE"

Rem Turn ON continuous sweep
Parser.Parse "INITiate:CONTinuous ON"
MsgBox("The calibration has been completed")
```

## Perform an Unguided Cal on a 4-Port PNA

---

This topic describes how to perform an unguided calibration on a multiport network analyzer using SCPI. The objective here is to make clear the relationship between the physical port on which a standard is being measured, the actual device in the cal kit, and the SCPI command used to acquire the device.

There are two sets of SCPI commands that acquire calibrations. One set is used for guided cal, the other for unguided. The SCPI commands that provide remote access to unguided cal are in the SENS:CORR:COLL block:

- SENS:CORR:COLL:METHod
- SENS:CORR:COLL:ACQuire
- SENS:CORR:COLL:SAVE

On a four port network analyzer, the remote programmer needs to be aware of the relationship between the physical port and the calibration kit class assignments. The example program (below) illustrates the usage by performing three unique 2 port cals, taking care to acquire the appropriate standards.

Calibration standards classes are 'categories' of standard types. To perform a 2 port calibration, the cal wizard requires the user to measure:

### **3 reflection standards on the forward port:**

- Class S11A typically an open
- Class S11B typically a short
- Class S11C typically a load

### **Likewise, 3 reflection standards are required for the reverse port:**

- Class S22A typically an open
- Class S22B typically a short
- Class S22C typically a load

### **There is also a transmission standard that is measured in both directions:**

- Class S21T typically a thru

The following illustrates the relationship between cal kit physical standards and calibration classes.

Here is a list of the physical devices in my calibration kit.

Standard #1 = "3.5 mm male short"

Standard #2 = "3.5 mm male open"

Standard #3 = "3.5 mm male broadband load"

Standard #4 = "Insertable thru standard"

Standard #5 = "3.5 mm male sliding load"

Standard #6 = "3.5 mm male lowband load"

Standard #7 = "3.5 mm female short"

Standard #8 = "female to female characterized thru adapter"

Standard #9 = "0-2 Load"

Standard #10 = "Open"

Standard #11 = "Non-insertable thru"

Standard #12 = "3.5 mm female lowband load"

Standard #13 = "3.5 mm female sliding load"

Standard #14 = "3.5 mm female broadband load"

Standard #15 = "3.5 mm female open"

When you perform a calibration remotely using SCPI, you don't specify the device number directly. Rather, you specify the class you want to measure. Each device in the calibration kit is assigned to a class. And since more than one device can be assigned to the same class, each class contains an ordered list of devices. The class assignments are user-settable using the Advanced Modify Cal Kit dialog or the SCPI command:

[SENS:CORR:COLL:CKIT:ORDER](#)<class>, <std>, <std>, <std>, <std>, <std>, <std>, <std>

The 85052B kit used in the example program had the following standard list for each class: The list was obtained by issuing the corresponding SCPI query:

[SENS:CORR:COLL:CKIT:OLIST1?](#) S11A = +2,+15,+0,+0,+0,+0,+0

[SENS:CORR:COLL:CKIT:OLIST2?](#) S11B = +1,+7,+0,+0,+0,+0,+0

[SENS:CORR:COLL:CKIT:OLIST3?](#) S11C = +6,+5,+3,+12,+13,+14,+0

[SENS:CORR:COLL:CKIT:OLIST4?](#) S21T = +4,+8,+0,+0,+0,+0,+0

[SENS:CORR:COLL:CKIT:OLIST5?](#) S22A = +2,+15,+0,+0,+0,+0,+0

[SENS:CORR:COLL:CKIT:OLIST6?](#) S22B = +1,+7,+0,+0,+0,+0,+0

[SENS:CORR:COLL:CKIT:OLIST7?](#) S22C = +6,+5,+3,+12,+13,+14,+0

[SENS:CORR:COLL:CKIT:OLIST8?](#) S12T = +4,+8,+0,+0,+0,+0,+0

When you perform the calibration, you acquire data by issuing the ACQUIRE command:

[SENS:CORR:COLL:ACQ](#) <class>[, <subst> ]

For example:

[SENS:CORR:COLL:SFOR](#) 1

[SENS:CORR:COLL:ACQ](#) STANA, SST2

The SFOR command tells the wizard to make the next acquisition in the forward direction. The ACQUIRE command specifies that we are measuring the 2nd device in the list for STANA. And since we are measuring SFORward, STANA refers to class #1 or S11A. The list of devices for this class are specified in the OLIST1 query above. The associations are shown in red.

Alternately, you could modify the device order for the S11A class to move device #15 into the first position ([SENS:CORR:COLL:CKIT:ORDER1](#)). When the desired device is in the first position, you needn't specify the order number in the ACQUIRE command. The default is the first device in the OLIST. This worked well for two port network analyzers where the order for S11A,B,C classes were setup for port 1 and the order for S22A,B,C was set up for port 2. With the kit setup in the proper order, you could eliminate the specification of the substandard

number (SST<n>).

When performing 2 port calibrations on 4 Port Network Analyzers (e.g.: PNA Model N5230A), the wizard applies S11A,B,C standards to the lower numbered port, S22A,B,C standards to the higher numbered port. Since the two classes (S11A,B,C and S22A,B,C) are applied to multiple ports, the programmer must take into account the ports being measured and take greater care when specifying the ACQUIRE command to ensure that the correct device is being measured.

**Port to class relationship**

Ports	S11A Port	S22A Port
1,2	1	2
1,3	1	3
1,4	1	4
2,3	2	3
2,4	2	4
3,4	3	4

The following example program shows one method of handling two port cals on a multiport network analyzer. The connectors at the measurement plane are assumed to be (1) male, (2) female, (3) male, and (4) male. In the example, three cals are performed: 1-2 (insertable male to female), 2-3 (insertable female to male), and 3-4 (noninsertable using an characterized adapter).

```
option explicit
public scpi
public pna
' assume a 4 port PNA with the following connectors:
' the standard measured on these ports will be the opposite gender
' PORT 1 = 3.5 male
' PORT 2 = 3.5 female
' PORT 3 = 3.5 male
' PORT 4 = 3.5 male
'To perform 2 port calibrations between 1-2, 2-3, and 3-4 you need to do the
following

call main

sub main
set pna = CreateObject("AgilentPnA835x.Application")
set scpi = pna.ScpStringParser
pna.Preset
' select a kit to use for this demonstration
' kit #1 for the N5230A is the 85052B 3.5mm kit with sliding load
scpi.execute("SENS:CORR:COLL:CKIT:SELECT 1" )
```

```

PrintKitStandardInfo 1
PrintKitOlist 1

' -----
'   CALIBRATE PORTS 1 and 2, insertable cal
' -----

wscript.echo
wscript.echo "Calibrating ports 1 and 2"
scpi.execute("SYST:PRES;")
scpi.execute("calc:par:sel CH1_S11_1")
scpi.execute("SENS:CORR:TST:STATE 0")
scpi.execute("SENS:CORR:COLL:METHod SPARSOLT")
scpi.execute("SENS:CORR:SFOR 1")
MeasureFemaleStandards 1
scpi.execute("SENS:CORR:SFOR 0")
MeasureMaleStandards 2
MeasureTransmissionStandards 1,2
scpi.execute("SENS:CORR:COLL:SAVE")

' -----
'   CALIBRATE PORTS 2 and 3, insertable cal
' -----

wscript.echo
wscript.echo "Calibrating ports 2 and 3"
scpi.execute("SYST:PRES;")
scpi.execute("calc:par:sel CH1_S11_1")
scpi.execute("calc:par:mod S23")
scpi.execute("SENS:CORR:TST:STATE 0")
scpi.execute("SENS:CORR:COLL:METHod SPARSOLT")
scpi.execute("SENS:CORR:SFOR 1")
MeasureMaleStandards 2
scpi.execute("SENS:CORR:SFOR 0")
MeasureFemaleStandards 3
MeasureTransmissionStandards 2,3
scpi.execute("SENS:CORR:COLL:SAVE")

' -----
'   CALIBRATE PORTS 3 and 4, non-insertable cal
' -----

wscript.echo
wscript.echo "Calibrating ports 3 and 4"
scpi.execute("SYST:PRES;")
scpi.execute("calc:par:sel CH1_S11_1")
scpi.execute("calc:par:mod S43")
scpi.execute("SENS:CORR:COLL:METHod SPARSOLT")
scpi.execute("SENS:CORR:SFOR 1")
MeasureFemaleStandards 3
scpi.execute("SENS:CORR:SFOR 0")
MeasureFemaleStandards 4
MeasureAdapter 3, 4
scpi.execute("SENS:CORR:COLL:SAVE")
end sub

```

```

sub MeasureMaleStandards ( portNumber )
dim portstr
portstr = formatnumber(portNumber,0)
Promptconnect1 1, 1, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN1;*OPC?")

Promptconnect1 2, 1, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN2;*OPC?")
Promptconnect1 3, 3, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN3,SST3;*OPC?")
end sub

sub MeasureFemaleStandards (portNumber)
dim portstr
portstr = formatnumber(portNumber,0)
Promptconnect1 1, 2, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN1,SST2;*OPC?")
Promptconnect1 2, 2, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN2,SST2;*OPC?")
Promptconnect1 3, 6, portNumber
scpi.execute("SENS:CORR:COLL:ACQ STAN3,SST6;*OPC?")
end sub

sub MeasureTransmissionStandards( port1, port2)
dim p1str
dim p2str
p1str = formatnumber( port1, 0)
p2str = formatnumber( port2, 0)

Promptconnect2 4, 1, port1, port2
scpi.execute("SENS:CORR:COLL:ACQ STAN4;*OPC?")
end sub

sub MeasureAdapter( port1, port2)
dim p1str
dim p2str
p1str = formatnumber( port1, 0)
p2str = formatnumber( port2, 0)

Promptconnect2 4, 2, port1, port2
scpi.execute("SENS:CORR:COLL:ACQ STAN4,SST2;*OPC?")
end sub

' return the nth item in the comma separated list
Function GetItemNumber( list, n)
dim strVector
strVector = split(list,",",-1,1)
GetItemNumber = strVector(n-1)
end function

' remove the trailing newline from str
function chop( str )

```

```

dim tmp
tmp = str
' remove the appended newline
dim pos
pos = InStrRev(tmp,vbCrLf)
if (pos >0) then
tmp = mid(tmp,1,pos-1)
end if
chop = tmp
end function

'return the label for the nth standard assigned to the class described by
class_index.
' if class_index = 1, class is S11A (STAN1)
' if class_index = 2, class is S11B (STAN2), etc
function GetStandardLabel( class_index, nth)
dim olist
dim stdnum
dim resp
olist = scpi.execute("SENS:CORR:COLL:CKIT:OLIST" + formatnumber(class_index,0)+"?")
stdnum = GetItemNumber( olist, nth)
scpi.execute("SENS:CORR:COLL:CKIT:STAN " + formatnumber(stdnum,0))
resp = scpi.execute("SENS:CORR:COLL:CKIT:STAN:LABEL?")
GetStandardLabel = chop(resp)
end function

sub PromptConnect1( class_index, nth, port)
wscript.echo "CONNECT " + GetStandardLabel( class_index, nth) + " to port " +
formatnumber(port,0)
end sub

sub PromptConnect2( class_index, nth, port1, port2)
wscript.echo "CONNECT " + GetStandardLabel( class_index, nth) + " between ports " +
formatnumber(port1,0) + " and " + formatnumber(port2,0)
end sub

' Print the order of standards per class for this kit
sub PrintKitOlist( kit )
dim i
dim cmd
dim resp
wscript.echo
dim olistcmd
olistcmd = "SENS:CORR:COLL:CKIT:OLIST"
' list the sub standards for each of the following classes
' S11A, S11B, S11C, FWD TRANS, FWD ISOL, S22A, S22B, S22C, REV TRANS, REV ISOL
for i = 1 to 8
cmd = olistcmd + formatNumber(i,0) + "?"
resp = scpi.execute(cmd)
wscript.echo cmd + "= " + chop(resp)
next
end sub

```

```

sub PrintKitStandardInfo( kit )
wscript.echo scpi.execute("SENS:CORR:COLL:CKIT:NAME?")
dim i
for i = 1 to 30
dim slabel
dim snum
snum = formatNumber(i,0)
scpi.execute("SENS:CORR:COLL:CKIT:STAN " + snum)
slabel=scpi.execute("SENS:CORR:COLL:CKIT:STAN:LABel?")
wscript.echo "Standard #" + snum + " = " + chop(slabel)
next
end sub

```

The output from this program is as follows:

Microsoft (R) Windows Script Host Version 5.6

Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

"85052B 3.5 mm with sliding load"

Standard #1 = "3.5 mm male short"

Standard #2 = "3.5 mm male open"

Standard #3 = "3.5 mm male broadband load"

Standard #4 = "Insertable thru standard"

Standard #5 = "3.5 mm male sliding load"

Standard #6 = "3.5 mm male lowband load"

Standard #7 = "3.5 mm female short"

Standard #8 = "female to female characterized thru adapter"

Standard #9 = "0-2 Load"

Standard #10 = "Open"

Standard #11 = "Non-insertable thru"

Standard #12 = "3.5 mm female lowband load"

Standard #13 = "3.5 mm female sliding load"

Standard #14 = "3.5 mm female broadband load"

Standard #15 = "3.5 mm female open"

Standard #16 = "Open"

Standard #17 = "Open"

Standard #18 = "Open"

Standard #19 = "Open"

Standard #20 = "Open"

Standard #21 = "Open"



Standard #22 = "Open"

Standard #23 = "Open"

Standard #24 = "Open"

Standard #25 = "Open"

Standard #26 = "Open"

Standard #27 = "Open"

Standard #28 = "Open"

Standard #29 = "Open"

Standard #30 = "Open"

SENS:CORR:COLL:CKIT:OLIST1?= +2,+15,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST2?= +1,+7,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST3?= +6,+5,+3,+12,+13,+14,+0

SENS:CORR:COLL:CKIT:OLIST4?= +4,+8,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST5?= +2,+15,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST6?= +1,+7,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST7?= +6,+5,+3,+12,+13,+14,+0

SENS:CORR:COLL:CKIT:OLIST8?= +4,+8,+0,+0,+0,+0,+0

Calibrating ports 1 and 2

CONNECT "3.5 mm female open" to port 1

CONNECT "3.5 mm female short" to port 1

CONNECT "3.5 mm female broadband load" to port 1

CONNECT "3.5 mm male open" to port 2

CONNECT "3.5 mm male short" to port 2

CONNECT "3.5 mm male broadband load" to port 2

CONNECT "Insertable thru standard" between ports 1 and 2

Calibrating ports 2 and 3

CONNECT "3.5 mm male open" to port 2

CONNECT "3.5 mm male short" to port 2

CONNECT "3.5 mm male broadband load" to port 2

CONNECT "3.5 mm female open" to port 3

CONNECT "3.5 mm female short" to port 3

CONNECT "3.5 mm female broadband load" to port 3

CONNECT "Insertable thru standard" between ports 2 and 3

Calibrating ports 3 and 4

CONNECT "3.5 mm female open" to port 3

CONNECT "3.5 mm female short" to port 3

CONNECT "3.5 mm female broadband load" to port 3

CONNECT "3.5 mm female open" to port 4

CONNECT "3.5 mm female short" to port 4

CONNECT "3.5 mm female broadband load" to port 4

CONNECT "female to female characterized thru adapter" between ports 3 and 4

## Perform an Unguided Cal on Multiple Channels

---

This VBScript program performs an Unguided Calibration simultaneously on two channels.

This could be used in the following cases:

- If you need more than the current number of data points per trace, so the additional points must be added to a different channel.
- If you need several channels with independent settings, but you want to calibrate all channels with a minimal number of standard connections. This would be especially critical for on wafer calibration.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as Unguided.vbs. [Learn how to setup and run the macro.](#)

```
Dim app
Dim scpi
Dim NumberOfActiveChannels
NumberOfActiveChannels = 2
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
' Query the list of connectors that the PNA system recognizes
scpi.Execute("SYST:PRES")
'Wait for successful preset before continuing
done=scpi.Execute("*OPC?")
'The following section sets up 2 channels with different frequency ranges
scpi.Execute("DISP:WIND1:STATE OFF")
'Reset Windows
scpi.Execute("DISP:WIND1:STATE ON")
scpi.Execute("DISP:WIND2:STATE ON")
'
' Assign a measurement to the first window
scpi.Execute("CALC1:PAR:DEF 'Meas1', S21")
scpi.Execute("DISP:WIND1:TRAC1:FEED 'Meas1'")
'Assign a measurement to the second window
scpi.Execute("CALC2:PAR:DEF 'Meas2', S21")
scpi.Execute("DISP:WIND2:TRAC1:FEED 'Meas2'")
```

```

'Set up two channels with independent parameters
scpi.Execute("SENS1:FREQ:SPAN 1e9")
scpi.Execute("SENS2:FREQ:SPAN 1e6")
'Wait for changes before continuing
done=scpi.Execute("*OPC?")
'

'This section sets the calibration kits for channel 1 and channel 2
'Select a trace from channel 1 and set calibration type and cal kit
scpi.Execute("CALC1:PAR:SEL 'Meas1'")
scpi.Execute("SENS1:CORR:COLL:METH SPARSOLT")
scpi.Execute("SENS1:CORR:COLL:CKIT 2") '85056D for default settings
'Same standards for forward and reverse direction
scpi.Execute("SENS1:CORR:TST OFF")
'Select a trace from channel 2 and set calibration type and cal kit
scpi.Execute("CALC2:PAR:SEL 'Meas2'")
scpi.Execute("SENS2:CORR:COLL:METH SPARSOLT")
scpi.Execute("SENS2:CORR:COLL:CKIT 2") '85056D for default settings
'Same standards for forward and reverse direction
scpi.Execute("SENS2:CORR:TST OFF")

'Set both channels to manual triggering
scpi.Execute("INIT1:CONT OFF")
scpi.Execute("INIT2:CONT OFF")
'

'The following assumes female port connector on port 1
' and male port connector on port 1
'Step through all active channels and calibrate and measure all standards.
scpi.Execute("SENS1:CORR:SFOR ON") 'Set acquisition to forward
scpi.Execute("SENS2:CORR:SFOR ON") 'Set acquisition to forward
MsgBox("Connect OPEN standard to port 1")
For CurrentChannel = 1 To NumberOfActiveChannels
scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan1")
done= scpi.Execute("*OPC?")
Next

```

```

MsgBox("Connect SHORT standard to port 1")
For CurrentChannel = 1 To NumberOfActiveChannels
    scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
    scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan2")
    done=scpi.Execute("*OPC?")
Next

MsgBox("Connect LOAD standard to port 1")
For CurrentChannel = 1 To NumberOfActiveChannels
    scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
    scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan3")
    done=scpi.Execute("*OPC?")
Next

scpi.Execute("SENS1:CORR:SFOR OFF") 'Set acquisition to reverse
scpi.Execute("SENS2:CORR:SFOR OFF") 'Set acquisition to forward

MsgBox("Connect OPEN standard to port 2")
For CurrentChannel = 1 To NumberOfActiveChannels
    scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
    scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan1")
    done=scpi.Execute("*OPC?")
Next

MsgBox("Connect SHORT standard to port 2")
For CurrentChannel = 1 To NumberOfActiveChannels
    scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
    scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan2")
    done=scpi.Execute("*OPC?")
Next

MsgBox("Connect LOAD standard to port 2")
For CurrentChannel = 1 To NumberOfActiveChannels
    scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
    scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan3")
    done=scpi.Execute("*OPC?")
Next

```

```

'
'Measure thru standard for all channels in both forward and reverse direction
MsgBox("Connect THRU between ports 1 and 2")
scpi.Execute("SENS1:CORR:SFOR ON") 'Set acquisition to forward
scpi.Execute("SENS2:CORR:SFOR ON") 'Set acquisition to forward
For CurrentChannel = 1 To NumberOfActiveChannels
scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan4")
done=scpi.Execute("*OPC?")
Next

scpi.Execute("SENS1:CORR:SFOR OFF") 'Set acquisition to reverse
scpi.Execute("SENS2:CORR:SFOR OFF") 'Set acquisition to reverse
For CurrentChannel = 1 To NumberOfActiveChannels
scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL stan4")
done=scpi.Execute("*OPC?")
Next

For CurrentChannel = 1 To NumberOfActiveChannels
scpi.Execute("CALC" & CurrentChannel & ":PAR:SEL 'Meas" & CurrentChannel & "'")
scpi.Execute("SENS" & CurrentChannel & ":CORR:COLL:SAVE")
done=scpi.Execute("*OPC?")
Next

'Set both channels to continuous triggering
scpi.Execute("INIT1:CONT ON")
scpi.Execute("INIT2:CONT ON")

```

## ECALConfidence Check using SCPI

---

This Visual Basic program performs a complete ECAL confidence check.

To run this program, you need:

- An established GPIB interface connection
- Agilent's VISA or National Instrument's VISA installed on your PC
- The module visa32.bas added to your VB project.
- A form with two buttons: cmdRun and cmdQuit
- A calibrated S11 1-port or N-port measurement active on Channel 1
- Window 1 is visible

[See Other SCPI Example Programs](#)

```
'Session to VISA Default Resource Manager
Private defRM As Long
'Session to PNA
Private viPNA As Long
'VISA function status return code
Private status As Long

Private Sub Form_Load()
    defRM = 0
End Sub

Private Sub cmdRun_Click()
'String to receive data from the PNA
Dim strReply As String * 200

' Open the VISA default resource manager
status = viOpenDefaultRM(defRM)
If (status < VI_SUCCESS) Then HandleVISAError

' Open a VISA session (viPNA) to the PNA at GPIB address 16.
status = viOpen(defRM, "GPIB0::16::INSTR", 0, 0, viPNA)
If (status < VI_SUCCESS) Then HandleVISAError

' Need to set the VISA timeout value to give all our GPIB Reads
' sufficient time to complete before a timeout error occurs.
' For this example, let's try setting the limit to
' 10000 milliseconds (10 seconds).
status = viSetAttribute(viPNA, VI_ATTR_TMO_VALUE, 10000)
If (status < VI_SUCCESS) Then HandleVISAError

' Get the catalog of all the measurements currently on Channel 1.
status = myGPIBWrite(viPNA, "CALC1:PAR:CAT?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError
```

```

' If an S11 measurement named "MY_S11" doesn't already exist,
' then create it.
If Instr(strReply, "MY_S11") = 0 Then
    status = myGPIBWrite(viPNA, "CALC1:PAR:DEF MY_S11,S11")
    If (status < VI_SUCCESS) Then HandleVISAError
End If
strReply = ""

' Get the catalog of all the trace numbers currently active
' in Window 1.
status = myGPIBWrite(viPNA, "DISP:WIND1:CAT?")
If (status < VI_SUCCESS) Then HandleVISAError

status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError

' If a trace number 4 already exists in Window 1, then this
' will remove it.
If Instr(strReply, "4") > 0 Then
    status = myGPIBWrite(viPNA, "DISP:WIND1:TRAC4:DEL")
    If (status < VI_SUCCESS) Then HandleVISAError
End If

' Set trace number 4 to MY_S11.
status = myGPIBWrite(viPNA, "DISP:WIND1:TRAC4:FEED MY_S11")
If (status < VI_SUCCESS) Then HandleVISAError

' Set up trace view so we are viewing only the data trace.
status = myGPIBWrite(viPNA, "DISP:WIND1:TRAC4 ON")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBWrite(viPNA, "DISP:WIND1:TRAC4:MEM OFF")
If (status < VI_SUCCESS) Then HandleVISAError

' Select MY_S11 as the measurement to be used for the
' Confidence Check.
status = myGPIBWrite(viPNA, "SENS1:CORR:CCH:PAR MY_S11")
If (status < VI_SUCCESS) Then HandleVISAError

' Acquire the S11 confidence check data from ECal Module A
' into the memory buffer (asking for an OPC reply when it's done).
status = myGPIBWrite(viPNA, "SENS1:CORR:CCH:ACQ ECAL1;*OPC?")
If (status < VI_SUCCESS) Then HandleVISAError

' The PNA sends an OPC reply ("1") when the confidence data
' acquisition into memory is complete, so this Read is waiting on
' the reply until it is received.
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError

' Turn on trace math so the trace shows data divided by memory.
' You can be confident the S11 calibration is reasonably good if

```



```

' the displayed trace varies no more than a few tenths of a dB
' from 0 dB across the entire span.
status = myGPIBWrite(viPNA, "CALC1:PAR:SEL MY_S11")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBWrite(viPNA, "CALC1:MATH:FUNC DIV")
If (status < VI_SUCCESS) Then HandleVISAError
End Sub

Private Sub cmdQuit_Click()
' Turn off trace math
status = myGPIBWrite(viPNA, "CALC1:MATH:FUNC NORM")
If (status < VI_SUCCESS) Then HandleVISAError

' Conclude the confidence check to set the ECal module
' back to it's idle state.
status = myGPIBWrite(viPNA, "SENS1:CORR:CCH:DONE")
If (status < VI_SUCCESS) Then HandleVISAError

' Close the resource manager session (which also closes
' the session to the PNA).
If defRM <> 0 Then Call viClose(defRM)

' End the program
End
End Sub

Private Function myGPIBWrite(ByVal viHandle As Long, ByVal strOut As String) As Long
' The "+ Chr$(10)" appends an ASCII linefeed character to the output, for
' terminating the write transaction.
myGPIBWrite = viVPrintf(viHandle, strOut + Chr$(10), 0)
End Function

Private Function myGPIBRead(ByVal viHandle As Long, strIn As String) As Long
myGPIBRead = viVScanf(viHandle, "%t", strIn)
End Function

Sub HandleVISAError()
Dim strVisaErr As String * 200
Call viStatusDesc(defRM, status, strVisaErr)
MsgBox "*** Error : " + strVisaErr, vbExclamation
End
End Sub

```

## Perform a Source and Receiver Power Cal using SCPI

---

Programming the PNA using COM or using SICL/VISA over LAN (as in this example) leaves the PNA free to control GPIB devices as needed.

The first example, using Visual Basic, demonstrates the following:

- Performing a source power calibration of Port 2 for Channel 1.
- Reading the calibration data.

The second example performs a [Receiver Power Cal](#) using VBScript.

Learn more about [Power Calibrations](#).

See an example that [Uploads a Source Power Cal](#).

### Other SCPI Example Programs

To run this program, you need:

- One of the following power meters connected to the PNA through GPIB: E4416A, E4417A, E4418A/B, E4419A/B, 437B, 438A, EPM-441A, EPM-442A

**Note:** If your power meter is other than these, you can [create your own Power Meter Driver](#) using our template.

- Your PC and PNA both connected to a LAN (for communicating with each other).
- The SICL and VISA components of Agilent's I/O Libraries software installed on your PC (both are included when you install the software, unless you already have another vendor's VISA installed. Then specify Full SICL and VISA installation to overwrite the other vendor's VISA).
- The module visa32.bas added to your VB project.
- A form with one button labeled cmdRun.
- A VISA interface configured on your remote PC to control the PNA. This could be GPIB interface or a [VISA LAN Client](#).
- On the PNA connect a Thru cable from port 1 to port 2.

Note: The [SOURce:POWer:CORRection:COLlect:ACQuire](#) command, when used with a power meter, cannot be sent over the GPIB unless the power meter is connected to a different GPIB interface. See the alternative methods described in the command details.

```
'Session to VISA Default Resource Manager  
Private defRM As Long
```

```

'Session to PNA
Private viPNA As Long
'VISA function status return code
Private status As Long

Private Sub Form_Load()
    defRM = 0
End Sub

Private Sub cmdRun_Click()
    ' String to receive data from the PNA.
    ' Dimensioned large enough to receive scalar comma-delimited values
    ' for 21 frequency points (20 ASCII characters per point)
    Dim strReply As String * 420

    Dim strStimulus, strCalValue
    Dim strResult As String

    ' Open the VISA default resource manager
    status = viOpenDefaultRM(defRM)
    If (status < VI_SUCCESS) Then HandleVISAError

    ' Open a session (viPNA) to the PNA at "address 16" on the VISA
    ' interface configured as "GPIB1" on this PC. This could be a
    ' VISA LAN Client pointing to the SICL LAN Server on the PNA, or
    ' an actual GPIB interface on this PC connected to the PNA GPIB
    ' (in which case the power meter would need to be connected to a
    ' different GPIB interface on the PNA, such as the Agilent 82357A
    ' USB-to-GPIB) .
    status = viOpen(defRM, "GPIB0::16::INSTR", 0, 0, viPNA)
    If (status < VI_SUCCESS) Then HandleVISAError

    ' Set the number of sweep points to 21 on Channel 1.
    status = myGPIBWrite(viPNA, "SENS1:SWE:POIN 21")
    If (status < VI_SUCCESS) Then HandleVISAError

```

```

' Specify the GPIB address of the power meter
' that will be used in performing the calibration.
status = myGPIBWrite(viPNA, "SYST:COMM:GPIB:PMET:ADDR 13")
If (status < VI_SUCCESS) Then HandleVISAError

' Turn use of the loss table OFF (this assumes there is
' virtually no loss in the RF path to the power sensor
' due to a splitter, coupler or adapter).
status = myGPIBWrite(viPNA, "SOUR:POW:CORR:COLL:TABL:LOSS OFF")
If (status < VI_SUCCESS) Then HandleVISAError

' Turn frequency checking OFF (so one power sensor is used for the entire cal
' acquisition sweep regardless of frequency span).
status = myGPIBWrite(viPNA, "SOUR:POW:CORR:COLL:FCH OFF")
If (status < VI_SUCCESS) Then HandleVISAError

' Specify a nominal power accuracy tolerance (NTolerance) in dB for the
calibration,
' and the maximum number (COUNT) of iterations to adjust power at each point,
' attempting to achieve within tolerance of the desired power. If at any
stimulus
' point the power fails to reach within the set tolerance of the desired power
' after the maximum number of iterations, the power at that point will be set to
the
' value determined by the last iteration (the Source Power Cal dialog box will
' indicate the FAIL, but we can still apply the cal if desired when it's
complete).
' Each iteration is based upon a SETTLED power reading (see comments preceding
the
' next two commands below).
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:ITER:NTOL 0.1")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:ITER:COUN 3")
If (status < VI_SUCCESS) Then HandleVISAError

' The worst-case window of power uncertainty (for a calibration which meets
' tolerance) is the sum of the iteration tolerance and the power meter settling
' tolerance (which is described below).

```

```
' At each stimulus point, the PNA takes power meter readings and determine when
' they have settled by comparing the magnitude difference between consecutive
' readings versus a nominal dB tolerance limit (NTOLerance) on that magnitude
' difference. When consecutive readings are within tolerance of each other, or
' if they are not within tolerance but we've taken a maximum number of readings
' (COUNT), the PNA does a weighted average of the readings taken at that stimulus
' point and that is considered our settled power reading.
```

```
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:AVER:NTOL 0.1")
```

```
If (status < VI_SUCCESS) Then HandleVISAError
```

```
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:AVER:COUN 5")
```

```
If (status < VI_SUCCESS) Then HandleVISAError
```

```
' Specify if the cal power level is offset (positive value for a gain, negative
' value for a loss) from the PNA port power setting on the channel when no source
' power cal is active. This is to account for components between the PNA test
' port and cal reference plane. In this example, we will calibrate at the PNA
' test port, so there is no offset (it is zero).
```

```
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:OFFS 0 DB")
```

```
If (status < VI_SUCCESS) Then HandleVISAError
```

```
' Show the source power cal dialog during the source power cal acquisition.
' (this is the default, so this command is only necessary if this setting
' may have been changed beforehand, perhaps by another program).
```

```
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:DISP ON")
```

```
If (status < VI_SUCCESS) Then HandleVISAError
```

```
' Clear the PNA's SCPI status registers because we are going to be monitoring
' the event status register to detect when the OPC bit gets set indicating
' the cal ACQUIRE completed.
```

```
status = myGPIBWrite(viPNA, "*CLS")
```

```
If (status < VI_SUCCESS) Then HandleVISAError
```

```
' Setting the I/O timeout value to 6000 milliseconds (6 seconds), because the
' PNA may take up to that amount of time to respond to some commands/queries
' while the cal ACQUIRE is progressing.
```

```
status = viSetAttribute(viPNA, VI_ATTR_TMO_VALUE, 6000)
```

```

If (status < VI_SUCCESS) Then HandleVISAError
' Specify the method (type of device) that will be used to perform the cal.
' Choose from power meter (PMETer), power meter and receiver (PMReceiver)
' or just receiver (RECeiver).
' PMReceiver uses the power meter for the first iteration of each point and
' the PNA's reference receiver for subsequent iterations, so is much faster
' than using power meter only. But the power meter accounts for compression
' when calibrating at the output of an active device, whereas the reference
' receiver cannot unless it is coupled to the cal reference plane (on a PNA
' which allows direct access to the receivers).
' Perform the source power cal acquisition sweep using the sensor attached to
' Channel A of the power meter (asking for an OPC reply when it's done). This
' assumes that the power sensor is already connected to Port 2 of the PNA.

' We'll put up an hourglass cursor while waiting for the acquire to complete.
Screen.MousePointer = vbHourglass
status = myGPIBWrite(viPNA, "SOUR1:POW:CORR:COLL:ACQ PMET,'ASEN','Port
2',ASYN;*OPC")
If (status < VI_SUCCESS) Then HandleVISAError
' Other valid selections would be the following:
' This mode uses Power Meter and Reference Receiver
'status = myGPIBWrite(viPNA, "SOUR1:POW:CORR:COLL:ACQ PMR,'BSEN','Port
2',ASYN;*OPC")
' This mode uses PNA receiver only (no power meter)
'status = myGPIBWrite(viPNA, "SOUR1:POW:CORR:COLL:ACQ REC,'b1','Port
2',ASYN;*OPC")

' Polling in a loop to detect when the OPC bit (bit 0, weight value 1) gets
' set in the Event Status Register indicating the ACQuire finished. In this
' type of loop is where you could do other operations in-between the polling
' (like having your program's user-interface still respond to user input).
' If instead of Visual Basic you are programming in C or C++, as an
' alternative to having a polling loop like this, you could set up an SRQ
' handling function in your program (for example, see the documentation
' supplied with your vendor's implementation of VISA on how to register for
' callback when an SRQ event occurs).
Do

```

```

    status = myGPIBWrite(viPNA, "*ESR?")
    If (status < VI_SUCCESS) Then HandleVISAError
    status = myGPIBRead(viPNA, strReply)
    If (status < VI_SUCCESS) Then HandleVISAError
    esrByte = CByte(strReply)
Loop While (esrByte And 1) = 0

' Change mouse cursor from hourglass back to normal
Screen.MousePointer = vbDefault

' Conclude the calibration. This applies the cal data to PNA channel memory,
' and turns the correction ON for Port 2 on Channel 1,
' but does NOT save the calibration.
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:COLL:SAVE")
If (status < VI_SUCCESS) Then HandleVISAError

' At this point, if you choose to save the instrument state as a ".CST" file,
' the calibration will be saved with the instrument state in that file.

' Prepare for doing data transfer in ASCII format.
status = myGPIBWrite(viPNA, "FORM:DATA ASCII")
If (status < VI_SUCCESS) Then HandleVISAError

' Read the stimulus values from Channel 1.
status = myGPIBWrite(viPNA, "SENS1:X?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError

' Tokenize the reply string into an array containing the values
strStimulus = Split(strReply, ",")

' Read the source power correction data.
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:DATA?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)

```

```

If (status < VI_SUCCESS) Then HandleVISAError

' Tokenize the reply string into an array containing the values
strCalValue = Split(strReply, ",")

' Print the data using a message box (here, Chr returns the ASCII characters
' for Tab (9) and Linefeed (10)).
strResult = "Stimulus" & Chr(9) & Chr(9) & "Cal Value" & Chr(10)
For i = 0 To UBound(strStimulus)
strResult = strResult & Val(strStimulus(i)) & Chr(9) & Val(strCalValue(i)) &
Chr(10)
Next
MsgBox strResult
End Sub

Private Function myGPIBWrite(ByVal viHandle As Long, ByVal strOut As String) As Long

' The "+ Chr$(10)" appends an ASCII linefeed character to the
' output, for terminating the write transaction.
myGPIBWrite = viVPrintf(viHandle, strOut + Chr$(10), 0)
End Function

Private Function myGPIBRead(ByVal viHandle As Long, strIn As String) As Long
myGPIBRead = viVScanf(viHandle, "%t", strIn)
End Function

Sub HandleVISAError()
Dim strVisaErr As String * 200
Call viStatusDesc(defRM, status, strVisaErr)
MsgBox "*** Error : " + strVisaErr, vbExclamation

' Close the resource manager session (which also closes
' the session to the PNA).

If defRM <> 0 Then Call viClose(defRM)
End

```



```

End Sub

Public Sub Wait(ByVal mS_delay As Long)

Dim t0 As Single
t0 = Timer
Do While Timer - t0 < mS_delay / 1000
Dim dummy As Integer
dummy = DoEvents() ' if we cross midnight, back up one day
If Timer < t0 Then t0 = t0 - 86400
Loop
End Sub

```

### Perform a Receiver Power Cal

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file, such as Notepad, and save it on the PNA hard drive as \*.vbs.

[Learn how to setup and run the macro.](#)

```

Dim pna
Dim scpi
Set pna = CreateObject("AgilentPNA835x.Application")
Set scpi = pna.ScpStringParser
' For simplicity, this example starts from the preset instrument state
scpi.Execute "SYST:PRESet"
' Turn off continuous sweep
scpi.Execute "INITiate:CONTinuous OFF"
' Select the S11 measurement that was created by the instrument preset
scpi.Execute "CALCulate:PARAMeter:SElect 'CH1_S11_1'"
' Change the measurement parameter to measure the B receiver
scpi.Execute "CALCulate:PARAMeter:MODify B,1"
' Specify the Calibration Type, then Prompt
' to ensure the receiver is connected to port 1.
scpi.Execute "SENSe:CORRection:COLLect:METHod RPOWer"
MsgBox "Connect port 1 to port 2 so power is supplied to the B receiver, then press
enter"
' Acquire the power measurement; returning reply to *OPC? when finished.
response = scpi.Execute( "SENSe:CORRection:COLLect:ACQuire POWer;*OPC?" )
' Compute the error term, store to calset and turn on the calibration.
response = scpi.Execute( "SENSe:CORRection:COLLect:SAVE" )
MsgBox "Done with calibration."

```

Last modified:

9/22/06 Modified for receiver only feature

## Uploading a Source Power Cal using SCPI

---

Programming the PNA using COM or using SICL/VISA over LAN (as in this example) leaves the PNA free to control GPIB devices as needed. This Visual Basic program demonstrates:

- Uploading a source power calibration of Port 2 for Channel 1.
- Reading the calibration data.

Learn more about [Power Calibrations](#)

### Other SCPI Example Programs

To run this program, you need:

- Your PC and PNA both connected to a LAN (if using VISA LAN server / client).
- The SICL and VISA components of Agilent's I/O Libraries software installed on your PC (both are included when you install the software, unless you already have another vendor's VISA installed. Then specify Full SICL and VISA installation to overwrite the other vendor's VISA.
- The module visa32.bas added to your VB project.
- A form with two buttons: cmdRun and cmdQuit.
- A VISA interface configured on your remote PC to control the PNA. This could be GPIB interface or a [VISA LAN Client](#).

```
'Session to VISA Default Resource Manager
Private defRM As Long
'Session to PNA
Private viPNA As Long
'VISA function status return code
Private status As Long
Private Sub Form_Load()
defRM = 0
End Sub
Private Sub cmdRun_Click()

' String to receive data from the PNA.
' Dimensioned large enough to receive scalar comma-delimited values
' for 21 frequency points (20 ASCII characters per point)
Dim strReply As String * 420
Dim strPower As String, strCalPower As String
Dim strStimulus, strCalValue
Dim strResult As String

' Open the VISA default resource manager
```

```

status = viOpenDefaultRM(defRM)
If (status < VI_SUCCESS) Then HandleVISAError

' Open a session (viPNA) to the PNA at "address 16" on the VISA
' interface configured as "GPIB0" on this PC.
status = viOpen(defRM, "GPIB0::16::INSTR", 0, 0, viPNA)
If (status < VI_SUCCESS) Then HandleVISAError

' Set the number of sweep points to 2 on Channel 1.
status = myGPIBWrite(viPNA, "SENS1:SWE:POIN 2")
If (status < VI_SUCCESS) Then HandleVISAError

' Ensure there's currently no source power cal on for this channel and port.
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR OFF")
If (status < VI_SUCCESS) Then HandleVISAError

' Specify if the cal power level is offset (positive value for a gain, negative
' value for a loss) from the PNA port power setting on the channel when no source
' power cal is active. This is to account for components between the PNA test
' port and cal reference plane. In this example, let's set up our calibration
' at the output of an amplifier with 15 dB gain.
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:OFFS 15 DB")
If (status < VI_SUCCESS) Then HandleVISAError

' Prepare for doing data transfer in ASCII format.
status = myGPIBWrite(viPNA, "FORM:DATA ASCII")
If (status < VI_SUCCESS) Then HandleVISAError

' Send our source power correction data to the PNA. For purpose of simplicity
' in this example, we'll set up for no correction (0) at our start stimulus and
' 0.5 dB at our stop stimulus (recall that our sweep currently has just 2 points).
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:DATA 0,0.5")
If (status < VI_SUCCESS) Then HandleVISAError

' Set the number of sweep points to 21 on Channel 1.
status = myGPIBWrite(viPNA, "SENS1:SWE:POIN 21")
If (status < VI_SUCCESS) Then HandleVISAError

' Read the fixed power level for this port on Channel 1.
status = myGPIBWrite(viPNA, "SOUR1:POW2:LEV?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError
strPower = strReply

' Turn the source power cal on.
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR ON")
If (status < VI_SUCCESS) Then HandleVISAError

' Again read the fixed power level for this port on Channel 1
' (with our calibration turned on, this should now include the 15 dB offset
' we indicated our power amplifier provides).

```

```

status = myGPIBWrite(viPNA, "SOUR1:POW2:LEV?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError
strCalPower = strReply

' Read the stimulus values from Channel 1.
status = myGPIBWrite(viPNA, "SENS1:X?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError

' Tokenize the reply string into an array containing the values
strStimulus = Split(strReply, ",")

' Read back the source power correction data, now interpolated for 21 points
status = myGPIBWrite(viPNA, "SOUR1:POW2:CORR:DATA?")
If (status < VI_SUCCESS) Then HandleVISAError
status = myGPIBRead(viPNA, strReply)
If (status < VI_SUCCESS) Then HandleVISAError

' Tokenize the reply string into an array containing the values
strCalValue = Split(strReply, ",")

' Print the data using a message box (here, Chr returns the ASCII characters
' for Tab (9) and Linefeed (10)).
strResult = "PNA port power = " & Val(strPower) & Chr(10)
strResult = strResult & "Power at reference plane = " & Val(strCalPower) & Chr(10)
strResult = strResult & "Stimulus" & Chr(9) & Chr(9) & "Cal Value" & Chr(10)
For i = 0 To UBound(strStimulus)
    strResult = strResult & Val(strStimulus(i)) & Chr(9) & Val(strCalValue(i)) &
Chr(10)
Next
MsgBox strResult
End Sub
Private Sub cmdQuit_Click()

' Close the resource manager session (which also closes
' the session to the PNA).
If defRM <> 0 Then Call viClose(defRM)

' End the program
End
End Sub
Private Function myGPIBWrite(ByVal viHandle As Long, ByVal strOut As String) As Long

' The "+ Chr$(10)" appends an ASCII linefeed character to the
' output, for terminating the write transaction.
myGPIBWrite = viPrintf(viHandle, strOut + Chr$(10), 0)
End Function
Private Function myGPIBRead(ByVal viHandle As Long, strIn As String) As Long

```

```
myGPIBRead = viVScanf(viHandle, "%t", strIn)

' Remove trailing linefeed character
If Right(strIn, 1) = Chr(10) Then strIn = Left(strIn, Len(strIn) - 1)
End Function
Sub HandleVISAError()
Dim strVisaErr As String * 200
Call viStatusDesc(defRM, status, strVisaErr)
MsgBox "*** Error : " + strVisaErr, vbExclamation

' Close the resource manager session (which also closes
' the session to the PNA).
If defRM <> 0 Then Call viClose(defRM)
End
End Sub
```

## Perform a Sliding Load Calibration using GPIB

---

This Visual Basic program does a **only** the sliding load portion of a Calibration.  
To run this program, you need:

- An established [GPIB interface connection](#)
- A measurement and calibration routine to call this sub-program
- STAN3 set up as a sliding load standard

---

[See Other SCPI Example Programs](#)

---

```
Sub slide()  
'Measure the sliding load for at least 5 and no more than 7 slides  
'Note that "SLSET" and "SLDONE" must be executed before the actual acquisition of a  
slide  
MsgBox "Connect Sliding Load; set to Position 1; then press OK"  
GPIB.Write "SENS:CORR:COLL SLSET"  
GPIB.Write "SENS:CORR:COLL STAN3;"  
  
MsgBox "Set Sliding Load to position 2; then press OK"  
GPIB.Write "SENS:CORR:COLL SLSET"  
GPIB.Write "SENS:CORR:COLL STAN3;"  
  
MsgBox "Set Sliding Load to position 3; then press OK"  
GPIB.Write "SENS:CORR:COLL SLDONE"  
GPIB.Write "SENS:CORR:COLL STAN3;"  
End Sub
```

## Load Error Terms during a Cal Sequence

---

This example requires that you already have a Cal Set named "foo" that contains a 1-port cal on port 1 and a 1-port cal on port 2.

This example starts a Guided Calibration specifying an Unknown Thru. It loads the 1-port Cals from the existing "foo" Cal Set, then recalculates the number of steps required to complete the cal. After loading the 1-port cals, only the Unknown Thru standard is left to acquire.

```
SENS:CORR:COLL:GUID:CONN:PORT1 "APC 3.5 female"
SENS:CORR:COLL:GUID:CONN:PORT2 "APC 3.5 female"
SENS:CORR:COLL:GUID:CKIT:PORT1 "85033D/E"
SENS:CORR:COLL:GUID:CKIT:PORT2 "85033D/E"
SENS:CORR:COLL:GUID:METH UNKN
' auto-create user calsets for SCPI
SENS:CORR:PREF:CSET:SAVU 1
SENS:CORR:COLL:GUID:INIT
' should return the number 7
SENS:CORR:COLL:GUID:STEPS?
' to port 1, from port 1 in calset
SENS:CORR:COLL:GUID:ETER:LOAD "foo",1,1
' to port 2, from port 2 in calset
SENS:CORR:COLL:GUID:ETER:LOAD "foo",2,2
' should now return the number 1
SENS:CORR:COLL:GUID:STEPS?
' measure the unknown thru
SENS:CORR:COLL:GUID:ACQ STAN1
' save the cal to new user calset
SENS:CORR:COLL:GUID:SAVE
```



## Create New Cal Kit using SCPI

---

When creating new cal kits programmatically, the order in which cal kit commands are sent can be important. For example to create a kit with opens, shorts, loads, and thrus. Be sure to use the following sequence for each newly defined standard.

1. Programmatically select the standard number
2. Programmatically select the standard type.
3. Program the cal standard's values.
4. Repeat steps 1, 2, 3 for additional new standards being defined.

```
10  !
20  !
30  ! This example program demonstrates how to create
40  ! new PNA calibration kits.
50  !
60  ! 1) Select a kit not previously defined
70  ! 2) Define open, short, load, and thru cal standards
80  !     Note: Each of the newly defined standards is assigned
90  !     a default connector name. These default connector names
100 !     will be replaced in subsequent steps.
110 ! 3) Use the delete connector command to remove default
120 !     connector names.
130 ! 4) Add connectors. Specify:
140 !     Start and Stop Freq
150 !     Z - Impedance
160 !     sex - MALE, FEMALE, NONE
170 !     media - COAX, WAVE
180 !     cutoff - Frequency for waveguide
190 ! 5) Assign the appropriate connector to each standard
200 ! 6) Modify the class assignments for the standards defined
210 ! 7) Verify the kit values
220 !
230 ! Additional Note: After setting each new cal kit value, it is
240 ! recommended that the program periodically perform queries to
250 ! verify the new values.
260 !
270 ! This will prevent program synchronization issues that can affect
280 ! final values stored within new cal kits.
290 !
300 !-----
310 !
320 ! Set up I/O path
330  ASSIGN @Na TO 716
340  DIM Calkname$[80],Conn$[80]
350  INTEGER Calkitnum
```

```

360  !
370  CLEAR SCREEN
380  !
390  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
400  ! Designate the kit selection to be used for performing cal's
410  OUTPUT @Na;":sens:corr:ckit:count?"
420  ENTER @Na;Calkitnum
430  Calkitnum=Calkitnum+1
440  OUTPUT @Na;":sens:corr:coll:ckit "&VAL$(Calkitnum)
450  !
460  ! Name this kit with your own name
470  OUTPUT @Na;":sens:corr:coll:ckit:name "Special 2.4 mm Model 85056""
480  !
490  !
500  DISP "Defining kit std 1..."
510  ! Now set up standard #1
520  OUTPUT @Na;":sens:corr:coll:ckit:stan 1"
530  OUTPUT @Na;":sens:corr:coll:ckit:stan:type SHORT"
540  Get_std
550  OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
560  OUTPUT @Na;":sens:corr:coll:ckit:stan:label "My Short""
570  Get_label
580  !
590  DISP "Defining kit std 2..."
600  ! Now set up standard #2
610  OUTPUT @Na;":sens:corr:coll:ckit:stan 2"
620  OUTPUT @Na;":sens:corr:coll:ckit:stan:type OPEN"
630  Get_std
640  OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
650  OUTPUT @Na;":sens:corr:coll:ckit:stan:label "My Open""
660  Get_label
670  !
680  DISP "Defining kit std 3..."
690  ! Now set up standard #3
700  OUTPUT @Na;":sens:corr:coll:ckit:stan 3"
710  OUTPUT @Na;":sens:corr:coll:ckit:stan:type LOAD"
720  Get_std
730  OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
740  OUTPUT @Na;":sens:corr:coll:ckit:stan:label "My Fixed Load""
750  Get_label
760  !
770  DISP "Defining kit std 4..."
780  ! Now set up standard #4
790  OUTPUT @Na;":sens:corr:coll:ckit:stan 4"
800  OUTPUT @Na;":sens:corr:coll:ckit:stan:type THRU"
810  Get_std
820  OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
830  OUTPUT @Na;":sens:corr:coll:ckit:stan:label "My Thru""
840  Get_label

```

```

850  !
860  DISP "Defining kit std 5..."
870  ! Now set up standard #5
880  OUTPUT @Na;":sens:corr:coll:ckit:stan 5"
890  OUTPUT @Na;":sens:corr:coll:ckit:stan:type SLOAD"
900  Get_std
910  OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
920  OUTPUT @Na;":sens:corr:coll:ckit:stan:label ""Sliding Load""
930  Get_label
940  !
950  DISP "Defining kit std 6..."
960  ! Now set up standard #6
970  !
980  OUTPUT @Na;":sens:corr:coll:ckit:stan 6"
990  OUTPUT @Na;":sens:corr:coll:ckit:stan:type SHORT"
1000 Get_std
1010 OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
1020 OUTPUT @Na;":sens:corr:coll:ckit:stan:label ""Short""
1030 Get_label
1040 !
1050 DISP "Defining kit std 7..."
1060 ! Now set up standard #7
1070 OUTPUT @Na;":sens:corr:coll:ckit:stan 7"
1080 OUTPUT @Na;":sens:corr:coll:ckit:stan:type SHORT"
1090 Get_std
1100 OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
1110 OUTPUT @Na;":sens:corr:coll:ckit:stan:label ""Short""
1120 Get_label
1130 !
1140 DISP "Defining kit std 8..."
1150 ! Now set up standard #8
1160 !
1170 OUTPUT @Na;":sens:corr:coll:ckit:stan 8"
1190 OUTPUT @Na;":sens:corr:coll:ckit:stan:type ARBI"
1200 Get_std
1210 OUTPUT @Na;":sens:corr:coll:ckit:stan:char coax"
1220 OUTPUT @Na;":sens:corr:coll:ckit:stan:TZR 15;"
1230 OUTPUT @Na;":sens:corr:coll:ckit:stan:TZI -9;"
1240 OUTPUT @Na;":sens:corr:coll:ckit:stan:label ""Z Load""
1250 Get_label
1260 !
1270 !
1280 !
1290 ! First remove any old connector names
1300 OUTPUT @Na;":sens:corr:coll:ckit:conn:del"
1310 ! Verify that no connectors are currently installed
1320 OUTPUT @Na;":sens:corr:coll:ckit:conn:cat?"
1330 ENTER @Na;Conn$
1340 PRINT "Verify empty list: ";Conn$

```

```

1350 !
1360 ! Define your new connectors
1370 OUTPUT @Na;":sens:corr:coll:ckit:conn:add '"PSC
2.4"',0HZ,999GHZ,50.0,MALE,COAX,0.0"
1380 OUTPUT @Na;":sens:corr:coll:ckit:conn:add '"PSC
2.4"',0HZ,999GHZ,50.0,FEMALE,COAX,0.0"
1390 !
1400 ! Verify that the new connectors are installed
1410 OUTPUT @Na;":sens:corr:coll:ckit:conn:cat?"
1420 ENTER @Na;Conn$
1430 PRINT "Verify new connectors: ";Conn$
1440 DISP ""
1450 !
1460 DISP "Defining conn std 1..."
1470 ! Now set up standard #1
1480 OUTPUT @Na;":sens:corr:coll:ckit:stan 1"
1490 Verify_std
1500 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',FEMALE,1"
1510 Print_connector
1520 !
1530 DISP "Defining conn std 2..."
1540 ! Now set up standard #2
1550 OUTPUT @Na;":sens:corr:coll:ckit:stan 2"
1560 Verify_std
1570 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',FEMALE,1"
1580 Print_connector
1590 !
1600 DISP "Defining conn std 3..."
1610 ! Now set up standard #3
1620 OUTPUT @Na;":sens:corr:coll:ckit:stan 3"
1630 Verify_std
1640 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',FEMALE,1"
1650 Print_connector
1660 !
1670 DISP "Defining conn std 4..."
1680 ! Now set up standard #4
1690 OUTPUT @Na;":sens:corr:coll:ckit:stan 4"
1700 Verify_std
1710 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',FEMALE,1"
1720 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',MALE,2"
1730 Print_connector
1740 !
1750 DISP "Defining conn std 5..."
1760 ! Now set up standard #5
1770 OUTPUT @Na;":sens:corr:coll:ckit:stan 5"
1780 OUTPUT @Na;":sens:corr:coll:ckit:stan:label '"Sliding Load"'
1790 Verify_std
1800 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam '"PSC 2.4"',MALE,1"
1810 Print_connector

```

```

1820 !
1830 DISP "Defining conn std 6..."
1840 ! Now set up standard #6
1850 !
1860 OUTPUT @Na;":sens:corr:coll:ckit:stan 6"
1870 Verify_std
1880 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam "PSC 2.4","MALE,1"
1890 Print_connector
1900 !
1910 DISP "Defining conn std 7..."
1920 ! Now set up standard #7
1930 OUTPUT @Na;":sens:corr:coll:ckit:stan 7"
1940 Verify_std
1950 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam "PSC 2.4","MALE,1"
1960 Print_connector
1970 !
1980 DISP "Defining conn std 8..."
1990 ! Now set up standard #8
2000 OUTPUT @Na;":sens:corr:coll:ckit:stan 8"
2010 Verify_std
2020 OUTPUT @Na;":sens:corr:coll:ckit:conn:snam "PSC 2.4","MALE,1"
2030 Print_connector
2040 !
2050 DISP "Class assignments..."
2060 !
2070 ! Designate the "order" associated with measuring the standards
2080 !
2090 !     Set Port 1, 1st standard measured to be standard #2
2100 OUTPUT @Na;":sens:corr:coll:ckit:order1 2"
2110 !     Set Port 1, 2nd standard measured to be standard #1
2120 OUTPUT @Na;":sens:corr:coll:ckit:order2 1,6,7"
2130 !     Set Port 1, 3rd standard measured to be standard #3 and #5
2140 OUTPUT @Na;":sens:corr:coll:ckit:order3 3,5"
2150 !     Set Port 1, 4th standard measured to be standard #4
2160 OUTPUT @Na;":sens:corr:coll:ckit:order4 4"
2170 !
2180 !     Set Port 2, 1st standard measured to be standard #2
2190 OUTPUT @Na;":sens:corr:coll:ckit:order5 2"
2200 !     Set Port 2, 2nd standard measured to be standard #1
2210 OUTPUT @Na;":sens:corr:coll:ckit:order6 1,6,7"
2220 !     Set Port 2, 3rd standard measured to be standard #3 and #6
2230 OUTPUT @Na;":sens:corr:coll:ckit:order7 3,5"
2240 !     Set Port 2, 4th standard measured to be standard #4
2250 OUTPUT @Na;":sens:corr:coll:ckit:order8 4"
2260 !
2270 !     Set Port 1, 1st standard
2280 OUTPUT @Na;":sens:corr:coll:ckit:olabel1 "MyOpen1""
2290 !     Set Port 1, 2nd standard
2300 OUTPUT @Na;":sens:corr:coll:ckit:olabel2 "MyShorts1""

```

```

2310 !      Set Port 1, 3rd standard
2320 OUTPUT @Na;":sens:corr:coll:ckit:olabel3 ""MyLoads1""
2330 !      Set Port 1, 4th standard measured to be standard #4
2340 OUTPUT @Na;":sens:corr:coll:ckit:olabel4 ""MyThru1""
2350 !
2360 !      Set Port 2, 1st standard
2370 OUTPUT @Na;":sens:corr:coll:ckit:olabel5 ""MyOpen2""
2380 !      Set Port 2, 2nd standard
2390 OUTPUT @Na;":sens:corr:coll:ckit:olabel6 ""MyShorts2""
2400 !      Set Port 2, 3rd standard
2410 OUTPUT @Na;":sens:corr:coll:ckit:olabel7 ""MyLoads2""
2420 !      Set Port 2, 4th standard
2430 OUTPUT @Na;":sens:corr:coll:ckit:olabel8 ""MyThrus2""
2440 !
2450 BEEP
2460 DISP "Done!"
2470 END
2480 SUB Get_label
2490     OUTPUT 716;":sens:corr:coll:ckit:stan:label?"
2500     ENTER 716;Label$
2510     PRINT Label$
2520 SUBEND
2530 !
2540 SUB Get_std
2550     OUTPUT 716;":sens:corr:coll:ckit:stan:type?"
2560     ENTER 716;Type$
2570     PRINT Type$
2580 SUBEND
2590 !
2600 SUB Print_connector
2610     DIM Nam$[40]
2620     OUTPUT 716;":sens:corr:coll:ckit:conn:sname?"
2630     ENTER 716;Nam$
2640     PRINT Nam$
2650 SUBEND
2660 !
2670 SUB Verify_std
2680     OUTPUT 716;":sens:corr:coll:ckit:stan:label?"
2690     ENTER 716;Label$
2700 SUBEND
2710 !

```

## Modify a Calibration Kit using SCPI

---

This Visual Basic program:

- Modifies Calibration kit number 3
- Completely defines standard #4 (thru)

To run this program, you need:

- An established [GPIB interface connection](#)

[See Other SCPI Example Programs](#)

```
'Modifying cal kit number 3
Calkitnum = 3

'Designate the kit selection to be used for performing cal's
GPIB.Write "SENSe:CORRection:COLLect:CKIT:SElect " & Val(Calkitnum)

'Reset to factory default values.
GPIB.Write "SENSe:CORRection:COLLect:CKIT:RESet " & Val(Calkitnum)

'Name this kit with your own name
GPIB.Write "SENSe:CORRection:COLLect:CKIT:NAME 'My Cal Kit'"

'Assign standard numbers to calibration classes
'Set Port 1, class 1 (S11A) to be standard #8
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer1 8"
'Set Port 1, class 2 (S11B) to be standard #7
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer2 7"
'Set Port 1, class 3 (S11C) to be standard #3
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer3 3"
'Set Port 1, class 4 (S21T) to be standard #4
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer4 4"
'Set Port 2, class 1 (S22A) to be standard #8
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer5 8"
'Set Port 2, class 2 (S22B) to be standard #7
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer6 7"
'Set Port 2, class 3 (S22C) to be standard #3
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer7 3"
'Set Port 2, class 4 (S12T) to be standard #4
GPIB.Write "SENSe:CORRection:COLLect:CKIT:ORDer8 4"

'Set up Standard #4 completely
'Select Standard #4; the rest of the commands act on it
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard 4"
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:FMIN 300KHz"
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:FMAX 9GHz"
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:IMPedance 50"
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:DELay 1.234 ns"
```

```
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:LOSS 23e6"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:C0 0"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:C1 1"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:C2 2"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:C3 3"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:L0 10"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:L1 11"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:L2 12"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:L3 13"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:LABel 'My Special Thru'"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:TYPE THRU"  
GPIB.Write "SENSe:CORRection:COLLect:CKIT:STANdard:CHARacteristic Coax"
```



## Create and Calibrate a VMC Measurement

---

This VB Script example creates and calibrates a Vector mixer measurement. To run this example **without modification** you need the following:

- A Mixer setup file saved on the PNA: C:/Program Files/Agilent/Network Analyzer/Documents/Mixer/MyMixer.mxr.
- If the mixer file uses an external LO source, it must also be attached and configured.
- An ECal module that covers the frequency range of the measurement.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example. However, some modification is necessary to make the program run on a traditional GPIB Interface.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as VMC.vbs. Learn how to setup and run the macro.

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
'---Create a Vector Mixer Measurement
'First, delete all measurements on the channel
scpi.Parse "CALC:PAR:DEL:ALL"
'Create a forward scalar mixer measurement and configure
'it in channel 1.
'The first parameter is a unique identifying string
'(specified by the user) to allow subsequent
'commands to be directed at this specific measurement.
scpi.Parse "CALC:CUST:DEF 'My VC21', 'Vector Mixer/Converter', 'VC21'"
'Setup the new measurement as the 2nd trace in the active window
scpi.Parse "DISP:WIND:TRAC2:FEED 'My VC21'"
'Make the new trace the active measurement
scpi.Parse "CALC:PAR:SEL 'My VC21'"
'The parameters of the mixer measurement can now be configured.
'This can be done by either using the SENS:MIX commands
'for each of the parameters or by loading a mixer setup file.
'This example loads a mixer setup file. The path name
'for the mixer file may be loaded from other mapped drives.
scpi.Parse "SENS:MIXer:Load 'C:/Program Files/Agilent/Network
Analyzer/Documents/Mixer/MyMixer.mxr'"

'-----Perform A Vector Mixer Calibration-----
'Initialize an VMC guided calibration for session number 6
scpi.Parse "SENS:CORR:COLL:SESS6:INIT ""VMC"""
'This sets the VMC operation to full system cal as opposed to
'performing a mixer characterization only.
```

```

scpi.Parse "SENS:CORR:COLL:SESS6:VMC:OPER ""CAL""
'This example uses ECal for the 2-port cal portion of the procedure
'To use a mechanical kit you will have to use the following command:
'scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOPort:OPTion""MECH""
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:TWOPort:OPTion ""ECAL""
'If you select the mechanical method then you also have to
'specify the connector types and the cal kits for each of the ports.
'The comments below show an example of how that is done:
'scpi.Parse "SENS:CORR:COLL:SESS6:CONN:PORT1:SEL ""APC 3.5 male""
'scpi.Parse "SENS:CORR:COLL:SESS6:CONN:PORT2:SEL ""APC 3.5 female""
'scpi.Parse "SENS:CORR:COLL:SESS6:CKIT:PORT1:SEL ""85052D""
'scpi.Parse "SENS:CORR:COLL:SESS6:CKIT:PORT2:SEL ""85052D""
'Choose the between ECal or Mechanical calibration for the
'Mixer Characterization portion of the VMC cal.
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:MIX:CHAR:CAL:OPT ""ECAL""
'This command sets the port mapping for the ECal to be used
'during the Mixer Characterization portion of the VMC cal.
'It is a required command if in the previous command the option
'was set to 'ECAL'. The only valid port maps are either 'A1'
'or 'B1'.
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:MIX:ECAL:PORT 1, ""A1""
'Specify the ECal module and the ECal characterization for the
'two port calibration portion of this session. FCA calibrations
'currently only support ECal module number 1. In this example
'the factory characterization is used by specifying 0 for the
'characterization number.
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:TWOPE:ECAL:CHAR 1,0"
'Specify the ECal module and the ECal characterization for the
'Mixer Characterization portion of this session. FCA calibrations
'currently only support ECal module number 1. In this example
'the factory characterization is used by specifying 0 for the
'characterization number.
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:MIX:ECAL:CHAR 1,0"
'Specify the thru measurement method. This applies to both ECal
'and mechanical calibrations. For ECal 'DEFAULT' will use the ECal
'thru. Other choices may be used depending on the genders and types
'of the connectors on the test interface.
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:TWOPE:METH ""DEFAULT""
'Omit the isolation part of the 2-port cal
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:TWOPE:OMIT 1"
'Turn on auto orientation for the ECal
scpi.Parse "SENS:CORR:COLL:SESS6:VMC:TWOPE:ECAL:ORI:STATE 1"

'Tell the wizard to generate and report the number of steps in this
'cal session
Dim steps
Dim desc
'Determine the number of steps required to complete the calibration.
'First send the write command, then the query.
scpi.Parse "SENS:CORR:COLL:SESS6:STEP"
steps = scpi.Parse ("SENS:CORR:COLL:SESS6:STEP?")
For i = 1 To steps

```

```
'Display the prompt for each step
desc = scpi.Parse ("SENS:CORR:COLL:SESS6:DESC? " & CStr(i))
MsgBox (desc)
'Perform the measurement for each step
scpi.Parse "SENS:CORR:COLL:SESS6:ACQ " & CStr(i)
Next
Dim calset
'Finish the cal and save the calset
calset = scpi.Parse ("SENS:CORR:COLL:SESS6:SAVE?")
scpi.Parse ("SENS:CORR:COLL:SESS6:DONE")
Msgbox ("VMC Cal Complete!")
```

---

Last Modified:

4-Jun-2009    Added 'Done' command

## Create and Cal an SMC Measurement

---

This Visual Basic example creates and calibrates a scalar mixer measurement.

To run this example **without modification** you need the following:

- A Mixer setup file saved on the PNA: C:/Program Files/Agilent/Network Analyzer/Documents/Mixer/MyMixer.mxr.
- If the mixer file uses an external LO source, it must also be attached and configured.
- An ECAL module that covers the frequency range of the measurement.
- A power meter must be attached to the PNA. If this example is run in the PNA, the power meter does not need to be attached using a GPIB/USB interface.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example. However, some modification is necessary to make the program run on a traditional GPIB Interface. For example, during the power meter portion of this calibration, scpi.Parse will not process a command until the power meter routine has completed. Traditional GPIB would require a [serial polling technique](#) to ensure the routine has completed before proceeding.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as SMC.vbs. [Learn how to setup and run the macro.](#)

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
'---Create a Scalar Mixer Forward Measurement
'First, delete all measurements on the channel
scpi.Parse "CALC:PAR:DEL:ALL"
'Create a forward scalar mixer measurement and configure it in
'channel 1. The first parameter is a unique
'identifying string (specified by the user) to allow subsequent
'commands to be directed at this specific measurement.
scpi.Parse "CALC:CUST:DEF 'My SC21', 'Scalar Mixer/Converter', 'SC21'"

'Setup the new measurement as the 2nd trace in the active window
scpi.Parse "DISP:WIND:TRAC2:FEED 'My SC21'"
'Make the new trace the active measurement
scpi.Parse "CALC:PAR:SEL 'My SC21'"
'The parameters of the mixer measurement can now be configured.
'This can be done by either using the individual SENS:MIX commands
'for each of the parameters or by loading a mixer setup file. This
'example loads a mixer setup file. The path name
'for the mixer file may be loaded from other mapped drives
scpi.Parse "SENS:MIXer:Load ""C:/Program Files/Agilent/Network
Analyzer/Documents/Mixer/MyMixer.mxr"" "
'-----Perform A Scalar Mixer Calibration-----
'Initialize an SMC guided calibration for session number 6
```

```

scpi.Parse "SENS:CORR:COLL:SESS6:INIT ""SMC""
'Select to use an ECal for the 2-port cal portion of the procedure
'To use a mechanical kit you will have to use the following command:
'scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOPort:OPTion ""MECH""
scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOPort:OPTion ""ECAL""
'If you select the mechanical method then you also have to
'specify the connector types and the cal kits for each of the ports.
'The comments below show an example of how that is done:
'scpi.Parse "SENS:CORR:COLL:SESS6:CONN:PORT1:SEL ""APC 3.5 male""
'scpi.Parse "SENS:CORR:COLL:SESS6:CONN:PORT2:SEL ""APC 3.5 female""
'scpi.Parse "SENS:CORR:COLL:SESS6:CKIT:PORT1:SEL ""85052D""
'scpi.Parse "SENS:CORR:COLL:SESS6:CKIT:PORT2:SEL ""85052D""
'Specify the ECal module and the ECal characterization for this
'session. FCA calibrations currently only support ECal module
'number 1. In this example the factory characterization is used
'by specifying 0 for the characterization number.
scpi.Parse "SENS:CORR:COLL:SESS6:SMC:ECAL:CHAR 1,0"
'Specify the thru measurement method. This applies to both ECal
'and mechanical calibrations. For ECal 'DEFAULT' will use the ECal
'thru. Other choices may be used depending on the genders and types
'of the connectors on the test interface.
scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOP:METH ""DEFAULT""
'Omit the isolation part of the 2-port cal
scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOP:OMIT 1"
'Turn on auto orientation for the ECal
scpi.Parse "SENS:CORR:COLL:SESS6:SMC:TWOP:ECAL:ORI:STATE 1"
'Tell the wizard to generate and report the number of steps in this
'cal session
Dim steps
Dim desc
'Determine the number of steps required to complete the calibration.
'First send the write command, then the query.
scpi.Parse "SENS:CORR:COLL:SESS6:STEP"
steps = scpi.Parse ("SENS:CORR:COLL:SESS6:STEP?")
For i = 1 To steps
'Display the prompt for each step
desc = scpi.Parse ("SENS:CORR:COLL:SESS6:DESC? " & CStr(i))
MsgBox (desc)
'Perform the measurement for each step
scpi.Parse "SENS:CORR:COLL:SESS6:ACQ " & CStr(i)
Next
Dim calset
'Finish the cal and save the calset
calset = scpi.Parse ("SENS:CORR:COLL:SESS6:SAVE?")
scpi.Parse ("SENS:CORR:COLL:SESS6:DONE")
Msgbox ("SMC Cal Complete!")

```

---

Last Modified:

4-Jun-2009 Added 'Done' command

## Create an SMC Fixed Output Measurement with SCPI

---

This VB Script example creates a calibrated SMC fixed output measurement using a controlled LO. Then a single sweep is taken and data is retrieved.

This example requires that an external LO be previously setup and named '8360'.

Fixed output measurements require that an external LO source be swept and synchronized with the PNA source. FCA performs this synchronization using the external source configuration settings.

The fastest, and recommended, method of controlling the LO source is [Hardware List \(BNC\) triggering mode](#). However, in this mode, FCA channels will not respond to manual triggers. Therefore, the example uses the following mechanism to trigger a sweep:

```
Write "SENS:SWE:MODE HOLD"      'place channel 1 in HOLD mode
Write "INIT:CONT ON"            'place PNA in internal trigger mode
Write "SENS:SWE:GRO:COUNT 1"   'specify that the group count is 1 sweep
Write "SENS:SWE:MODE GROUPS"    'execute group count (1 sweep)
Write "*OPC?"                   'wait until the sweep is complete
Read
```

The SCPI commands in this example are sent over a COM interface using the [SCPIStringParser](#) object. You can run a VBScript (\*.vbs) program from the PNA [using Macros](#). To run this program, copy the following code into a text editor and save it as a \*.vbs file.

```
option explicit

' Setup infrastructure to use the SCPI over COM
dim app
set app = createobject("Agilentpna835x.application")
dim p
set p = app.scpistringparser
dim returnStr
sub Write (command)
if len(returnStr) <> 0 then
err.Raise 55, "Write", "Query Unterminated"
end if
returnStr = p.parse(command)
end sub
sub WriteIgnoreError(command)
returnStr = p.Execute(command)
p.Parse("SYST:ERR?") ' clear error queue
end sub
function Read
if len(returnStr) = 0 then
err.Raise 55, "Read", "Bad read"
```

```

end if
Read = returnStr
returnStr = ""
end function
Write "SYST:PRES"
' When programming in remote mode, hold mode is recommended
Write "SENS:SWE:MODE HOLD"
' Delete the standard measurement
Write "CALC:PAR:DEL:ALL"
' Create an SC21 measurement
Write "CALC:CUST:DEF 'MySMC', 'Scalar Mixer/Converter', 'SC21'"
Write "DISP:WIND:TRACE:FEED 'MySMC'"
Write "CALC:PAR:SEL 'MySMC'"
' Set number of points to 11
Write "SENS:SWE:POIN 11 "
' Setup the mixer parameters for a swept LO, fixed output measurement
Write "SENS:MIX:INP:FREQ:START 200e6"
Write "SENS:MIX:INP:FREQ:STOP 700e6"
Write "SENS:MIX:LO:FREQ:MODE Swept"
Write "SENS:MIX:OUTPUT:FREQ:FIX 3.4e9"
Write "SENS:MIX:OUTP:FREQ:SID HIGH"
Write "SENS:MIX:INP:POW -17"
Write "SENS:MIX:LO:POW 10"
' Specify the LO name, for controlled LO.
' This name is setup in the External Source Config Dialog
Write "SENS:MIX:LO:NAME '8360'"
' The CALC method calculates the LO frequency from the other parameters,
' It also applies the mixer parameters to the channel.
Write "SENS:MIX:CALC LO_1"
' Create an S11 in the same channel
Write "CALC:CUST:DEF 'MyS11', 'Scalar Mixer/Converter', 'S11'"
Write "DISP:WIND:TRACE2:FEED 'MyS11'"
Write "CALC:PAR:SEL 'MyS11'"
' Create an IPwr in the same channel
Write "CALC:CUST:DEF 'MyIPwr', 'Scalar Mixer/Converter', 'IPwr'"
Write "DISP:WIND:TRACE3:FEED 'MyIPwr'"

```



```
Write "CALC:PAR:SEL 'MyIPwr'"
' Create an OPwr in the same channel
Write "CALC:CUST:DEF 'MyOPwr', 'Scalar Mixer/Converter', 'OPwr'"
Write "CALC:PAR:SEL 'MyOPwr'"
Write "DISP:WIND:TRACE4:FEED 'MyOPwr'"
' Perform a single sweep, synchronously. When *OPC returns, the sweep is done
Write "SENS:SWE:GRO:COUN 1"
Write "SENS:SWE:MODE GROUPS"
Write "*OPC?"
Read
' Retrieve the SC21 data
Write "CALC:PAR:SEL 'MySMC'"
Write "CALC:DATA? SDATA"
dim data
data = Read()
wscript.echo(data)
'Retrieve the S11 data
Write "CALC:PAR:SEL 'MyS11'"
Write "CALC:DATA? SDATA"
data = Read()
wscript.echo(data)
```

## Create and Cal a GCA Measurement

---

This VBS program does the following:

- creates and configures GCA to perform a SMART Sweep
- performs a calibration using an ECal with 3.5 mm Female on Port A and 3.5 mm Male connectors on Port B

This program can be run as a macro in the PNA. To do this, copy the code into a text editor file such as Notepad and save on the PNA hard drive as GCA.vbs. Learn how to setup and run the macro.

### See the Gain Compression commands

```
option explicit
dim CompLevel , Tolerance , StartFreq , StopFreq , NumFreqs , Scale , LinearPower
dim AcqMode , BackOff , StartPower , StopPower , NumPowers , EnableInterp , CompAlg
dim DwellTime , IFBandwidth , ShowIterations , host , app , parser
' GCA Settings/Values
''
' Acquisition Mode:
' naSmartSweep = 0
' naSweepPowerAtEachFreq2D = 1
' naSweepFreqAtEachPower2D = 2
''
' Compression Algorithm
' naCompressionFromLinearGain = 0
' naCompressionFromMaximumGain = 1
' naBackoffCompression = 2
' naXYCompression = 3
''
' EndOfSweepOperation
' naDefaultPowerSet = 0
' naSetToStartPower = 1
' naSetToStopPower = 2
' naSetRFOff = 3
''
CompLevel          = 1          ' 1 dB compression level
Tolerance          = 0.05       ' SMART Sweep tolerance
```

```

StartFreq      = 1E9
StopFreq       = 9E9
NumFreqs       = 201
Scale          = 0.1
LinearPower    = -20
BackOff        = 10      ' Not used for Deviation from linear gain
StartPower     = -20
StopPower      = 8
NumPowers      = 60      ' Not used for SMART Sweep
DwellTime      = 0.0005 ' Allow some time for DUT bias/thermal effects
IFBandwidth    = 1000    ' Reasonable trace noise at -20 dBm
EnableInterp   = False   ' Disable interpolation
AcqMode        = 0       ' Smart Sweep
CompAlg        = 0       ' Deviation from linear gain
ShowIterations = False   ' Configure SMART to not show iteration results

dim objargs
set objargs = wscript . Arguments
if ( objArgs . Count = 1) then host = objargs (0)
' .....

' Create and Configuration GCA Channel:
' .....

set app = CreateObject ("Agilentpna835x.application" )
set parser = app .ScpiStringParser
call SetupGCA ( parser ,_
               StartFreq ,_
               StopFreq ,_
               NumFreqs ,_
               EnableInterp ,_
               Scale ,_
               CompLevel ,_
               LinearPower ,_
               AcqMode ,_
               BackOff ,_
               StartPower ,_
               StopPower ,_
               NumPowers ,_

```

```

        CompAlg , _
        DwellTime , _
        IFBAndwidth , _
        ShowIterations )

call CalGCA ( parser )
call Analysis( parser )
.....

'' GCA Setup
.....

sub SetupGCA ( parser , StartFreq , StopFreq , NumFreqs , EnableInterp , Scale ,
CompLevel , LinearPower , _
        AcqMode , BackOff , StartPower , StopPower , NumPowers , CompAlg ,
DwellTime , IFBAndwidth , _
        ShowIterations )

parser . Parse "*RST "
parser . Parse "CALC:PAR:DEL:ALL "
parser . Parse "CALC:CUST:DEF ""S21"", ""Gain Compression"", ""S21"" "
parser . Parse "DISP:WIND:TRAC1:FEED ""S21"" "
parser . Parse "CALC:PAR:SEL ""S21"" "
parser . Parse "CALC:CUST:DEF ""CompIn21"", ""Gain Compression"", ""CompIn21"" "
parser . Parse "DISP:WIND:TRAC2:FEED ""CompIn21"" "
parser . Parse "CALC:CUST:DEF ""DeltaGain21"", ""Gain Compression"", ""DeltaGain21""
"

parser . Parse "DISP:WIND:TRAC3:FEED ""DeltaGain21"" "
parser . Parse "SENS:SWE:MODE HOLD"
parser . Parse "DISP:WIND1:TRAC3:Y:SCAL:PDIV " & Scale
parser . Parse "DISP:WIND1:TRAC3:Y:RLEV " & -CompLevel

select case AcqMode
case 0 ' SMART Sweep
    parser . Parse "SENS:GCS:AMOD SMAR"
case 1 ' 2D Power Sweeps
    parser . Parse "SENS:GCS:AMOD PFREQ"
case 2 ' 2D Freq Sweeps
    parser . Parse "SENS:GCS:AMOD FPOW"
end select

select case CompAlg
case 0 ' Deviation from linear gain

```

```

    parser . Parse "SENS:GCS:COMP:ALG CFLG"
case 1 ' Deviation from max gain
    parser . Parse "SENS:GCS:COMP:ALG CFMG"
case 2 ' Back Off
    parser . Parse "SENS:GCS:COMP:ALG BACK"
case 3 ' XY
    parser . Parse "SENS:GCS:COMP:ALG XYCOM"
end select

if EnableInterp then
    parser . Parse "SENS:GCS:COMP:INT ON"
else
    parser . Parse "SENS:GCS:COMP:INT OFF"
end if

if ShowIterations then
    parser . Parse "SENS:GCS:SMAR:SIT ON"
else
    parser . Parse "SENS:GCS:SMAR:SIT OFF"
end if

parser . Parse "SENS:GCS:COMP:LEV " & CompLevel
parser . Parse "SENS:GCS:COMP:BACK:LEV " & BackOff
parser . Parse "SENS:GCS:COMP:DELT:X " & BackOff
parser . Parse "SENS:GCS:COMP:DELT:Y " & BackOff
parser . Parse "SENS:GCS:SWE:FREQ:POIN " & NumPowers
parser . Parse "SENS:GCS:SMAR:STIM " & DwellTime
parser . Parse "SENS:BAND " & IFBandwidth
parser . Parse "SENS:SWE:DWEL " & DwellTime
parser . Parse "SOUR:POW:STAR " & StartPower
parser . Parse "SOUR:POW:STOP " & StopPower
parser . Parse "SENS:FREQ:STAR " & StartFreq
parser . Parse "SENS:FREQ:STOP " & StopFreq
parser . Parse "SENS:SWE:POIN " & NumFreqs
parser . Parse "SENS:SWE:MODE SING"

```

```

dim str
str = parser .Parse ( "* OPC ?" )

end sub

.....

' GCA Calibration
.....

sub CalGCA ( parser )
    dim CalSteps , I
    parser . parse "SENS:CORR:COLL:GUID:CONN:PORT1 'APC 3.5 female'"
    parser . parse "SENS:CORR:COLL:GUID:CONN:PORT2 'APC 3.5 male'"
    parser . parse "SENS:CORR:COLL:GUID:CKIT:PORT1 'N4691-60004 ECal '"
    parser . parse "SENS:CORR:COLL:GUID:CKIT:PORT2 'N4691-60004 ECal '"
    parser . parse "SENS:CORR:GCSetup:POW 0"
    parser . parse "SENS:CORR:COLL:GUID:INIT "
    CalSteps = parser . parse ( " SENS:CORR:COLL:GUID:STEP ?" )
    for I = 1 to CalSteps
        msgBox parser .parse ( "SENS:CORR:COLL:GUID:DESC ? " & I )
        parser . parse ( "SENS:CORR:COLL:GUID:ACQ STAN"& I )
    next
    parser . parse "SENS:CORR:COLL:GUID:SAVE "
    msgBox "Done"
end sub

.....

' GCA Analysis
.....

sub Analysis( parser )
'select measurement 1
parser.parse "CALC:PAR:MNUM 1"
parser.parse "CALC:GCM:ANAL:ENABLE 1" 'turn on the analysis mode
parser.parse "CALC:GCM:ANAL:CWFR 1e9" 'set the analysis cw frequency
'select measurement 2
parser.parse "CALC:PAR:MNUM 2"
parser.parse "CALC:GCM:ANAL:ENABLE 1"

```

```
parser.parse "CALC:GCM:ANAL:CWFR 2e9"  
parser.parse "CALC:GCM:ANAL:XAX PSO" 'set the axis to power settings  
'select measurement 3  
parser.parse "CALC:PAR:MNUM 3"  
parser.parse "CALC:GCM:ANAL:ENABLE 1"  
parser.parse "CALC:GCM:ANAL:CWFR 3e9"  
parser.parse "CALC:GCM:ANAL:ISD 0" ' set the discrete frequency option to false  
e nd sub
```

---

Last Modified:

3-Sep-2009 Added Analysis

173-Jun-2010 MX New topic

## Create and Cal a Noise Figure Measurement

---

This program does the following:

- Setup a Noise Figure Measurement
- Calibrate Noise Figure

This program can be run as a macro in the PNA. To do this, copy the code into a text editor file such as Notepad and save on the PNA hard drive as NF.vbs. [Learn how to setup and run the macro.](#)

[See the Noise figure commands.](#)

---

[See Other SCPI Example Programs](#)

```
' This section gets the PNA application
' starts the scpi parser, and presets the PNA
windowNum = 1
channelNum = 1
set pna=CreateObject("AgilentPNA835x.Application")
set scpi = pna.ScpiStringParser
' Create noise figure measurement
scpi.Parse "SYST:FPR"
scpi.Parse "DISP:WIND ON"
scpi.Parse "CALC:CUST:DEF 'noiseFig', 'Noise Figure Cold Source', 'NF'"
scpi.Parse "DISP:WIND:TRAC:FEED 'noiseFig'"
scpi.Parse "CALC:PAR:SEL 'noiseFig'"
' Substitute appropriate Ecal identification strings here
tunerEcal = "N4691-60004 ECal 02821"
pullEcal = "N4691-60004 ECal 02297"
' configure channel
ConfigureChannel
ConfigureNoiseSettings
' perform calibration
SetupNoiseSource
SetupCalAttributes_Insertable
FinishCalibration
' ----- Support subroutines -----
```



```

' Configure noise channel
sub ConfigureChannel
    scpi.Parse "SENS:FREQ:START 750MHz"
    scpi.Parse "SENS:FREQ:STOP 5.0GHz"
    scpi.Parse "SENS:SWEEP:POINTS 401"
    scpi.Parse "SENS:BWID 1.0E3"
end sub

' Configure noise-specific channel settings
sub ConfigureNoiseSettings
    scpi.Parse "SENS:NOIS:AVER:STAT ON"      ' turn averaging ON
    scpi.Parse "SENS:NOIS:AVER 40"           ' noise averaging
    scpi.Parse "SENS:NOIS:BWID 8MHz"         ' noise bandwidth
    scpi.Parse "SENS:NOIS:GAIN 30"           ' gain of noise receiver
    scpi.Parse "SENS:NOIS:TEMP:AMB 301"      ' ambient temperature, in Kelvin
    scpi.Parse "SENS:NOIS:IMP:COUN 5"        ' number of tuner impedance states
    scpi.Parse "SENS:NOIS:TUN:ID '" & tunerEcal & "'" ' set ID of tuner Ecal module
    scpi.Parse "SENS:NOIS:TUN:INP 'B'"       ' orientation of tuner input port
    scpi.Parse "SENS:NOIS:TUN:OUTP 'A'"      ' orientation of tuner output port
    scpi.Parse "SENS:CORR:TCOL:USER:VAL 300" ' noise source cold temperature
end sub

sub SetupCalAttributes_Insertable
    scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT1 'APC 3.5 female'"
    scpi.Parse "SENS:CORR:COLL:GUID:CONN:PORT2 'APC 3.5 male'"
    scpi.Parse "SENS:CORR:COLL:GUID:CKIT:PORT1 '" & pullEcal & "'" ' port 1
    calkit
    scpi.Parse "SENS:CORR:COLL:GUID:CKIT:PORT2 '" & pullEcal & "'" ' port 2
    calkit

    scpi.Parse "SENS:NOIS:SOUR:CONN 'APC 3.5 male'"      ' noise source connector
type
    scpi.Parse "SENS:NOIS:SOUR:CKIT '" & pullEcal & "'"  ' noise source calkit
    scpi.Parse "SENS:NOISE:CAL:METHOD 'Vector'"          ' cal method
    scpi.Parse "SENS:CORR:COLL:GUID:INIT"
end sub

sub SetupNoiseSource
    ' specify the ENR file for the noise source
    enrfile = "C:\Program Files\Agilent\Network Analyzer\Noise\346C_MY44420454.enr"

```

```
    scpi.Parse "SENS:NOIS:ENR:FILENAME '" & enrfile & "'"
' set noise source cold temperature
    scpi.Parse "SENS:CORR:TCOLd:USER:VAL 301.1"
end sub
' Build the connection list and acquire the calibration
sub FinishCalibration
    steps = scpi.Parse("SENS:CORR:COLL:GUID:STEPS?")
    for i = 1 to steps
        str = scpi.Parse("SENS:CORR:COLL:GUID:DESC? " & i)
        msgbox str
        scpi.Parse "SENS:CORR:COLL:GUID:ACQ STAN" & i
    next
    scpi.Parse "SENS:CORR:COLL:GUID:SAVE 0"
    wscript.echo "Calibration complete"
end sub
```

## Show Custom Cal Windows during a Guided Calibration

---

This VBScript program shows how to send commands that allow you to view specific 'custom' windows, and sweep specific channels, during a UI (Cal Wizard) or remote calibration.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as CalWindow.vbs. [Learn how to setup and run the macro.](#)

These commands are used to show and sweep windows and channels:

- [SENS:CORR:COLL:DISP:WIND](#)
- [SENS:CORR:COLL:SWE:CHAN](#)
- [SENS:CORR:COLL:DISP:WIND:AOff](#)
- [SENS:CORR:COLL:SWE:CHAN:AOff](#)
- [SENS:CORR:COLL:GUID:PACQuire](#)

### [See Other SCPI Example Programs](#)

```
Dim app
Dim scpi
' Create / Get the PNA application.
Set app = CreateObject("AgilentPNA835x.Application")
Set scpi = app.ScpiStringParser
' A comment
'Preset the analyzer
'This creates an S11 measurement in channel 1, window 1
scpi.Execute "SYST:PReset"
' Create and turn on window 2
scpi.Execute "DISPlay:WINDow2:STATE ON"
'Define an S21 measurement in channel 2
scpi.Execute "CALCulate2:PARameter:DEFine 'MyMeas',S21"
'Associate ("FEED") the measurement name ('MyMeas') to WINDow2
'and give the new TRACe a number (1).
scpi.Execute "DISPlay:WINDow2:TRACe1:FEED 'MyMeas' "
```

```

'The following lines are all you need in order to:
'show and sweep the custom Cal windows during a UI Calibration
'If sending ONLY these commands, make sure you know the
'correct window and channel numbers to show and sweep.
'Flag windows 1 and 2 to show during Ch1 calibration
scpi.Execute "SENS:CORR:COLL:DISP:WIND1 ON"
scpi.Execute "SENS:CORR:COLL:DISP:WIND2 ON"
'Flag channels 1 and 2 to sweep during Ch1 calibration
scpi.Execute "SENS1:CORR:COLL:SWE:CHAN1 ON"
scpi.Execute "SENS1:CORR:COLL:SWE:CHAN2 ON"

' =====
' The following code performs a remote guided Cal on Ch1.
' From a remote cal, the Cal window does not normally show and sweep
' after the previous standard has been acquired.
' This shows how to include the PACquire (preview) to view and sweep the Cal
Window.
' The Custom window also shows and sweeps due to the flag commands above.
' The flags are cleared at the end of this section.

' Specify the DUT connectors
scpi.Execute "sens:corr:coll:guid:conn:port1 ""APC 3.5 female"" "
scpi.Execute "sens:corr:coll:guid:conn:port2 ""APC 3.5 male"" "
' Select the Cal Kit for each port being calibrated.
scpi.Execute "sens:corr:coll:guid:ckit:port1 ""85052D"" "
scpi.Execute "sens:corr:coll:guid:ckit:port2 ""85052D"" "
' Initiate the calibration and query the number of steps
scpi.Execute "sens:corr:coll:guid:init"
numSteps = scpi.Execute("sens:corr:coll:guid:steps?")
MsgBox "Number of steps is " + CStr(numSteps)
' Measure the standards
For i = 1 to numSteps
step = "Step " + CStr(i) + " of " + CStr(numSteps)
strPrompt = scpi.Execute("sens:corr:coll:guid:desc? " + CStr(i))
'send the Preview Acquire command, then prompt
scpi.Execute "sens:corr:coll:guid:PACquire STAN" + CStr(i)

```

```
' Do NOT send any Guided Cal commands here or the cal window will not sweep
MsgBox strPrompt, vbOKOnly, step
scpi.Execute "sens:corr:coll:guid:acq STAN" + CStr(i)
Next
' Conclude the calibration
scpi.Execute "sens:corr:coll:guid:save"
MsgBox "Cal is done!"

'Remove the Custom Window flags
scpi.Execute "SENS:CORR:COLL:DISP:WIND:AOFF"
'Remove the channel sweep flags
scpi.Execute "SENS:CORR:COLL:SWE:CHAN:AOFF"
```

---

Last Modified:

1-Nov-2007    New topic

## Getting and Putting Data using SCPI

---

This Visual Basic Program does the following:

- Reads data from the analyzer
- Puts the data back into memory
- To see the data on the analyzer after running the program, from the front panel click:  
**Trace - Math/Memory - Memory Trace**

To run this program, you need:

- An established [GPIB interface connection](#)

### [See Other SCPI Example Programs](#)

**Note:** To change the read and write location of data, removing the comment from the beginning of ONE of the lines, and replace the comment in the beginning of the SDATA and SMEM lines.

```
Private Sub ReadWrite_Click()  
Dim i As Integer  
Dim t As Integer  
Dim q As Integer  
Dim dat As String  
Dim cmd As String  
Dim datum() As Double  
  
GPIB.Configure  
GPIB.Write "SYSTem:PRESet;*wai"  
  
'Select the measurement  
GPIB.Write "CALCulate:PARAmeter:SElect 'CH1_S11_1'"  
  
'Read the number of data points  
GPIB.Write "SENSel:SWEep:POIN?"  
numpts = GPIB.Read  
  
'Turn continuous sweep off  
GPIB.Write "INITiate:CONTinuous OFF"  
  
'Take a sweep  
GPIB.Write "INITiate:IMMediate;*wai"  
  
'Ask for the Data  
  
'PICK ONE OF THESE LOCATIONS TO READ  
'GPIB.Write "CALCulate:DATA? FDATA" 'Formatted Meas  
'GPIB.Write "CALCulate:DATA? FMEM" 'Formatted Memory  
GPIB.Write "CALCulate:DATA? SDATA" 'Corrected, Complex Meas  
'GPIB.Write "CALCulate:DATA? SMEM" 'Corrected, Complex Memory  
'GPIB.Write "CALCulate:DATA? SCORR1" 'Error-Term Directivity  
  
'Number of values returned per data point
```

```

'q = 1 ' Pick this if reading FDATA or FMEM
q = 2 ' Otherwise pick this

'Parse the data
ReDim datum(q, numpts)
For i = 0 To numpts - 1
    For t = 0 To q - 1
        'Read the Data
        dat = GPIB.Read(20)
        'Parse it into an array
        datum(t, i) = Val(dat)
    Next t
Next i

'PUT THE DATA BACK IN
GPIB.Write "format ascii"

'PICK ONE OF THESE LOCATIONS TO PUT THE DATA
'cmd = "CALCulate:DATA FDATA," 'Formatted Meas
'cmd = "CALCulate:DATA FMEM," 'Formatted Memory
'cmd = "CALCulate:DATA SDATA," 'Corrected, Complex Meas
cmd = "CALCulate:DATA SMEM," 'Corrected, Complex Memory
'cmd = "CALCulate:DATA SCORR1," 'Error-Term Directivity

For i = 0 To numpts - 1
    For t = 0 To q - 1
        If i = numpts - 1 And t = q - 1 Then
            cmd = cmd & Format(datum(t, i))
        Else
            cmd = cmd & Format(datum(t, i)) & ", "
        End If
    Next t
Next i

GPIB.Write cmd
End Sub

```

## Getting and Putting Data

---

This Rocky Mountain Basic example does the following:

1. Takes a sweep, and reads the formatted data trace into an array. The trace is read as a definite length block.
2. Instructs you to remove DUT
3. Downloads the trace back to the analyzer as an definite length block.

### [See Other SCPI Example Programs](#)

```
100 DIM A$(10),Data1(1:51)
110 INTEGER Digits,Bytes
120 !
130 COM /Sys_state/ @Hp87xx,Scode
140 ! Identify I/O Port
150 CALL Iden_port
160 !
170 !
180 OUTPUT @Hp87xx;"SYST:PRES"
190 !
200 OUTPUT @Hp87xx;"CALC:PAR:SEL 'CH1_S11_1'"
210 !
220 ! Set up the analyzer to measure 51 data points.
230 OUTPUT @Hp87xx;"SENS1:SWE:POIN 51;*OPC?"
240 ENTER @Hp87xx;Opc
250 !
260 ! Take a single sweep, leaving the analyzer
270 ! in trigger hold mode.
280 OUTPUT @Hp87xx;"ABOR;:INIT1:CONT OFF;:INIT1;*WAI"
290 !
300 ! Select binary block transfer
310 OUTPUT @Hp87xx;"FORM:DATA REAL,64"
320 !
330 ! Request the channel 1 formatted data array
340 ! from the analyzer.
350 OUTPUT @Hp87xx;"CALC:DATA? FDATA"
360 !
370 ! Turn on ASCII formatting on the I/O path.
380 ! It is needed for reading the header
390 ! information.
400 ASSIGN @Hp87xx;FORMAT ON
410 !
420 ! Get the data header. "A$" will contain the
430 ! "#" character indicating a block data transfer.
440 ! "Digits" will contain the number of characters
```



```

450 ! for the number of bytes value which follows.
460 ENTER @Hp87xx USING "%,A,D";A$,Digits
470 !
480 ! Get the rest of the header. The number of
490 ! bytes to capture in the data array will be
500 ! placed in "Bytes". Note the use of "Digits"
510 ! in the IMAGE string.
515 !
520 ENTER @Hp87xx USING "%,&VAL$(Digits)&"D";Bytes
525 PRINT "HEADER",A$,Digits,Bytes
530 !
540 ! Turn off ASCII formatting on the I/O path;
550 ! it is not needed for transferring binary
560 ! formatted data.
570 ASSIGN @Hp87xx;FORMAT OFF
580 !
590 ! Get the data.
600 ENTER @Hp87xx;Data1(*)
610 !
620 ! Turn on ASCII formatting again.
630 ASSIGN @Hp87xx;FORMAT ON
640 !
650 ! Get the "end of data" character.
660 ENTER @Hp87xx;A$
670 !
680 ! Display the first three numbers in the array.
690 DISP "Trace: ";Data1(1);Data1(2);Data1(3);"..."
700 !
710 ! Use this time to visually compare the
720 ! numbers to the visible data trace.
730 WAIT 5
740 !
750 ! Prompt the operator to disconnect the test
760 ! device and how to continue the program.
770 DISP "Disconnect the test device -- Press Continue"
780 PAUSE
790 !
800 ! Update the display line.
810 DISP "Taking a new sweep...";
820 !
830 ! Take a sweep so the display shows new data.
840 OUTPUT @Hp87xx;":INIT1;*WAI"
850 DISP " Done."
860 WAIT 5
870 !
880 ! Send the header for an indefinite block length
890 ! data transfer.
900 DISP "Downloading saved trace...";
915 ! The first byte '3' indicates the next three digits equal number of transfer
bytes
916 ! The number of transfer bytes equals 8x the number of tracepoints.
920 OUTPUT @Hp87xx;"CALC:DATA FDATA, #3408";

```

```

930  !
940  ! Turn off ASCII formatting.
950  ASSIGN @Hp87xx;FORMAT OFF
960  !
970  ! Send the data array back to the analyzer.
980  OUTPUT @Hp87xx;Data1(*),END
990  !
1000 ! Turn on ASCII formatting again.
1010 ASSIGN @Hp87xx;FORMAT ON
1020 DISP " Done!"
1030 END
1040 !
1050 !*****
1060 ! Iden_port:   Identify io port to use
1070 ! Description: This routines sets up the I/O port address for
1080 !               the SCPI interface.  For "HP 87xx" instruments,
1090 !               the address assigned to @Hp87xx = 800 otherwise,
1100 !               716.
1110 !*****
1120 SUB Iden_port
1130   COM /Sys_state/ @Hp87xx,Scode
1140  !
1150   IF POS(SYSTEM$("SYSTEM ID"),"HP 87")<>0 THEN
1160     ASSIGN @Hp87xx TO 800
1170     Scode=8
1180   ELSE
1190     ASSIGN @Hp87xx TO 716
1200     Scode=7
1210   END IF
1220  !
1230 SUBEND !Iden_port
1240  !

```

## External Test Set Control using SCPI

---

This program demonstrates the use of several External Test Set Control commands.

The SCPI commands in this example are sent over a COM interface using the SCPIStringParser object. You do NOT need a GPIB connection to run this example.

This VBScript (\*.vbs) program can be run as a macro in the PNA. To do this, copy the following code into a text editor file such as Notepad and save it on the PNA hard drive as ExtTS.vbs. [Learn how to setup and run the macro.](#)

```
' Demonstrate some SCPI commands for external testsets.
Dim pna
Set pna = CreateObject("AgilentPNA835x.Application")
Set scpi = pna.ScpIStringParser
' The K64 testset is only usable on a 4-port PNA
If (pna.NumberOfPorts <> 4) Then
MsgBox("This program only runs on 4-port analyzers.")
Else
'If Help is active, show the measurement window and help
scpi.Execute("DISP:ARR TILE")
'Return the list of supported test sets
list=scpi.Execute("SENS:MULT:CATalog?")
MsgBox(list)
'***** K64 *****
'The K64 is connected using the Testset I/O
'connector. There is no handshake information.
'Therefore, a testset need not be connected.
' Load a configuration file.
scpi.Execute("SENS:MULT1:TYPE 'Z5623AK64'")
scpi.Execute("SENS:MULT1:ADDR 0")
'return stuff about the test set
' Returns number of input ports
Inports=scpi.Execute("SENS:MULT1:INCount?")
MsgBox("Input Ports: " & CStr(Inports))
' Returns number of output ports
ports=scpi.Execute("SENS:MULT1:COUNT?")
MsgBox("Output Ports: " & CStr(ports))
' Returns valid output ports for each input port
```

```
For portNum = 1 To Inports
ports=scpi.Execute("SENS:MULT1:PORT" & CStr(portNum) & ":CAT?")
MsgBox("Port " & CStr(portNum) & " catalog: " & (ports))
Next
'Set different port mapping
scpi.Execute("SENS:MULT1:ALLPorts '1 ext R,2 ext R,3 ext R,4 ext R'")
'Return port mapping
portMap=scpi.Execute("SENS:MULT1:ALLPorts?")
MsgBox("Ports will be mapped to " & CStr(portMap))
' Enable external testset control and execute port mapping. This automatically
enables status bar display as well.
scpi.Execute("SENS:MULT1:STATE 1")
MsgBox("Z5623A K64 Enabled")
End If
```

## Transfer Data using GPIB

---

The following RMB examples transfer data to and from a remote PC using the [MMEM:TRANsfer](#) command.

### Transferring data FROM the PNA -- TO a remote PC:

```
30      !
40      !           Set up I/O paths
50      !
60      ! Network analyzer address
70      ASSIGN @Na TO 716
75      !
77      ! File to be stored on local computer
80      ! First time -- need to create the file.
90      ! After file name, number records set to 0 (ignored by WinOS)
95      ! Use "PURGE" command to delete if desired.
100     CREATE "mytestdata.s2p",0
110     ASSIGN @File TO "mytestdata.s2p"
120     !
122     !           TRANSFER the data (download)
123     !
125     ! Analyzer has file 'testdata.s2p' in default directory
130     OUTPUT @Na;":MMEM:TRAN? "testdata.s2p""
135     !
137     ! Now read the bytes coming back from the analyzer in four steps
138     ! (1) Read and dump the first character - '#'
140     ENTER @Na USING "#,A";A$
141     !
142     ! (2) Read the next character which is the number of digits in the file size
150     ENTER @Na USING "#,A";Digit$
160     !
161     ! (3) Use the value of the number of digits to read back the file byte size
170     ! Create query string using this number of digits
180     Img$="#,&Digit$&"A"
190     !
200     ! Byte$ holds the number of bytes in string format
210     ENTER @Na USING Img$;Byte$
220     !
225     ! (4) Read the file contents into a buffer and store the buffer contents to a
local file
230     ! Allocate a buffer for holding the data
240     ALLOCATE Dat$[VAL(Byte$)]
250     !
260     ! Set up a different image for filling the buffer
270     Img$=Byte$&"A"
280     !
290     ! Retrieve the actual file data
300     ENTER @Na USING Img$;Dat$
305     !
307     ! Now save the file locally.
310     OUTPUT @File;Dat$
```

320    END

#### Transferring data FROM the remote PC - TO the PNA:

```
40      !                               Set up I/O paths
50      !
60      ! Network analyzer address
70      ASSIGN @Na TO 716
77      ! File to be retrieved from local computer
78      ASSIGN @File TO "mytestdata.s2p"
79      !
120     !
122     !               TRANSFER the data
123     !
230     ! Allocate a buffer for holding the data
240     ALLOCATE Dat$[26236]
250     !
260     ! Get data from the file and fill Dat$
270     ENTER @File;Dat$
280     !
325     ! Data to be transferred to analyzer file 'testupld.s2p'
325     ! in default directory.
326     !
327     ! A specific block transfer designator must follow the
328     ! file name:
329     !       '#' specifies a block transfer.
330     !       '6' specifies 6 digits to follow.
331     !       '026236' matches the buffer size allocated above
332     !       not counting <NL><END> (new line and end of file).
430     OUTPUT @Na;":MMEM:TRAN ""testupld.s2p",#6026236",Dat$
520     END
```

Last Modified:

26-Jul-2007    Added comments to example

## Establish a VISA Session

---

This Visual Basic program demonstrates how to send a SCPI command using VISA and the Agilent IO libraries. To run this program, you need:

- Your PC and PNA both connected to a LAN (for communicating with each other).
- The SICL and VISA components of Agilent's I/O Libraries software installed on your PC. Both are included when you install the software, unless you already have another vendor's VISA installed. Then specify Full SICL and VISA installation to overwrite the other vendor's VISA.
- The module visa32.bas added to your VB project. After you install VISA, the module will be located at C:/VXIIPNP/WINNT (or equivalent)/INCLUDE/Visa32.bas
- A form with two buttons: cmdRun and cmdQuit.
- Your PC configured to be a VISA LAN Client, and the SICL Server capability enabled on the PNA. See [Configure for VISA and SICL](#)

### [See Other SCPI Example Programs](#)

**Note:** This example is a piece of a larger VISA program that [performs a source power calibration](#).

```
'Session to VISA Default Resource Manager
Private defRM As Long
'Session to PNA
Private viPNA As Long
'VISA function status return code
Private status As Long

Private Sub Form_Load()
defRM = 0
End Sub

Private Sub cmdRun_Click()
' String to receive data from the PNA.
' Dimensioned large enough to receive scalar comma-delimited values
' for 21 frequency points (20 ASCII characters per point)
Dim strReply As String * 420

' Open the VISA default resource manager
status = viOpenDefaultRM(defRM)
If (status < VI_SUCCESS) Then HandleVISAError

' Open a VISA session (viPNA) to the SICL LAN server
' at "address 16" on the PNA pointed to by the "GPIB0"
' VISA LAN Client on this PC.
' CHANGE GPIB0 TO WHATEVER YOU PNA IS SET TO
status = viOpen(defRM, "GPIB0::16::INSTR", 0, 0, viPNA)
If (status < VI_SUCCESS) Then HandleVISAError

' Need to set the VISA timeout value to give all our calls to
```

```

' myGPIBRead sufficient time to complete before a timeout
' error occurs.
' For this example, let's try setting the limit to
' 30000 milliseconds (30 seconds).
status = viSetAttribute(viPNA, VI_ATTR_TMO_VALUE, 30000)
If (status < VI_SUCCESS) Then HandleVISAError

' Preset the PNA
status = myGPIBWrite(viPNA, "SYST:PRES")
If (status < VI_SUCCESS) Then HandleVISAError

' Print the data using a message box
MsgBox strReply
End Sub

Private Sub cmdQuit_Click()
' Close the resource manager session (which also closes
' the session to the PNA).
If defRM <> 0 Then Call viClose(defRM)

' End the program
End
End Sub

Private Function myGPIBWrite(ByVal viHandle As Long, ByVal strOut As String) As Long
' The "+ Chr$(10)" appends an ASCII linefeed character to the
' output, for terminating the write transaction.
myGPIBWrite = viVPrintf(viHandle, strOut + Chr$(10), 0)
End Function

Private Function myGPIBRead(ByVal viHandle As Long, strIn As String) As Long
myGPIBRead = viVScanf(viHandle, "%t", strIn)
End Function

Sub HandleVISAError()
Dim strVisaErr As String * 200
Call viStatusDesc(defRM, status, strVisaErr)
MsgBox "*** Error : " + strVisaErr, vbExclamation
End
End Sub

```



## Status Reporting using SCPI

---

This Visual Basic program demonstrates two methods of reading the analyzer's status registers:

- Polled Bit Method - reads the Limit1 register continuously.
- SRQ Method - enables an interrupt of the program when bit 6 of the status byte is set to 1. The program then queries registers to determine if the limit line failed.

To run this program, you need:

- An established [GPIB interface connection](#)
- A form with two buttons: Poll and SRQ Method
- A means of causing the limit line to fail, assuming it passes initially.

```
Private Sub Poll_Click()  
' POLL THE BIT METHOD  
' Clear status registers  
GPIB.Write "*CLS"  
  
'Loop FOREVER  
Do  
    DoEvents  
    GPIB.Write "STATus:QUESTionable:LIMit1:EVENT?"  
    onn = GPIB.Read  
Loop Until onn = 2  
  
MsgBox "Limit 1 Failed "  
End Sub  
  
Private Sub SRQMethod_Click()  
'SRQ METHOD  
GPIB.Write "SYSTem:PRESet"  
GPIB.Write "CALCulate:PARAmeter:SElect 'CH1_S11_1'"  
'slow down the trace  
GPIB.Write "SENS:BWID 150"  
  
'Setup limit line  
GPIB.Write "CALC:LIM:DATA 2,3e9,6e9,-2,-2"  
GPIB.Write "CALC:LIMit:DISP ON"  
GPIB.Write "CALC:LIMit:STATe ON"  
  
' Clear status registers.  
GPIB.Write "*CLS;*wai"  
' Clear the Service Request Enable register.  
GPIB.Write "*SRE 0"  
' Clear the Standard Event Status Enable register.  
GPIB.Write "*ESE 0"
```

```

' Enable questionable register, bit(10) to report to the status byte.
GPIB.Write "STATus:QUESTionable:ENABle 1024"

' Enable the status byte register bit3 (weight 8) to notify controller
GPIB.Write "*SRE 8"

' Enable the onGPIBNotify event
GPIB.NotifyMask = cwGPIBRQS
GPIB.Notify
End Sub

-----
Private Sub GPIB_OnGPIBNotify(ByVal mask As Integer)
' check to see what failed
' was it the analyzer?
GPIB.Write "*STB?"
onn = GPIB.Read
If onn <> 0 Then
' If yes, then was it the questionable register?
GPIB.Write "STATus:QUESTionable:EVENT?"
onn = GPIB.Read
' Determine if the limit1 register, bit 8 is set.
If onn = 1024 Then
'if yes, then was it trace 1?
GPIB.Write "STAT:QUES:LIMIT1:EVEN?"
onn = GPIB.Read
If onn = 2 Then MsgBox ("Limit Line1 Failed")
End If
End If
End Sub

```

## Create a Custom Power Meter Driver

---

This topic requires that you have a working knowledge of Visual Basic.

This topic will help you create your own power meter driver for use with Source Power Calibration on the PNA. If you are using an Agilent Power Meter to perform a Source Power Calibration, you do NOT need to create your own driver.

Your Power Meter driver will be created from a template written in Visual Basic using VISA over the GPIB bus.

**Note:** This procedure applies to Visual Basic 6.0. Applicability to Visual Basic .NET has not yet been investigated.

- [Prepare Template Files](#)
- [Modify Template Files](#)
- [Compile, Copy, and Register, Your New Driver](#)
- [Test Your new Driver](#)

### Other SCPI Example Programs

#### Prepare Template Files

1. Copy all the files from the PNA hard drive C:/Program Files/Agilent/Network Analyzer/Automation/Power Meter Driver Template folder, to a folder on your development PC.
2. In Visual Basic click **File**, then **Open Project...**, find **MyPowerMeter.vbp** (a file you copied from the PNA). Click **Open**. This is a VB ActiveX EXE template, which you will fill in to become your driver.
3. Click **Project**, then **MyPowerMeter Properties**. Click the **General** tab.
4. Overwrite the Project Name with a name of your own choosing. This will be the name of your driver's type library (also the default name of your exe).

**Note** If the name of your exe does not match the VB Project Name with which it was compiled, registration of the exe on the PNA will not succeed.

5. Set the Project Description. After building your driver if you wish to test it using VB, this is the string that will show up in the VB References list of your test project, and also in the lower pane of the VB Object Browser.
6. Set the Thread Pool size to 1 thread.
7. Click **OK** to close the project properties dialog.
8. From the VB **Project** menu, click **References...** Ensure that **Agilent PNA Power Meter 1.0 Type Library** and **VISA Library** are checked. Click **OK**.

**Note:** Agilent's implementation of VISA is installed as part of the Agilent I/O Libraries on the PNA. For help on

VISA, go to the Windows Start button on your PNA, select Programs, Agilent IO Libraries, VISA Help.

---

## Modify Template Files

From Visual Basic **View** menu click **Project Explorer**. Expand the **Modules** and **Class Modules** folders. Ensure there is one module (WinAPI) and one class module (PowerMeter).

Let's look at the WinAPI module first.

1. In the **Project Explorer** window, click **WinAPI**.
2. From the **View** menu click **Code**.

There is only one line of code you should need to modify in this module: the value of the string constant named sIDSEARCH. The comments preceding the declaration of that string describe how to change it. The rest of this module contains functions which will use the Microsoft Windows API to insure proper registration of your driver on the PNA. If you know of other Windows API functions you feel might be helpful to call from within your PowerMeter class module (to help in formatting data, for example), this module would be the place to declare them.

Now let's look at the class module.

1. In the Project Explorer window, click **PowerMeter**.
2. From the **View** menu click **Properties Window**. The **Instancing** property must be set to MultiUse. This allows other applications to create objects from this class, such that one instance of your driver EXE can supply more than one such object at a time.
3. From the **View** menu click **Code**.

Do NOT modify the Interfaces to IPowerMeter subroutines and functions. PNA source power cal expects to find these interfaces as they are currently defined.

The only members that you need to supply code to are those containing “**Your code here**” comments.

In addition, comments have been provided at the beginning of each member to describe the information that member needs to be read from or written to the power meter.

To get an idea of how communicate with the power meter using the VISA functions **viWrite** and **viRead**, examine the code which has been implemented for you in IPowerMeter\_Connect, IPowerMeter\_QueryMeter, and IPowerMeter\_WriteMeter.

---

## Compile, Copy, and Register Your New Driver

When your driver is ready to run, you will first need to compile it into an EXE.

From the File menu select **Make exe**.

After compiling, the following will instruct VB to use the same ID (GUID) every time you re-compile your project.

1. From the **Project** menu, click **PowerMeter Properties**.
2. On the **Component** tab, select **Binary Compatibility** and click ...
3. Browse to and select your project EXE. Click **Open**.

4. Click **OK** to close **Project Properties**.
5. Save your project.
6. Copy your driver EXE file to a folder on your PNA (do NOT use C:/Program Files/Agilent/Network Analyzer/Automation/Power Meter Driver Template folder).
7. Run the EXE file. A message box will pop up reporting whether or not registration was successful. If not successful, it will make a suggestion on what to fix.

When your driver is properly registered, PNA Source Power Cal should be able to associate it with the ID string of your power meter.

---

### Test Your Power Meter Driver

We have also provided a Visual Basic project to test your new Power Meter driver. This project individually calls every IPowerMeter method and property in your driver to verify that it performs correctly. Before running the test your PC and PNA must be configured to communicate using DCOM.

1. Connect your PC and the PNA to LAN.
2. Add your PC logon to the PNA. Both logons and password must match to communicate using DCOM. See [Additional PNA users](#).
3. Configure your driver using DCOM Config on the PNA. This will give you permission to launch and access the driver. See [Configure for COM-DCOM Programming](#).

### Modify the Test Project

1. In Visual Basic click **File**, then **Open Project...**, find **MyPowerMeterTest.vbp** (a file you copied from the PNA). Click **Open**.
2. From the **Project** menu, click **References...** From the list, find and check your new Power Meter Driver. (It should have been registered on your PC when you successfully made your driver EXE.) Click **OK**.
3. From the **View** menu click **Code**.
4. Modify the **CreateObject** line as follows:  
Replace **MyPowerMeter** with the Project Name that you chose for your driver  
Replace **MyPNA** with the Computer Name of your PNA.  
For example:

```
Set PowerMeterObj = CreateObject("AcmeBrand.PowerMeter", "AGILENT-PNA123")
```

(This assumes that you kept **PowerMeter** as class module name in your driver.)

### Run the Test Project

Ensure your power meter is connected to the PNA with a GPIB cable.

Put the PNA in system controller mode:

1. From the PNA **System** menu point to **Configure** then click **SICL/GPIB**.
2. In the GPIB box click **System Controller**.

Run the test project. If there are no errors, the driver is created successfully. If there are errors, try to figure out what went wrong and fix it. Then re-compile, re-copy the .exe to the PNA, and re-run the test. You should not need to re-register the driver or re-modify the test program.

## GPIB Pass-Through Example

---

The SCPI [SYSTem](#) commands used in this example allow you to send GPIB commands to another GPIB device through the PNA. The other GPIB device cannot be connected to the GPIB bus through the PNA rear panel if the PNA is being controlled by a remote PC using that connector. The other device would typically be connected to the PNA using a [USB/GPIB interface](#).

This VB Script example uses the COM [SCPIStringParser](#) object. However, this is not critical to the use of these commands; they can be sent using the normal syntax of your programming environment. Using the SCPIStringParser over LAN allows you to communicate with GPIB devices without requiring your remote PC to have a GPIB interface card installed.

Although this method of pass-through works for most applications, there are a couple of limitations:

- All data is transferred using ASCII format. Therefore, transferring large blocks of data is very slow.
- Only read and write functions are possible. Service Interrupts are not supported.

[See Other SCPI Example Programs](#)

---

```
option explicit
dim app
set app = CreateObject("AgilentPNA835x.Application")

dim p
set p = app.ScpiStringParser

' Open a new GPIB session on Bus:2 Device:14 Timeout: 100ms
p.Parse "SYST:COMM:GPIB:RDEV:OPEN 2,14,100"
dim handleAsStr

' Retrieve the handle (ID number)
handleAsStr = p.Parse ("SYST:COMM:GPIB:RDEV:OPEN?")

' Convert the handle to an integer
dim handleAsInt
handleAsInt = CInt(handleAsStr)

' Send the "*IDN?" query
p.Parse "SYST:COMM:GPIB:RDEV:WRITE " & handleAsInt & ", '*IDN?'"

' Read its results
dim idn
idn = p.Parse("SYST:COMM:GPIB:RDEV:READ? " & handleAsInt)
msgbox idn

' Close the GPIB session
p.Parse "SYST:COMM:GPIB:RDEV:CLOSE " & handleAsInt
```

## PNA as Controller and Talker / Listener

---

This Visual Basic Program uses VISA to do the following:

- Control the PNA using a VISA LAN Client interface on the PNA.
- Control another instrument using the PNA as GPIB controller.
- Queries both the analyzer and other instrument to identify themselves with \*IDN?

**Note:** This program can be modified to work from a remote PC to control both instruments. In that case, set up the PNA to be a talker/listener.

To run this program, you need to do the following:

- Add module **visa32.bas** to the VB project. It is located on the analyzer at C:/Program Files/HP/VXIPNP/WINNT/Include/VISA32.bas
- [Configure the PNA for VISA / SICL](#)
- Set up the PNA to be GPIB system controller.
  1. On the **System** menu, point to **Configure**. Click **SICL / GPIB**
  2. Click **System Controller**
- Connect another instrument to the analyzer through a GPIB cable with Primary address of 13 on GPIB0 interface

### [See Other SCPI Example Programs](#)

```
Sub main()  
  
'This application run from onboard the PNA  
'can control both the PNA and another GPIB instrument.  
'  
'To run this program the module visa32.bas must be added  
'to the project.  
  
'VISA function status return code  
Dim status As Long  
'Session to Default Resource Manager  
Dim defRM As Long  
'Session to instrument  
Dim viPNA As Long  
'Session to other GPIB instrument  
Dim viInstrument As Long  
'String to hold results  
Dim strRes As String * 200  
On Error GoTo ErrorHandler  
  
status = viOpenDefaultRM(defRM)
```



```

If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Open the session to the PNA
status = viOpen(defRM, "GPIB1::16::INSTR", 0, 0, viPNA)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Ask for the PNA's ID.
status = viVPrintf(viPNA, "*IDN?" + Chr$(10), 0)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Read the ID as a string.
status = viVScanf(viPNA, "%t", strRes)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler
'Display the results
MsgBox "PNA is: " + strRes

'Open the session to the other instrument
status = viOpen(defRM, "GPIB0::13::INSTR", 0, 0, viInstrument)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Ask for the instrument's ID.
status = viVPrintf(viInstrument, "*IDN?" + Chr$(10), 0)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Read the ID as a string.
status = viVScanf(viPNA, "%t", strRes)
If (status < VI_SUCCESS) Then GoTo VisaErrorHandler

'Display the results
MsgBox "Other instrument is: " + strRes
' Close the resource manager session (which closes everything)
Call viClose(defRM)
End

ErrorHandler:
'Display the error message
MsgBox "*** Error : " + Error$, MB_ICONEXCLAMATION
End

VisaErrorHandler:
Dim strVisaErr As String * 200
Call viStatusDesc(defRM, status, strVisaErr)
MsgBox "*** Error : " + strVisaErr

End
End Sub

```

## Socket Client

---

The following C# example demonstrates how to send SCPI commands to the PNA via a TCP socket connection and how to use a TCP 'control' connection. If the command is a query, the program will read the instrument's response. You can add or replace the SCPI commands in this program with your own.

[Learn how to enable Sockets communication on the PNA.](#)

For both of the following methods, first copy the example text below into a Notepad file and name it SocketClient.cs.

### To run using Microsoft Visual Studio 2003 or 2005

1. From the Visual Studio **File** menu, select **New**, then **Project**.
2. In the **New Project** window, select the following items (noting the location of the file folder it is creating for you) then click **OK**.
  - Project Type: Visual C#
  - Template: Console Application
  - Project Name: SocketClient
1. Copy SocketClient.cs into the folder that was created in the previous step.
2. In the Solution Explorer window pane, right-click **Class1.cs** (if Visual Studio 2003) or **Program.cs** (if Visual Studio 2005). Select **Delete** to delete that file.
3. In the Solution Explorer, right-click **SocketClient** , and select **Add**, then **Existing Item....**
4. Browse to select **SocketClient.cs** and click **OK**.

You should then be able to build the project, and test the resulting **SocketClient.exe** from a command prompt (shell) window.

### To run using Mono

Mono is a cross-platform version of .NET. You can download a free version of Mono at <http://www.mono-project.com>. Once downloaded and installed:

1. Run the Mono command prompt (shell) window.
2. Navigate to the directory where the example SocketClient.cs is stored.
3. Type: **MCS SocketClient.cs** (builds the .exe and saves in that same folder.)
4. Type **mono SocketClient.exe** <PNA name or IP address>

This example was compiled and tested successfully with Mono version 1.1.13. It was run on a PC using the Red

Hat version 9.0 distribution of the Linux operating system. It was also run on a PC using Windows XP. This program has not been tested with other versions of Mono, or on other operating systems.

### To run with Agilent T&M Toolkit

Agilent T&M Toolkit 2.0 is the first version to support communication using Sockets.

Use the following to address the Sockets port: **TCPIP0::<PNA name or IP address>::5025::SOCKET**

```
using System;
using System.Net;
using System.Net.Sockets;

// This C# "Console Application" example program demonstrates SCPI
// communication with an Agilent TCP socket-enabled instrument that
// supports socket "control connections" (such as PNA network analyzers,
// which have support for control connections in their socket
// implementation as of PNA Firmware A.08.33.01).
namespace CSharpSocketClient
{
    /// <summary>
    /// The class supporting the main entry point for the application.
    /// </summary>
    class MainClass
    {
        static AsyncCallback m_pCallbackFunc;
        static string m_AsyncReply;

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static int Main(string[] args)
        {
            try
            {
                if (args.Length != 1)
                {
                    Console.WriteLine("");
                }
            }
        }
    }
}
```

```

        Console.WriteLine("Usage -- with Microsoft's .NET runtime:");
        Console.WriteLine("SocketClient servernameoraddress");
        Console.WriteLine("Example: SocketClient 192.168.0.1");
        Console.WriteLine("");
        Console.WriteLine("Usage -- with Mono's (www.mono-project.com) .NET
runtime:");
        Console.WriteLine("mono SocketClient.exe servernameoraddress");
        Console.WriteLine("Example: mono SocketClient.exe 192.168.0.1");
        return 1;
    }

    string server = args[0];
    Int32 port = 5025; // default socket port number for the PNA

    // Create the primary client socket instance
    Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

    // Get the DNS IP addresses associated with the instrument.
    // (if 'server' string contains the IP address rather than DNS name, this
still works)
    IPHostEntry hostInfo = Dns.Resolve(server);
    IPAddress[] IPAddresses = hostInfo.AddressList;
    if (IPAddresses.GetLength(0) < 1)
        return 1;

    // Create an endpoint to use for opening the socket connection
    IPEndPoint endpoint1 = new IPEndPoint (IPAddresses[0], port);
    // Open the connection to the server instrument
    client.Connect(endpoint1);
    if(!client.Connected)
        return 1;

    // Query the instrument's ID string.
    string id = Parse(client, "*IDN?");

    // Clear the instrument's Status Byte

```

```

Parse(client, "*CLS");

// Enable for the OPC bit (bit 0, which has weight 1) in the instrument's
// Event Status Register, so that when that bit's value transitions from 0
to 1
// then the Event Status Register bit in the Status Byte (bit 5 of that
byte)
// will become set.
Parse(client, "*ESE 1");

// Enable for bit 5 (which has weight 32) in the Status Byte to generate an
// SRQ when that bit's value transitions from 0 to 1.
Parse(client, "*SRE 32");

// Ask the instrument for the number of a port on which a 'control'
// socket connection can be opened.
string controlPortNumStr = Parse(client, "SYSTem:COMMunicate:TCPIP:CONTROL?");
Int32 controlPortNum = System.Convert.ToInt32(controlPortNumStr);

// Create the client "control connection" socket instance
Socket controlClient = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

// Create an endpoint to use for opening the control connection
IPEndPoint endpoint2 = new IPEndPoint(IPAddresses[0], controlPortNum);
// Connect to the server instrument via the port number that was returned
by the instrument.
controlClient.Connect(endpoint2);
if(!controlClient.Connected)
    return 1;

// Start the control connection listening for an SRQ message.
BeginListeningForAsyncReply(controlClient);

// Now send a preset command to the instrument, accompanied by '*OPC' such
// that when that operation is complete an SRQ event will be generated
// which posts the Status Byte message on the control connection.

```

```

Parse(client, "SYSTem:PRESet;*OPC");

// Normally at this point you would want to have your program do other
things
// right here until the SRQ callback occurs, instead of just idling here
waiting
// for it.
do {} while (m_AsyncReply == null);

// Now that the SRQ has occurred, we can issue a Device Clear via the
control connection.
Parse(controlClient, "DCL");

// The instrument will respond back with "DCL" (and linefeed character
appended
// on the end) via the control connection when it has finished processing
the
// Device Clear request. Note that this 'Response' method uses the
synchronous
// form of 'Receive', so it could potentially time out if the instrument
were
// to take a long time to process the Device Clear. So alternatively the
// 'BeginListeningForAsyncReply' could be used for this instead of
'Response'.
string deviceClearResponse = Response(controlClient);

// Close both of the socket client sessions.
controlClient.Close();
client.Close();
}
catch (ArgumentNullException e)
{
    Console.WriteLine("ArgumentNullException: {0}", e);
}
catch (SocketException e)
{
    Console.WriteLine("SocketException: {0}", e);
}

```

```

    Console.WriteLine("/n Press Enter to continue...");
    Console.Read();
    return 0;
}

static string Parse(Socket client, string command)
{
    // Translate the passed command into ASCII and store it as a Byte array.
    Byte[] data = System.Text.Encoding.ASCII.GetBytes(command);
    // Send the command to the socket-enabled instrument.
    client.Send(data);
    // Has to be followed by a linefeed character as terminator.
    Byte[] lf = {(Byte)'\n'};
    client.Send(lf);
    Console.WriteLine("Sent: {0}", command);
    // If the message was a query (involved a question mark), receive the
    instrument response.
    if (command.IndexOf("?") >= 0)
    {
        return Response(client);
    }
    return "";
}

static string Response(Socket client)
{
    // Buffer to store the response bytes.
    // For simplicity of this example, we allocate just for a 256-byte maximum
    // response size.
    Byte[] data = new Byte[256];
    // Read the batch of response bytes.
    Int32 byteCount = client.Receive(data);
    // String to store the response ASCII representation.
    string responseData = System.Text.Encoding.ASCII.GetString(data, 0,
byteCount);
    Console.WriteLine("Received: {0}", responseData);
    return responseData;
}

```

```

}

static void BeginListeningForAsyncReply(Socket client)
{
    if (m_pCallbackFunc == null)
    {
        m_pCallbackFunc = new AsyncCallback(OnMessageReceived);
    }
    SocketPacket socPkt = new SocketPacket();
    socPkt.thisSocket = client;
    // Start asynchronously listening for a response from this client
    IAsyncResult result = client.BeginReceive (socPkt.data,
        0, socPkt.data.Length,
        SocketFlags.None,
        m_pCallbackFunc,
        socPkt);
}

class SocketPacket
{
    public Socket thisSocket;
    // For simplicity of this example, we allocate just for a 256-byte maximum
    // response size.
    public Byte[] data = new Byte[256];
}

static void OnMessageReceived(IAsyncResult asyn)
{
    SocketPacket socPkt = (SocketPacket)asyn.AsyncState ;
    Int32 byteCount = socPkt.thisSocket.EndReceive(asyn);
    m_AsyncReply = System.Text.Encoding.ASCII.GetString(socPkt.data, 0,
byteCount);
    Console.WriteLine("Received: {0}", m_AsyncReply);
}
}
}

```



---

Last Modified:

28-Aug-2008    Updated with new TCPIP command

## GPIB Fundamentals

---

The General Purpose Interface Bus (GPIB) is a system of hardware and software that allows you to control test equipment to make measurements quickly and accurately. This topic contains the following information:

- [The GPIB Hardware Components](#)
- [The GPIB / SCPI Programming Elements](#)
- [Specifications](#)
- [GPIB Interface Capability Codes](#)

**Note:** All of the topics related to programming assume that you already know how to program, preferably using a language that can control instruments.

### Other Topics about GPIB Concepts

## The GPIB Hardware Components

The system bus and its associated interface operations are defined by the IEEE 488 standard. The following sections list and describe the main pieces of hardware in a GPIB system:

Early PNA models had only ONE GPIB connector. These models could control other GPIB devices using one of, or a combination of, the following methods:

- Use the SCPI [SYST:COMM:GPIB:RDEV](#): commands.
- Use VISA or SICL over LAN to accomplish this. See an [example](#).
- Use [USB / GPIB Interface](#)

**Note:** Current PNA models have dedicated Controller and Talker/Listener GPIB ports. [See how to configure these ports.](#)

### Controllers

Controllers specify the instruments that will be the talker and listener in a data exchange. The controller of the bus must have a GPIB interface card to communicate on the GPIB.

- The **Active Controller** is the computer or instrument that is currently controlling data exchanges.
- The **System Controller** is the only computer or instrument that can take control and give up control of the GPIB to another computer or instrument, which is then called the active controller.

### Talker / Listener Instruments and GPIB Addresses

- **Talkers** are instruments that can be addressed to send data to the controller.

- **Listeners** are instruments that can be addressed to receive a command, and then respond to the command. All devices on the bus are required to listen.

Every GPIB instrument must have its own unique address on the bus. The PNA address (default = 716) consists of two parts:

1. **The Interface select code** (typically 7) indicates which GPIB port in the system controller is used to communicate with the device.
2. **The primary address** (16) is set at the factory. You can change the primary address of any device on the bus to any number between 0 and 30. To change the analyzer address click [System / Configure / SICL-GPIB](#).

**A secondary address** is sometimes used to allow access to individual modules in a modular instrument system, such as a VXI mainframe. The PNA does NOT have a secondary address.

## Cables

GPIB Cables are the physical link connecting all of the devices on the bus. There are eight data lines in a GPIB cable that send data from one device to another. There are also eight control lines that manage traffic on the data lines and control other interface operations.

You can connect instruments to the controller in any arrangement with the following limitations:

- Do not connect more than 15 devices on any GPIB system. This number can be extended with the use of a bus extension.
- Do not exceed a total of 20 meters of total cable length or 2 meters per device, whichever is less.
- Avoid stacking more than three connectors on the back panel of an instrument. This can cause unnecessary strain on the rear-panel connector.

## The GPIB / SCPI Programming Elements

The following software programming elements combine to become a GPIB program:

- [GPIB / SCPI Commands](#)
- [Programming Statements](#)
- [Instrument Drivers](#)

## GPIB Commands

The GPIB command is the basic unit of communication in a GPIB system. The analyzer responds to three types of GPIB commands:

### 1. IEEE 488.1 Bus-management Commands

These commands are used primarily to tell some or all of the devices on the bus to perform certain interface operations.

All of the functions that can be accomplished with these commands can also be done with IEEE 488.2 or SCPI commands. Therefore, these commands are not documented in this Help system. For a complete list of IEEE 488.1 commands refer to the IEEE 488 standard. **Examples** of IEEE 488.1 Commands

- **CLEAR** – Clears the bus of any pending operations
- **LOCAL** – Returns instruments to local operation

## 2. IEEE 488.2 Common Commands

These commands are sent to instruments to perform interface operations. An IEEE 488.2 common command consists of a single mnemonic and is preceded by an asterisk ( \* ). Some of the commands have a query form which adds a "?" after the command. These commands ask the instrument for the current setting. See a complete list of the [Common Commands](#) that are recognized by the analyzer. **Examples** of IEEE 488.2 Common Commands

- **\*OPC** – Operation Complete
- **\*RST** – Reset
- **\*OPT?** – Queries the option configuration

## 3. SCPI Commands

The Standard Commands for Programmable Instruments (SCPI) is a set of commands developed in 1990. The standardization provided in SCPI commands helps ensure that programs written for a particular SCPI instrument are easily adapted to work with a similar SCPI instrument. SCPI commands tell instruments to do device specific functions. For example, SCPI commands could tell an instrument to make a measurement and output data to a controller. **Examples** of SCPI Commands:

```
CALCULATE:AVERAGE:STATE ON
```

```
SENSE:FREQUENCY:START?
```

For more information on SCPI:

- [The Rules and Syntax of SCPI Commands](#) provides more detail of the SCPI command structure.
- [SCPI Command Tree](#) is a complete list of the SCPI commands for the analyzer

## Programming Statements

SCPI commands are included with the language specific I/O statements to form program statements. The programming language determines the syntax of the programming statements. SCPI programs can be written in a variety of programming languages such as VEE, HP BASIC, or C++. **Example** of a Visual Basic statement:

- `GPIB.Write "SOURCE:FREQUENCY:FIXED 1000 MHz"`

## Note about examples

## Instrument Drivers

Instrument drivers are subroutines that provide routine functionality and can be reused from program to program. GPIB industry leaders have written standards for use by programmers who develop drivers. When programmers write drivers that comply with the standards, the drivers can be used with predictable results. To comply with the

standard, each instrument driver must include documentation describing its functionality and how it should be implemented.

## GPIO Specifications

**Interconnected devices** - Up to 15 devices (maximum) on one contiguous bus.

**Interconnection path** - Star or linear (or mixed) bus network, up to 20 meters total transmission path length or 2 meters per device, whichever is less.

**Message transfer scheme** - Byte-serial, bit-parallel, asynchronous data transfer using an interlocking 3-wire handshake.

**Maximum data rate** - 1 megabyte per second over limited distances, 250 to 500 kilobytes per second typical maximum over a full transmission path. The devices on the bus determine the actual data rate.

**Address capability** - Primary addresses, 31 Talk and 31 Listen; secondary addresses, 961 Talk and 961 Listen. There can be a maximum of 1 Talker and up to 14 Listeners at a time on a single bus. See also previous section on [GPIO addresses](#).

## GPIO Interface Capability Codes

The IEEE 488.1 standard requires that all GPIO compatible instruments display their interface capabilities on the rear panel using codes. The codes on the analyzer, and their related descriptions, are listed below:

SH1 full source handshake capability

AH1 full acceptor handshake capability

T6 basic talker, serial poll, no talk only, unaddress if MLA (My Listen Address)

TEO no extended talker capability

L4 basic listener, no listen only, unaddress if MTA (My Talk Address)

LEO no extended listener capability

SR1 full service request capability

RL1 full remote / local capability

PPO **no parallel poll capability**

DC1 full device clear capability

DT1 full device trigger capability

C1 system controller capability

C2 send IFC (Interface Clear) and take charge controller capability

C3 send REN (Remote Enable) controller capability

#### C4 respond to SRQ (Service Request)

---

Last Modified:

30-Oct-2008    Removed legacy content

## The Rules and Syntax of SCPI

---

Most of the commands used for controlling instruments on the GPIB are SCPI commands. The following sections will help you learn to use SCPI commands in your programs.

[Branches on the Command Tree](#)

[Command and Query](#)

[Multiple Commands](#)

[Command Abbreviation](#)

[Bracketed \(Optional\) Keywords](#)

[Vertical Bars \(Pipes\)](#)

[MIN and MAX Parameters](#)

### [Other Topics about GPIB Concepts](#)

---

## Branches on the Command Tree

All major functions on the analyzer are assigned keywords which are called ROOT commands. (See GPIB Command Finder for a list of SCPI root commands). Under these root commands are branches that contain one or more keywords. The branching continues until each analyzer function is assigned to a branch. A root command and the branches below it is sometimes known as a subsystem.

For example, under [SOURCE:POWer](#) are several branch commands.

Sometimes the same keyword, such as `STATE`, is used in several branches of the command tree. To keep track of the current branch, the analyzer's command parser uses the following rules:

- **Power On and Reset** - After power is cycled or after `*RST`, the current path is set to the root level commands.
- **Message Terminators** - A message terminator, such as a `<NL>` character, sets the current path to the root command level. Many programming language output statements send message terminators automatically. Message terminators are described in [Sending Messages to the Analyzer](#).
- **Colon (:)** - When a colon is between two command keywords, it moves the current path down one level in the command tree. For example, the colon in `:SOURCE:POWER` specifies that `POWER` is one level below `SOURCE`. When the colon is the first character of a command, it specifies that the following keyword is a root level command. For example, the colon in `:SOURCE` specifies that `source` is a root level command.

**Note:** You can omit the leading colon if the command is the first of a new program line. For example, the following two commands are equivalent:

```
SOUR:POW:ATT:AUTO
:SOUR:POW:ATT:AUTO
```

- **<WSP>** - Whitespace characters, such as <tab> and <space>, are generally ignored. There are two important exceptions:
  - Whitespace inside a keyword, such as :CALCULATE, is not allowed.
  - Most commands end with a parameter. You must use whitespace to separate these ending parameters from commands. **Always refer to the command documentation.** In the following example, there is whitespace between STATE and ON.

```
CALCULATE1:SMOOTHING:STATE ON
```

- **Comma (,)** - If a command requires more than one parameter, you must separate adjacent parameters using a comma. For example, the SYSTEM:TIME command requires three values to set the analyzer clock: one for hours, one for minutes, and one for seconds. A message to set the clock to 8:45 AM would be SYSTEM:TIME 8,45,0. Commas do not affect the current path.
- **Semicolon (;)** - A semicolon separates two commands in the same message without changing the current path. See [Multiple Commands](#) later in this topic.
- **IEEE 488.2 Common Commands** - Common commands, such as \*RST, are not part of any subsystem. An instrument interprets them in the same way, regardless of the current path setting.

## Command and Query

A SCPI command can be an Event command, Query command (a command that asks the analyzer for information), or both. The following are descriptions and examples of each form of command. GPIB Command Finder lists every SCPI command that is recognized by the analyzer, and its form.

### Form

**Event commands** - cause an action to occur inside the analyzer.

**Query commands** - query only; there is no associated analyzer state to set.

**Command and query** - set or query an analyzer setting. The query form appends a question mark (?) to the set form

### Examples

```
:INITIATE:IMMEDIATE
```

```
:SYSTem:ERRor?
```

```
:FORMat:DATA ! Command
:FORMat:DATA? ! Query
```

## Multiple Commands

You can send multiple commands within a single program message. By separating the commands with semicolons the current path does not change. The following examples show three methods to send two commands:

### 1. Two program messages:

```
SOURCE:POWER:START 0DBM
SOURCE:POWER:STOP 10DBM
```



2. **One long message.** A colon follows the semicolon that separates the two commands causing the command parser to reset to the root of the command tree. As a result, the next command is only valid if it includes the entire keyword path from the root of the tree:

```
SOURCE:POWER:START 0DBM; :SOURCE:POWER:STOP 10DBM
```

3. **One short message.** The command parser keeps track of the position in the command tree. Therefore, you can simplify your program messages by including only the keyword at the same level in the command tree.

```
SOURCE:POWER:START 0DBM;STOP 10DBM
```

## Common Commands and SCPI Commands

You can send Common commands and SCPI commands together in the same message. (For more information on these types of commands see [GP-IB Fundamentals](#).) As in sending multiple SCPI commands, you must separate them with a semicolon.

**Example** of Common command and SCPI commands together

```
*RST;SENSE:FREQUENCY:CENTER 5MHZ;SPAN 100KHZ
```

## Command Abbreviation

Each command has a long form and an abbreviated short form. The syntax used in this Help system use uppercase characters to identify the short form of a particular keyword. The remainder of the keyword is lower case to complete the long form.

```
SOUR - Short form  
SOURce - Long form
```

Either the complete short form or complete long form must be used for each keyword. However, the keywords used to make a complete SCPI command can be a combination of short form and long form.

The following is **unacceptable** - The first three keywords use neither short or long form.

```
SOURc:PowE:Atten:Auto on
```

The following is **acceptable** - All keywords are either short form or long form.

```
SOUR:POWer:ATT:AUTO on
```

In addition, the analyzer accepts lowercase and uppercase characters as equivalent as shown in the following equivalent commands:

```
source:POW:att:auto ON  
Source:Pow:Att:Auto on
```

## Optional [Bracketed] Keywords

You can omit some keywords without changing the effect of the command. These optional, or default, keywords are used in many subsystems and are identified by brackets in syntax diagrams.

**Example** of Optional Keywords

The `HCOPY` subsystem contains the optional keyword `IMMediate` at its first branching point. Both of the following commands are equivalent:

```
"HCOpy:IMMEDIATE"
```

```
"HCOpy"
```

The syntax in this Help system looks like this:

```
HCOpy[:IMMediate]
```

## Vertical Bars | Pipes

Vertical bars, or "pipes", can be read as "**or**". They are used in syntax diagrams to separate alternative parameter options.

**Example** of Vertical Bars:

```
SOURce:POWer:ATTenuation:AUTO <on|off>
```

Either ON or OFF is a valid parameter option.

## MIN and MAX Parameters

The special form parameters "MINimum" and "MAXimum" can be used with commands that specify single frequency (Hz) and time (seconds) as noted in the command documentation. **Note:** Also with these commands, kHz, MHz, and GHz are accepted as suffixes/units.

The short form (min) and long form (minimum) of these two keywords are equivalent.

- **MAX**imum refers to the largest value that the function can currently be set to
- **MIN**imum refers to the smallest value that the function can currently be set to.

**For example**, the following command sets the start frequency to the smallest value that is currently possible:

```
SENS:FREQ:START MIN
```

In addition, the max and min values can also be queried for these commands.

**For example**, the following command returns the smallest value that Start Frequency can currently be set to:

```
SENS:FREQ:START? MIN
```

An error will be returned if a numeric parameter is sent that exceeds the MAX and MIN values.

**For example**, the following command will return an "Out of range" error message.

```
SENS:FREQ:START 1khz
```

---

Last Modified:

10-Jul-2007    Removed image

## Configure for GPIB, SCPI, and SIDL

The following settings are used to configure the PNA for remote control using SCPI commands.

### How to Configure for SIDL / GPIB Operation

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Configure]**
3. then **[SIDL/GPIB]**

#### PNA Menu using a mouse

1. Click **Utility**
2. then **System**
3. then **Configure**
4. then **SIDL/GPIB/SCPI**

### Programming Commands



### SIDL / GPIB dialog box help

#### GPIB

**Talker/Listener Address** Sets the PNA address used to send and receive GPIB/SCPI commands to the system controller (external computer).

Use the National Instruments interface or the ACE (Agilent Connection Expert) interface to change the

System Controller address. Use the PNA as the system controller of external devices. Learn about the [PNA as controller](#).

See the rear panels of the [PNA-X](#) or [PNA 'C'](#) models.

## SICL

**SICL Enabled** When checked, the analyzer is capable of running GPIB programs on its computer to control analyzer functions. The programs must be run from a GPIB-capable programming environment (VEE, Visual Basic). This mode does not allow control of external GPIB instruments. To uncheck this box, exit the PNA application - (Click File, then Exit). The PNA restarts with the SICL enabled box unchecked unless **Automatically Enable on Startup** is checked.

Learn more about [Configuring for VISA and SICL](#).

**Address** Sets the PNA address.

**Automatically Enable on Startup** When checked, SICL Enabled is automatically selected when starting the PNA application.

## LAN Sockets/Telnet

Provides ability to communicate with the PNA from a PC that uses a Windows, or non-Windows, operating system.

- These settings are checked by default. If you have security concerns, clear these check boxes.
- These settings remain after the PNA is shutdown and restarted.

**Sockets Enabled** When checked, provides the ability to control the PNA from a remote SCPI program using port number 5025. [See the C# example that illustrates how this is done.](#)

**Telnet Enabled** When checked, provides the ability to send single SCPI commands from a remote Windows, or non-Windows, PC to the PNA using port number 5024.

How to send single SCPI commands using Telnet:

1. On the remote PC, click **Start**, then **Run**
2. Type: **telnet <computer name> 5024**  
where <computer name> is the full computer name of the PNA. [See how to find the computer name of the PNA.](#)
3. A Telnet window with a **SCPI>** prompt should appear on the remote PC screen.
4. From the SCPI prompt:
  - Type single SCPI commands
  - If an invalid SCPI command is sent, the prompt will disappear. Press **Enter** or **Ctrl C** to recover the SCPI prompt.
  - To exit the telnet window click **X** in the upper-right corner.
  - To get a normal telnet prompt, press **Ctrl ]** (closing bracket).

- To close the normal telnet window, type **Quit** and press **Enter**.

### SCPI Monitor / Input

**GPIB Command Processor Console**  Launches a window that is used to send single SCPI/GPIB commands from the PNA keyboard.

**Note:** Press **Control+Z** , then enter, to close the console window.

**Note:** The Status Register system can NOT be used from the GPIB Console.

- Type a valid command, with appropriate arguments and press enter.
- Use the arrow keys to recall previous commands.
- The console window may launch behind the PNA application. Press **Control+Tab** to bring the console window to the top.

**Monitor GPIB Bus** Enables monitoring activity on the GPIB.

**Show GPIB Bus Monitor Window** Shows and hides the window monitoring GPIB activity.

### Local and Remote Operation

The analyzer **LCL** and **RMT** (Local and Remote) operation labels appear in the lower right corner of the status bar.

**Note:** The status bar is NOT visible when the analyzer is preset. See [how to make the status bar visible](#).

- **LCL** appears when NOT under SCPI control
- **RMT** appears when under SCPI control. The RMT label does NOT appear when under COM control. Remote operation disables the front panel keys except for the **Macro/Local** key.

To return to Local (front panel) operation, press the Macro / Local key

Sending the GPIB "GTL" (go to local) command also returns the analyzer to Local operation.

Sending the GPIB "LLO" (local lockout) command disables the front panel Local button.

---

Last Modified:

1-May-2009    Added note about Status Register and Console

30-Oct-2008    Removed legacy content.

21-Feb-2008    Include Windows OS in Telnet/Sockets

## Getting Data from the Analyzer

Data is sent from the analyzer in response to program queries. Data can be short response messages, such as analyzer settings, or large blocks of measurement data. This topic discusses how to read query responses and measurement data from the analyzer in the most efficient manner.

### [Response Message Syntax](#)

### [Clearing the Output Queue](#)

### [Response Data Types](#)

### [Transferring Measurement Data](#)

**Note:** Some PCs use a modification of the IEEE floating point formats with the byte order reversed. To reverse the byte order for data transfer into a PC, use the [FORMat:BORDER](#) command.

## [Other Topics about GPIB Concepts](#)

### Response Message Syntax

Responses sent from the analyzer contain data, appropriate punctuation, and message terminators.

<NL><^END> is always sent as a response message terminator. Most programming languages handle these terminators transparent to the programmer.

Response messages use commas and semicolons as separators in the following situations:

- a comma separates response data items when a single query command returns multiple values

```
FORM:DATA? 'Query  
ASC, +0 'Analyzer Response
```

- a semicolon separates response data when multiple queries are sent within the same messages

```
SENS:FREQ:STAR?;STOP? --Example Query
```

```
+1.23000000000E+008; +7.89000000000E+008<NL><^END> 'Analyzer Response
```

### Clearing the Output Queue

After receiving a query, the analyzer places the response message in its output queue. Your program should read the response immediately after the query is sent. This ensures that the response is not cleared before it is read. The response is cleared when one of the following conditions occur:

- When the query is not properly terminated with an ASCII carriage return character or the GPIB <^END> message.
- When a second program query is sent.

- When a program message is sent that exceeds the length of the input queue
- When a response message generates more response data than fits in the output queue.
- When the analyzer is switched ON.

## Response Data Types

The analyzer sends different response data types depending on the parameter being queried. You need to know the type of data that will be returned so that you can declare the appropriate type of variable to accept the data. For more information on declaring variables see your programming language manual. The GPIB Command Finder lists every GPIB command and the return format of data in response to a query. The analyzer returns the following types of data:

### [Numeric Data](#)

### [Character Data](#)

### [String Data](#)

### [Block Data](#)

## Numeric Data

All numeric data sent over the GPIB is ASCII character data. Your programming environment may convert the character data to numeric data for you. Boolean data (1 | 0 ) is a type of numeric data.

## Character Data

Character data consists of ASCII characters grouped together in mnemonics that represent specific analyzer settings. The analyzer always returns the short form of the mnemonic in upper-case alpha characters. Character data looks like string data. Therefore, refer to the GPIB Command Finder to determine the return format for every command that can be queried.

### Example of Character Data

```
MLOG
```

## String Data

String data consists of ASCII characters. String parameters can contain virtually any set of ASCII characters. When sending string data to the analyzer, the string **must** begin with a single quote ( ' ) or a double quote ( " ) and end with the same character (called the delimiter).

**Note:** The analyzer responds best to all special characters if the string is enclosed in single quotes. If quotes are not used, the analyzer will convert the text to uppercase. The analyzer may not respond as you expect.

The analyzer always encloses data in double quotes when it returns string data.

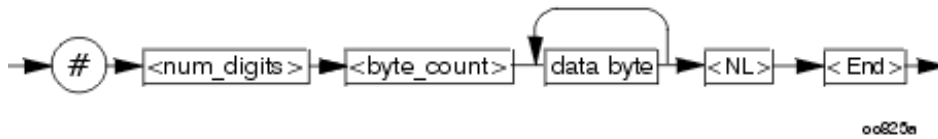
### Example of String Data

```
GPIB.Write "DISP:WINDow:TITLe:DATA?"
```

```
"This is string response data."
```

## Block Data

Block data is used to transfer measurement data. Although the analyzer will accept either definite length blocks or indefinite length blocks, it always returns definite length block data in response to queries unless the specified format is ASCII. The following graphic shows the syntax for definite block data:



<num\_digits> specifies how many digits are contained in <byte\_count>

<byte\_count> specifies how many data bytes will follow in <data bytes>

#### Example of Definite Block Data

#17ABC+XYZ<nl><end>

Where:

# - always sent before definite block data

1 - specifies that the byte count is one digit (7)

7 - specifies the number of data bytes that will follow, not counting <NL><END>

ABC+XYZ - Data

<NL><END> - always sent at the end of block data

#### Transferring Measurement Data

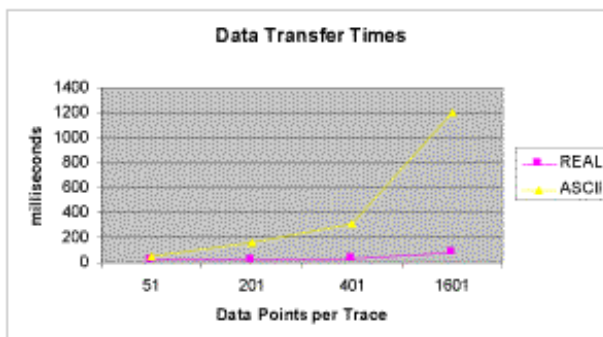
Measurement data is blocks of numbers that result from an analyzer measurement. Measurement data is available from various processing arrays within the analyzer. For more information on the analyzer's data processing flow, see [Accessing Data Map](#). Regardless of which measurement array is read, transferring measurement data is done the same.

[See an example.](#)

When transferring measurement data, the [FORMat:DATA](#) command allows you to choose from the following two data types:

- REAL
- ASCII

The following graphic shows the differences in transfer times between the two:





## REAL Data

REAL data (also called floating-point data) types transfer faster. This is because REAL data is binary and takes about half the space of ASCII data. The disadvantage of using REAL data is that it requires a header that must be read. See [definite length block data](#). The binary floating-point formats are defined in the IEEE 754-1985 standard. The following choices are available in REAL format:

- **REAL,32** - IEEE 32-bit format - single precision (not supported by HP BASIC)
- **REAL,64** - IEEE 64-bit format - double precision

## ASCII Data

The easiest and slowest way to transfer measurement data is to use ASCII data. ASCII data is sent if the data contains both numbers and characters (the setting of FORMat:DATA is ignored). ASCII data is separated by commas.

---

Last Modified:

28-Jul-2009    Added three zeros to example

26-Jul-2007    Added link to example

## Synchronizing the PNA and Controller

---

In this topic, Synchronizing the PNA and Controller means to keep PNA and the controller working at approximately the same pace.

### See Also

[Synchronize an External PSG Source](#)

[Triggering the PNA using SCPI](#)

### The Problem

The controller sends commands to the PNA as fast as the GPIB bus will allow. The PNA stores these commands in the PNA [Input queue](#). However, the PNA executes those commands at a slower rate than they are accepted. If left unchecked, the PNA input buffer will contain a long list of commands waiting to be executed.

At some point, the controller will send a query command which requires a response from the PNA. The controller will not send more commands until a response is received. It will wait for a response from the PNA for the amount of time set by the Timeout setting. If the PNA is working off a long list of commands in the input buffer, it may not execute and respond to the query command until the controller has quit waiting, or "timed out".

### The Solution

The easiest way to keep the controller and the PNA "synched" is to send query commands often. This stops the controller from sending more commands until the PNA executes and responds to the query. This limits the number of commands that are waiting in the PNA input queue to be processed.

Although any query will stop the controller from sending more commands, a good practice is to send [\\*OPC?](#) Most of the time, as soon as this query is executed, it will immediately reply. The exception to this is the Overlapped command.

- **Sequential** commands are executed quickly and in the order in which they are received.
- **Overlapped** commands take longer to execute. Therefore, they allow the PNA to execute other commands while waiting. However, the programmer may want to prevent the analyzer from processing new commands until the overlapped command has completed. If the PNA is executing an overlapped command when a \*OPC? is received, it will wait until the overlapped command is complete.

**Note:** The analyzer has two overlapped commands:

- [INITiate:IMMediate](#)
- [SENSe:SWEp:MODE GROUPS](#) (when INIT:CONT is ON)
- [Calibration Acquire commands using the ASYN argument](#)

### Analyzer Queues

Queues are memory buffers that store messages until they can be processed. The analyzer has the following queues:

## [Input Queue](#)

## [Output Queue](#)

## [Error Queue](#)

### **Input Queue**

The controller sends statements to the analyzer without regard to the amount of time required to execute the statements. The input queue is very large (31k bytes). It temporarily stores commands and queries from the controller until they are read by the analyzer's command parser. The input queue is cleared when the analyzer is switched ON.

### **Output Queue**

When the analyzer parses a query, the response is placed in the output queue until the controller reads it. Your program should immediately read the response or it may be cleared from the output queue. The following conditions will clear a query response:

- When a second query is sent before reading the response to the first. This does not apply when multiple queries are sent in the same statement.
- When a program statement is sent that exceeds the length of the input queue.
- When a response statement generates more data than fits in the output queue.
- When the analyzer is switched ON.

### **Error Queue**

Each time the analyzer detects an error, it places a message in the error queue. When the `SYSTEM:ERROR?` query is sent, one message is moved from the error queue to the output queue so it can be read by the controller. Error messages are delivered to the output queue in the order they were received. The error queue is cleared when any of the following conditions occur:

- When the analyzer is switched ON.
- When the `*CLS` command is sent to the analyzer.
- When all of the errors are read.

If the error queue overflows, the last error is replaced with a "Queue Overflow" error. The oldest errors remain in the queue and the most recent error is discarded.

### **Synchronization Methods**

The following common commands are used to synchronize the analyzer and controller. Examples are included that illustrate the use of each command in a program. See the SCPI command details to determine if a command is an overlapped command.

#### [\\*WAI](#)

#### [\\*OPC?](#)

#### [\\*OPC](#)

## \*WAI

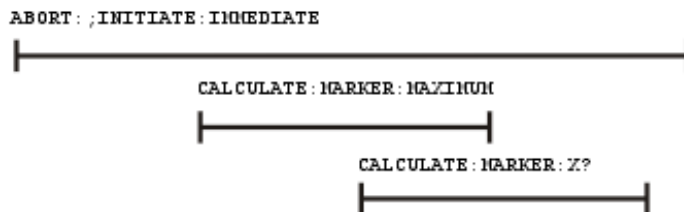
The \*WAI command:

- **Stops the analyzer** from processing subsequent commands until all overlapped commands are completed.
- **It does NOT stop the controller** from sending commands to this and other devices on the bus. This is the easiest method of synchronization.

Example of the \*WAI command

```
GPIB.Write "ABORT;;INITIATE:IMMEDIATE" 'Restart the measurement.  
GPIB.Write "CALCULATE:MARKER:SEARCH:MAXIMUM" 'Search for max amplitude.  
GPIB.Write "CALCULATE:MARKER:X?" 'Which frequency?
```

The following time line shows how the processing times of the three commands relate to each other:

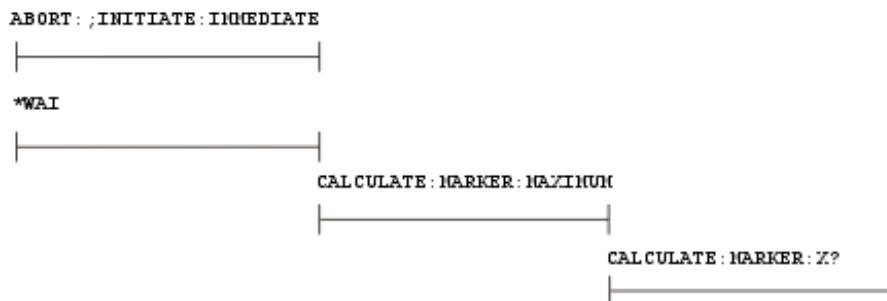


`INITIATE:IMMEDIATE` is an overlapped command; it allows the immediate processing of the sequential command, `CALCULATE:MARKER:SEARCH:MAXIMUM`. However, the `INITIATE:IMMEDIATE` is not considered complete until the measurement is complete. Therefore, the marker searches for maximum amplitude before the measurement completes. **The `CALCULATE:MARKER:X?` query could return an inaccurate value.**

To solve the problem, insert a \*WAI command.

```
GPIB.Write "ABORT;;INITIATE:IMMEDIATE" 'Restart the measurement.  
GPIB.Write "*WAI" 'Wait until complete.  
GPIB.Write "CALCULATE:MARKER:MAXIMUM" 'Search for max amplitude.  
GPIB.Write "CALCULATE:MARKER:X?" 'Which frequency
```

The time line now looks like this:



The \*WAI command keeps the `MARKER:SEARCH:MAXIMUM` from taking place until the measurement is completed. The `CALCULATE:MARKER:X?` query returns the correct value.

**Note:** Although \*WAI stops the analyzer from processing subsequent commands, it does not stop the controller. The controller could send commands to other devices on the bus.

## \*OPC?

The \*OPC? query stops the controller until all pending commands are completed.

In the following example, the **Read** statement following the \*OPC? query will not complete until the analyzer responds, which will not happen until all pending commands have finished. Therefore, the analyzer and other devices receive no subsequent commands. A "1" is placed in the analyzer output queue when the analyzer completes processing an overlapped command. The "1" in the output queue satisfies the **Read** command and the program continues.

### Example of the \*OPC? query

This program determines which frequency contains the maximum amplitude.

```
GPIB.Write "ABORT; :INITIATE:IMMEDIATE"! Restart the measurement
GPIB.Write "*OPC?" 'Wait until complete
Meas_done = GPIB.Read 'Read output queue, throw away result
GPIB.Write "CALCULATE:MARKER:MAX" 'Search for max amplitude
GPIB.Write "CALCULATE:MARKER:X?" 'Which frequency?
Marker_x = GPIB.Read
PRINT "MARKER at " & Marker_x & " Hz"
```

### \*OPC

The \*OPC command allows the analyzer and the controller to process commands while processing the overlapped command.

When the analyzer completes processing an overlapped command, the \*OPC command sets bit 0 of the standard event register to 1. This requires polling of status bytes or use of the service request (SRQ) capabilities of your controller. See [Reading the Analyzer's Status Registers](#) for more information about the standard event status register, generating SRQs, and handling interrupts.

**Note:** Be careful when sending commands to the analyzer between the time you send \*OPC and the time you receive the interrupt. Some commands could jeopardize the integrity of your measurement. It also could affect how the instrument responds to the previously sent \*OPC.

[Example](#) of polled bit and SRQ processes.

## When To Synchronize the Analyzer and Controller

The need to synchronize depends upon the situation in which the overlapped command is executed. The following section describes situations when synchronization is required to ensure a successful operation.

### [Completion of a Measurement](#)

#### [Averaged Measurements](#)

### Completion of a Measurement

To synchronize the analyzer and controller to the completion of a measurement, use the `ABORT; INITIATE:IMMEDIATE` command sequence to initiate the measurement.

This command sequence forces data collection to start (or restart) under the current measurement configuration. A restart sequence, such as `ABORT; INITIATE:IMMEDIATE` is an overlapped command. It is complete when all operations initiated by that restart command sequence, including the measurement, are finished. The `*WAI`, `*OPC?` and `*OPC` commands allow you to determine when a measurement is complete. This ensures that valid measurement data is available for further processing.

## Measurements with External Trigger

See [Triggering the PNA using SCPI](#).

[External Triggering](#)

## Averaged Measurements

Averaged measurements are complete when the average count is reached. The average count is reached when the specified number of individual measurements is combined into one averaged measurement result. Use synchronization to determine when the average count has been reached.

If the analyzer continues to measure and average the results after the average count is reached, use synchronization to determine when each subsequent measurement is complete.

---

Last Modified:

6-Apr-2009    Added links to External trig and triggering

17-Feb-2009    Added Cal ACQ to Overlapped commands

## Calibrating the PNA Using SCPI

---

There are several ways to calibrate the PNA using SCPI depending on your measurement needs. As from the Cal Wizard, you can perform a Guided Cal, Unguided Cal, or ECal. This topic explains the differences in these calibration choices when using SCPI commands.

### [Guided Calibrations](#)

#### [ECal](#)

#### [Perform Computer Operations During Acquire](#)

#### [Creating Cal Sets](#)

#### [Applying Cal Sets and Cal Types](#)

#### [Uploading Error Terms](#)

#### [Unguided Cals and Calibration Classes](#)

### [See SCPI Calibration Examples](#)

## Guided Calibrations

- Use [Sens:Corr:Coll:Guided](#) commands.
- These commands calibrate the ACTIVE channel. Activate a channel by selecting a measurement on the channel to be calibrated using [Calc:Par:Select](#).
- Full 1,2,3,4-port SOLT and TRL calibrations - No response cals.
- All of the advanced calibration features (Thru method, specify DUT connectors and Cal kits for each port, port pairings).
- A Cal Set is applied to the channel and saved at the completion of a guided cal according to the preference setting [SENS:CORR:PREF:CSET:SAVE](#)

## ECal

From the Cal Wizard or from a SCPI program, ECal is fast, accurate, and very repeatable. Unlike from the Cal Wizard, you can use SCPI to perform ECal using either the Guided or Unguided commands. The Unguided commands are easiest to use. However, the following situations require that you use the Guided commands.

- To maximize accuracy, all ECal calibrations on the PNA perform an Unknown Thru measurement of the ECal module Thru state **IF** the PNA model being used has [1 reference receiver per port](#). If your PNA does NOT have 1 reference receiver per port, use Guided ECal commands and specify a Thru method.
- If your ECal module connectors do NOT match the DUT connectors, and you choose not to perform a User

Characterization, use Guided ECal commands and specify the Thru method.

#### ECAL Notes:

- When using either Guided or Unguided ECal commands under low power situations, use the Orientation settings. The Guided example shows the use of these commands. When using Unguided, they must appear before the Acquire command.
- The frequency range of the measurement must be within the range of the ECal module. Otherwise, the calibration will fail.
- Although we recently provided the command, you do NOT have to send the ECal module state command. The ECal algorithm switches ECal states automatically.
- All of these ECal choices are listed in the [Programming Command Search](#) function in this Help file.

See [Using ECal](#) to learn about all of the ECal features.

### Perform Computer Operations During Acquire

During a calibration with slow sweep speeds, such as when using a narrow IF bandwidth, you may want to have your program perform other operations, such as checking for the click event of a Cancel button.

To do this, use the optional ASYNchronous argument with the ACQUIRE command as shown in several calibration example programs. The PNA parser returns immediately while the cal step measurement proceeds. It does NOT block commands and wait for the measurement step to finish. You can send \*ESR? or \*STB? queries to monitor the status register bytes to see when the OPC (operation complete) bit gets set, which indicates the cal measurement step has finished. Learn more about [status registers](#).

If your program is using the [ScpiStringParser](#), then you can ONLY use \*OPC? to detect when the OPC bit is set.

When using the ASYN argument, set the timeout value in the IO settings to at least 5 seconds. There are intervals during the cal acquires when the PNA takes a several seconds to respond to additional commands, such as when the processor is calculating error terms.

The following Acquire commands have this argument:

Command	Example
<a href="#">SENS:CORR:COLL:GUID:ACQUIRE</a> (Guided Cal)	<a href="#">Guided 2-Port or 4-Port Cal</a>
<a href="#">SENS:CORR:COLL:ACQUIRE</a> (Unguided Cal)	<a href="#">Perform Unguided ECal</a>
<a href="#">SENS:CORR:COLL:SESS:ACQUIRE</a> (FCA)	
<a href="#">SOUR:POW:CORR:COLL:ACQUIRE</a> (Source Power Cal)	<a href="#">Perform a Source and Receiver Power Cal</a> (shows polling loop)

Learn more about [Synchronizing the PNA and Controller](#).

### Creating Cal Sets



There are several ways to store guided cal data into a unique Cal Set. The following is probably the easiest. It does not require the name of an existing Cal Set and it allows you to name the Cal Set.

```
SENS:CORR:COLL:GUID:INIT 'start the cal with no cal set argument  
'Perform the cal  
SENS:CORR:COLL:GUID:SAVE 'create cal set with auto-generated name or to cal  
register  
SENS:CORR:CSET:NAME 'MyCalSet' 'name the current cal set.'
```

## Applying Cal Sets and Cal Types

A Cal Set is applied to the channel and saved at the completion of a guided cal according to the preference setting [SENS:CORR:PREF:CSET:SAVE](#).

When you select a Cal Set to apply to an uncalibrated channel, the PNA attempts to find the most comprehensive calibration type in the Cal Set and turn it ON. In addition, changing a measurement parameter (for example, from S11 to S21) will also initiate an attempt to apply the best Cal Type and turn correction ON.

There may be times when you do not want the most comprehensive Cal Type. For example, say there is a Full 2-port Cal Set applied, but there is only an S11 measurement displayed. If measurement speed is a concern, you can apply a Full 1-Port Cal Type from that same Cal Set and save time by not doing the extra background sweeps.

[Learn more about background sweeps.](#)

If you change the measurement parameter, the PNA will reapply the Full 2-Port Cal Type.

See the SCPI and COM commands for [Cal Sets](#) and [Cal Types](#).

## Uploading Error Terms

There are two ways to upload error terms using SCPI: the old way and the recommended way. The old way will still work but requires a 'preference' setting.

**The old way is this:**

```
SENS:CORR:COLL:METHod <cal type>  
SENS:CORR:COLL:SAVE  
CALC:DATA SCORR[n] < data to upload >
```

This technique, used in WinCal software, starts a calibration and immediately saves it without acquiring any standards. In PNA Rev 6.0, executing SAVE without acquiring data will return an error. To suppress the error and continue to use the above technique to upload error terms, send the following command to set the preference:

[SENS:CORR:PREF:SIMCal 1](#)

Or you can execute the script that is saved on the PNA at C:/Program Files/Agilent/Network Analyzer/System/wincal32.reg.

Setting this preference defeats some error checking when performing unguided cal using SCPI. This is not recommended unless needed for backward compatibility.

**The recommended way** is to upload error terms into a created or selected Cal Set:

```
SENS:CORR:CSET:CREate or SENS:CORR:CSET:GUID  
SENS:CORR:CSET:Data <term> <port> <port> <data>  
SENS:CORR:CSET:SAVE
```

This method puts error terms into a Cal Set, outside of a Guided or Unguided calibration session. The Cal Set can then be applied at any time.

See [SENS:CORR:CSET](#) commands.

## Unguided Cals and Calibration Classes

- Use [Sens:Correction](#) commands.
- 1-port, 2-port, Response.
- Can select 2 sets of standards.
- TRL is NOT recommended.

The following describes how to perform an unguided calibration using SCPI. The objective here is to make clear the relationship between the physical port on which a standard is being measured, the actual device in the cal kit, and the SCPI command used to acquire the device.

Calibration standards classes are 'categories' of standard types. To perform a 2 port calibration, the cal wizard requires the following types of standards to be measured:

### 3 reflection standards on the forward port:

- Class S11A typically an open
- Class S11B typically a short
- Class S11C typically a load

### Likewise, 3 reflection standards are required for the reverse port:

- Class S22A typically an open
- Class S22B typically a short
- Class S22C typically a load

### There is also a transmission standard that is measured in both directions:

- Class S21T typically a thru

The following illustrates the relationship between cal kit physical standards and calibration classes. Here is a list of the physical devices in my calibration kit.

Standard #1 = "3.5 mm male short"

Standard #2 = "3.5 mm male open"

Standard #3 = "3.5 mm male broadband load"

Standard #4 = "Insertable thru standard"

Standard #5 = "3.5 mm male sliding load"

Standard #6 = "3.5 mm male lowband load"

Standard #7 = "3.5 mm female short"

Standard #8 = "female to female characterized thru adapter"

Standard #9 = "0-2 Load"

Standard #10 = "Open"

Standard #11 = "Non-insertable thru"

Standard #12 = "3.5 mm female lowband load"

Standard #13 = "3.5 mm female sliding load"

Standard #14 = "3.5 mm female broadband load"

Standard #15 = "3.5 mm female open"

When you perform a calibration remotely using SCPI, you don't specify the device number directly. Rather, you specify the class you want to measure. Each device in the calibration kit is assigned to a class. And since more than one device can be assigned to the same class, each class contains an ordered list of devices. The class assignments are set using the Advanced Modify Cal Kit dialog or the SCPI command:

[SENS:CORR:COLL:CKIT:ORDer](#)<class>, <std>, <std>, <std>, <std>,<std>,<std>,<std>

The 85052B kit used in the example program has the following standard list for each class: The list was obtained by issuing the corresponding SCPI query:

[SENS:CORR:COLL:CKIT:OLIST1?](#) S11A = +2,+15,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST2? S11B = +1,+7,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST3? S11C = +6,+5,+3,+12,+13,+14,+0

SENS:CORR:COLL:CKIT:OLIST4? S21T = +4,+8,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST5? S22A = +2,+15,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST6? S22B = +1,+7,+0,+0,+0,+0,+0

SENS:CORR:COLL:CKIT:OLIST7? S22C = +6,+5,+3,+12,+13,+14,+0

SENS:CORR:COLL:CKIT:OLIST8? S12T = +4,+8,+0,+0,+0,+0,+0

When you perform the calibration, you acquire data by issuing the ACQUIRE command:

[SENS:CORR:COLL:ACQ](#) <class>[, <subst> ]

For example:

[SENS:CORR:COLL:SFOR](#) 1

SENS:CORR:COLL:ACQ STANA, SST2

The SFOR command tells the wizard to make the next acquisition in the forward direction. The ACQUIRE command specifies that we are measuring the 2nd device in the list for STANA. And since we are measuring SFORward, then STANA refers to class #1 or S11A. The list of devices for this class are specified in the OLIST1 query above.

Alternately, you could modify the device order for the S11A class to move device #15 into the first position (SENS:CORR:COLL:CKIT:ORDER1). When the desired device is in the first position, you need not specify the order number in the ACQUIRE command. The default is the first device in the OLIST. This works well for two port network analyzers where the order for S11A,B,C classes is set up for port 1 and the order for S22A,B,C is set up for port 2. With the kit set up in the proper order, you eliminate the need to specify the substandard number (SST<n>).

[See an example: Perform an Unguided 2-port Cal on a \*\*4-port PNA\*\*.](#)

---

Last Modified:

17-Apr-2009    Remove \*OPC? from Sync note

13-Feb-2009    Added Asynch section

## The PNA as a USB Device

---

Beginning with PNA Rev. A.09.00, the PNA can be controlled as a USB Device using SCPI. This is done through the Agilent I/O Libraries which must be installed on your remote computer.

All data types, especially [Binary block data](#), transfer MUCH faster using USB as compared to GPIB.

### To communicate with the PNA as a USB device

1. Connect the PNA to the remote computer using the rear-panel device-side USB connector. This connector is available ONLY on PNA models with a [1.1 GHz CPU board](#), and the [PNA-X](#).
2. The 'Found New Hardware' wizard is launched. Follow the prompts to install the PNA driver software.
3. The Agilent I/O Libraries will recognize the PNA as a Test and Measurement device and show the following dialog.

**Note:** The PNA is not a USB Mass Storage Device. Therefore, Windows Explorer does NOT recognize it as a USB device. You can NOT use Windows Explorer to transfer files to and from the PNA. For file transfer, use the SCPI command [MMEM:TRANsfer](#).



**Alias name** Change this to a name that is easy to recognize. Once configured, use the Alias name to communicate with the USB device using applications such as VISA and SICL:

- VISA: `viOpen (...,"UsbDevice1",...)`
- SICL: `iopen ("UsbDevice1")`

For more information, see the Connectivity Guide in the Agilent I/O libraries.

---

Last Modified:

26-Jul-2007    MX New topic

## Reading the Analyzer's Status Register

The PNA has several status registers that your program can read to know when specific events occur. There are two methods of reading the status registers in the analyzer: the Polled Bit method and the Service Request method.

- [The Status Registers](#)
- [Setting and Reading Bits in Status Registers](#)
- [Polled Bit Method](#)
- [Service Request Method](#)

### See Also

[IEE 482 Common commands](#)

[Example: Status Reporting](#)

[Status Commands](#)

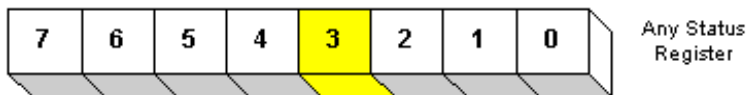
### Other Topics about GPIB Concepts

### Important Notes:

- A new [Limit Line Fail command](#) that makes it easy to determine if Limit Line testing has failed.
- [\\*OPC?](#) can be used to easily determine when a channel has completed a sweep. This requires no interaction with the Status Register system. Most [PNA programming examples](#) use [\\*OPC?](#).
- Most of the Status Register system can NOT be used with the [SCPIStringParser Object](#). However, [\\*OPC?](#) can be used.

### The Status Registers

Most of the status registers in the analyzer have sixteen bits. For simplicity, this topic will illustrate their use with 8-bit registers. Bits in registers represent the status of different conditions inside of the analyzer. In the following graphic, a register is represented by a row of boxes; each box represents a bit. Bit 3 is ON.



Each PNA Status Register is actually comprised of the following registers. [See an image of the PNA Status registers.](#)

- **Enable Registers** - When using the [SRQ method of polling](#), you first set bits in the enable register which tells the PNA which events to monitor. This is not necessary using the [Polled Bit method](#), as you can only monitor a single event. A [\\*CLS](#) (clear status) command will not clear the enable register. The [\\*ESE](#) and

\*ESE? commands are used to set and query Enable bits, while \*ESR is used to read and clear an Enable register. [Learn how to set bits.](#)

- **Condition Registers** - A condition register continuously monitors events in the PNA. Bits in the condition register change real time as conditions occur. These bits are not latched, so this register is used mainly for diagnostic purposes. The registers that only summarize lower level registers do NOT have a condition register.
- **Event Registers** - This is the register that is read to determine if an event has occurred. An event register latches the bits from the corresponding condition register. When an event register bit is set, subsequent changes to the corresponding condition register bit are ignored. The bit remains set until a query command such as \*CLS clears the bit. [Learn how to read the Event Register.](#)
- **Positive and Negative Transition Registers** - Transition registers control what type of in a condition register will set the corresponding bit in the event register.
  - **Positive** transitions (**0 to 1**) are only reported to the event register if the corresponding positive transition bit is set to 1.
  - **Negative** transitions (**1 to 0**) are only reported to the event register if the corresponding negative transition bit is set to 1.
  - Setting **both** transition bits to 1 causes both **positive and negative** transitions to be reported.

Transition registers are read-write and are unaffected by \*CLS (clear status) or queries. They are reset to their default settings at power-up and after \*RST and SYSTem:PRESet commands. The **following are the default settings** for the transition registers:

- All Positive Transition registers = 1
- All Negative Transition registers = 0

This means that, by default, the analyzer will latch all event registers on the negative to positive transition (0 to 1).

The following is an example of why you would set transition registers:

A critical measurement requires that you average 10 measurements and then restart averaging. You decide to poll the averaging bit. When averaging is complete, the bit makes a positive transition. After restart, you poll the bit to ensure that it is set back from 1 to 0, a negative transition. You set the negative transition bit for the averaging register.

### Setting and Reading Bits in Status Registers

Both the Polled-Bit method and Service Request method require that you set and read status register bits. Most of the PNA status registers contain 16 bits, numbered 0 to 15. Each bit has a weighted value. The following example shows how to set the bits in a 8-bit status register.

8-bit register

Bit	0	1	2	3	4	5	6	7
Weight	1	2	4	8	16	32	64	128

How to set bits 4 and 5 in the Standard Event Status Enable register:



Step	Example
1. Determine the weighted bit value for these bits	weights 16 and 32 (respectively)
2. Add these values together	$16 + 32 = 48$
3. Send this number as an argument in the appropriate command. (see <a href="#">Status Commands</a> )	STAT:QUES:LIMIT1:ENAB 48

---

### The Polled Bit Method

With the Polled Bit Method, your program monitors a bit in the status register that represents the condition of interest to you. When the PNA sets the bit to 1, your program sees it and responds accordingly.

- If your program **periodically** monitors a bit in the status register, it is free to do other things as well. However, your program can respond only as fast as the bit is polled.
- If your program **continually** monitors a bit, it can respond immediately, but will be unavailable to do anything other than poll the bit.

**Advantage:** This method requires very little programming.

#### Procedure:

1. Decide which condition to monitor. The [Status Commands](#) topic lists all of the possible conditions that can be monitored in the analyzer.
  2. Determine the command to be used to monitor the bit.
  3. Determine how often to poll the bit until it is set.
  4. Construct the routine to respond when the bit is set.
- 

### The Service Request (SRQ) Method

Your program enables the bits in the status registers representing the condition of interest. When the condition occurs, the PNA actively interrupts your program from whatever it is doing, and an event handler in your program responds accordingly. Do this method if you have several conditions you want to monitor or the conditions are such that it is not practical to wait for the condition to occur.

**Advantage:** This method frees your program to do other things until the condition occurs. The program is interrupted to respond to the condition.

**Disadvantage:** This method can require extensive programming depending on the number and type of conditions that you want to monitor.

#### Procedure:

1. Decide which conditions to monitor. The [Status Commands](#) topic lists all of the possible analyzer conditions that can be monitored.
2. Set the enable bits in the **summary** registers and the **status byte** register.

**Enabling** is like making power available to a light. Without power available, the switch can be activated, but the light won't turn ON. In the analyzer, without first enabling a bit, the condition may occur, but the controller won't see it unless it is enabled.

The condition, and the bit in the **summary** registers in the reporting path, must be enabled. This is like streams (conditions) flowing into rivers (summary registers), and rivers flowing into the ocean (controller). See the diagram of status registers in [Status Commands](#).

Bit 6 of the **status byte** register is the only bit that can interrupt the controller. When **any** representative bit in the status byte register goes ON, bit 6 is automatically switched ON.

3. Enable your program to interrupt the controller. This is done several ways depending on the programming language and GPIB interface card you use. An [example program](#) is provided showing how this is done with in Visual Basic with a National Instruments GPIB card.

4. Construct a subroutine to handle the interrupt event. If you are monitoring more than one condition in your system, your event handler must determine which condition caused the interrupt. Use the \*SPE command to determine the instrument that caused the interrupt, then poll the summary registers, then poll condition registers to determine the cause of the interrupt.

---

Last Modified:

4-Jun-2009    Modified Polled Bit method

1-May-2009    Several edits

## Configure for SCPI LAN using SICL / VISA

---

### [PNA Supported Interfaces](#)

### [Agilent I/O Libraries](#)

### [SICL / VISA Programs Running on the PNA](#)

### [Configure the PNA for SICL / VISA](#)

### [Configure the External Controller](#)

### [Other Topics about GPIB Concepts](#)

## PNA Supported Interfaces

The PNA supports the following interfaces for SICL / VISA communication:

- **LAN** - as a remote GPIB interface. The PNA LAN is presented as a virtual GPIB interface. It does NOT support simple TCPIP-based control. Therefore, when configuring the Agilent IO libraries on your PC, add a **REMOTE GPIB** interface, which uses the LAN client interface.
- **GPIB** - requires that your external controller have a GPIB card.

**Note:** For optimum LAN interface performance, use [COM](#) to control the PNA. SCPI commands can be sent to the PNA using the COM [SCPIStringParser](#) object.

The following interfaces are NOT supported:

- **USB**
- **Serial**

### Important Note:

To enable VISA or SICL communication over LAN, you must do the following:

1. On the PNA, click **System**, point to **Configure**, then click **SICL/GPIB**.
2. Check **SICL Enabled**. To automatically enable SICL when the PNA is booted, check **Automatically enable on Startup**.
3. Click **OK**.

The PNA is now ready to be controlled over LAN.

[Learn more about this dialog box.](#)

### Agilent I/O Libraries

The Agilent I/O libraries includes the drivers to allow you to communicate with Agilent test instruments. Every PNA is shipped with the Agilent I/O libraries installed. We recommend you do NOT upgrade the Agilent I/O libraries on the PNA as unexpected results may occur. If you choose to upgrade the Agilent I/O libraries on the PNA, do NOT change the default folder path in the InstallShield Wizard.

To communicate with the PNA, the Agilent I/O libraries must also be installed on your external controller. To purchase the Agilent I/O libraries, or download a free upgrade, go to [www.agilent.com](http://www.agilent.com) and search for IO Libraries. Scroll to find Software, Firmware & Drivers.

### SICL / VISA Programs Running on the PNA

You can run your SICL / VISA program on the PNA to control the PNA. Although the Agilent I/O libraries are already installed on the PNA, it is configured as the **Host**. You must also configure a SICL or VISA LAN **Client** interface on the PNA, specifying the LAN hostname of that same PNA.

If your program uses the COM interface to VISA, and is compiled on a PC with the Agilent IO Libraries Suite (version 14 or later), and the resulting executable is copied and run on the PNA, it will produce a "type mismatch error". This is because the PNA has the 'M' version of Agilent I/O libraries. The following Visual Basic code is an example of how to avoid this error when communicating with the PNA from within the PNA:

```
Dim rm As IResourceManager
Dim fmio As IFormattedIO488
Set rm = CreateObject("AgilentRM.SRMCLs")
Set fmio = CreateObject("VISA.BasicFormattedIO")
Set fmio.IO = rm.Open("GPIB0::22")
fmio.WriteString "*IDN?" & Chr(10)
MsgBox fmio.ReadString()
```

### Controlling the PNA over LAN while controlling other instruments over GPIB

The PNA can NOT be both a controller and talker/listener on the same GPIB bus. Using SICL / VISA, you can use LAN to control the PNA, leaving the PNA free to use the rear-panel GPIB interface to control other GPIB devices.

## Configure the PNA for SICL / VISA

1. On the PNA, click **System** then check **Windows Taskbar**
2. Click **Start** then point to **Program Files, Agilent IO Libraries**, then click **IO Config**
3. Select each GPIB Interface and click **Edit** to verify (or make) the default settings in the following table. These settings are REQUIRED when using a [82357A USB / GPIB](#) Interface with the PNA.
4. When complete, click **OK** to close the edit dialog.
5. Click **OK** to close the IO Config dialog.

VISA Interface Name	SICL Interface Name	Dialog box title	Description
GPIB0	gpib0	GPIB Using NI-488.2	PNA Rear-panel GPIB connector. This GPIB interface can be used to control the PNA <b>OR</b> for the PNA to control external equipment. IT CAN NOT DO BOTH IN THE SAME PROGRAM. <a href="#">Learn more about pass-through options.</a>
GPIB1	hpib7	Internal Instrument Configuration	Internal interface for programs running on the PNA to control itself.
GPIB4	inst0	Internal Instrument Configuration	Used for <a href="#">LXI compliance</a> . <b>Do NOT delete this interface.</b>

## Configure the External Controller

Please refer to the Agilent I/O libraries documentation to learn how to configure your controller to communicate with the PNA. These links can show you how to find the following PNA information:

- [PNA full computer name](#)
- [GPIB Address](#)
- [IP Address](#)

This [example program](#) can help test your VISA configuration.

---

Last Modified:

13-Aug-2008    Added GPIB4

## Agilent VEE Pro RunTime Installed

---

Beginning in Dec. 2005, Agilent VEE Pro RunTime is installed on new PNAs. This means that programs written with Agilent VEE (.vxe files) can be run directly on the PNA. New PNA models with [1.1 GHz CPU](#) have VEE 7.5 runtime with Agilent I/O Libraries suite 14.

PNAs **without** Agilent VEE installed can go to the [Agilent VEE website](#) and download Agilent **VEE Pro 6.2** RunTime to the PNA and begin to run VEE programs directly on the PNA. This version does not require Agilent I/O Libraries suite 14. [Do NOT upgrade to Agilent I/O libraries suite 14 on the PNA.](#)

With Agilent VEE Pro RunTime installed on the PNA, the following examples can be run directly on the PNA:

- [Basic Control](#) of the PNA

For more VEE examples, see the [PNA support website](#).

For more information on Agilent VEE, see [www.agilent.com/find/VEE](http://www.agilent.com/find/VEE)

## Basic Control using VEE

---

This VEE Pro 6.0 example does the following:

- Controls PNA windows and traces.
- Changes stimulus settings.
- Measures all four S parameters.
- Create markers and displays marker readout.

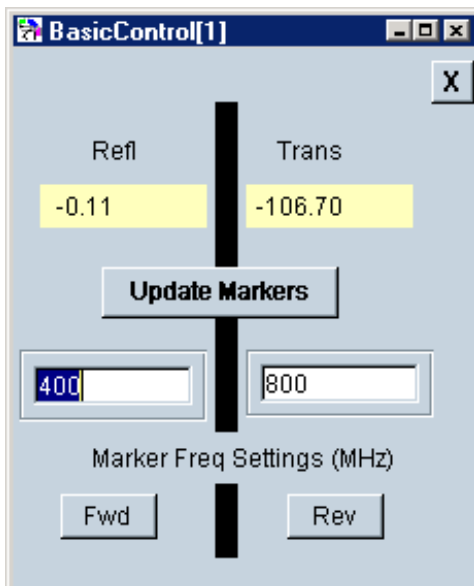
If this Help file is on a PNA and [VEE Pro RunTime is installed](#), then:

1. [Run the BasicControl.vxe example](#)
2. Then click **Open** on the following dialog box to run the program.

Otherwise, you can modify the example program using VEE, [save the VEE BasicControl.vee](#)  
[Learn how to run this program as a Macro on the PNA.](#)

---

The following dialog box will be visible on the PNA when the example program is running.



- Click **Fwd** to activate the Forward (S11 and S21) measurements.
- Click **Rev** to activate the Reverse (S22 and S12) measurements.
- Click **Update Markers** to sweep the PNA.
- Type values to change Marker Frequencies.





## ECal with Confidence Check using VEE

---

This VEE Pro 6.0 example performs an ECal and subsequent ECal confidence Check.

If this Help file is on a PNA and [VEE Pro RunTime is installed](#):

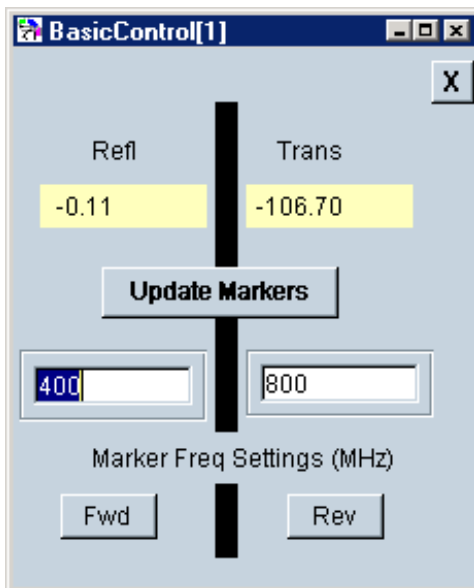
- [Run the .vxe example](#)
- Then click **Open** on the following dialog box to run the program.

Or to modify the example program using VEE, [save the VEE BasicControl.vee](#)

[Learn how to run this program as a Macro on the PNA.](#)

---

The following dialog box will be visible on the PNA when the example program is running.



- Click **Fwd** to activate the Forward (S11 and S21) measurements.
- Click **Rev** to activate the Reverse (S22 and S12) measurements.
- Click **Update Markers** to sweep the PNA.
- Type values to change Marker Frequencies.



## Interface Control

---

The Interface Control feature allows you to send remote commands and data to the following PNA rear-panel Interfaces: GPIB, Material Handler I/O, Test Set I/O, and Auxiliary I/O.

- [Overview](#)
- [How to Access Interface Control Settings](#)
- [Interface Control Dialog Box](#)
- [Z5623A H08 Test Set Commands](#)

### Other System Configuration Topics

## Overview

The Interface Control feature allows you to send data to control external equipment such as GPIB instruments, a material handler, test set, or other equipment, without needing to create a remote program. The PNA manages the timing and required interface setup. See [Rear Panel Tour](#).

- A unique set of control data can be sent for each channel. In addition, a unique set of control data can be sent before the channel sweep starts, and after the sweep ends.
- Interface Control settings can be saved and recalled from the [Interface Control dialog box](#), or with [Instrument State Save and Recall](#).
- Interface Control settings can be copied to other channels using [Copy Channels](#).
- Control data can only be WRITTEN to the interfaces, NOT READ from the interfaces.
- Control data is sent in the following order. This order cannot be changed.

1. [GPIB Interface](#)
2. [Material Handler Interface](#)
3. [Test Set Interface](#)
4. [Aux Interface](#)
5. [Dwell Time](#)

## How to access Interface Control settings

### Using front-panel HARDKEY [softkey] buttons

1. Press TRACE/CHAN
2. then [Channel]
3. then [More]
4. then [Interface Control]

### Using a mouse with PNA Menus

1. Click **Trace/Chan**
2. then **Channel**
3. then **More**
4. then **Interface Control**

## Programming Commands

**Interface Control**

☒ Enable Interface Control

Channel **1** Channel Control Label:

Before Sweep Start | After Sweep End

GPIB Commands - BEFORE ☒ Enable Control

Handler I/O Control ☒ Enable Control

Port A  (0-255)

Port B  (0-255)

Port C  (0-15)

Port D  (0-15)

Dwell After Command  ms

Test Set I/O Control (addr.data) ☒ Enable Control

Aux I/O Output Voltage ☒ Enable Control

DAC1  (-10V -> +10V)

DAC2  (-10V -> +10V)

Reset All Save Recall **OK** Cancel Help

### Interface Control dialog box help

[See Interface Control Overview \(scroll up\)](#)

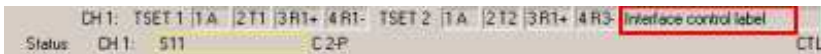
An [Instrument Preset](#) will reset all of the fields to their default settings.

**Note:** If an error is encountered when sending Interface Control data, an error message is displayed on the PNA screen. The [Channel Trigger State](#) is set to Hold. You must fix the condition that caused the error, then change the Channel Trigger State to its original setting.

**Enable Interface Control** Enables and disables ALL Interface Control communication. When cleared (default setting) Interface Control is disabled and NO data is sent. To send data, the individual interfaces must also be enabled.

**Channel** Specifies the channel number for dialog settings. Each channel is configured individually. The list box shows the channels that currently have measurements. There must be at least one measurement present in order to make settings.

**Channel Label** Specifies the label to be displayed on the second status bar at the bottom of the PNA screen. This field is shared with [External Testset control](#). The second status bar is automatically displayed when Interface Control is enabled.



[Learn about the primary status bar.](#)

### Before Sweep Start - After Sweep End Tabs

Commands / data for all four interfaces can be sent both Before Sweep Start and After Sweep End. However, they are configured and enabled on separate tabs of the Interface Control dialog box. For example, to send GPIB commands both Before and After a PNA sweep, the Enable Control checkbox must be selected and commands entered on BOTH the Before Sweep Start and After Sweep End tabs.

**Before Sweep Start** The data is sent BEFORE the first trace on the channel begins sweeping.

**After Sweep End** The data is sent AFTER the last trace on the channel completes sweeping.

## GPIB Commands

### Notes:

- While using the rear-panel GPIB port with Interface Control, the PNA must be in GPIB [System Controller mode](#). If the PNA is NOT in System Controller mode, an error message appears AND Interface Control is disabled. To correct this situation,
  1. Put the PNA in System Controller mode, and
  2. Re-enable Interface Control.
- GPIB instruments CAN be connected to the PNA using a [USB/GPIB adapter](#). In this case, the PNA can be in talker-listener mode.
- GPIB Queries are NOT supported.

**Enable Control** Enables and disables sending commands out the GPIB interface.

**Multi-line edit control** Each line contains a GPIB command using the following syntax:

```
address    command
```

Where:

**address** a number between 0 and 31. The PNA will look through all of the GPIB interfaces for an instrument connected to the specified address. If an instrument with that address is not recognized, an error is returned.

**command** a GPIB command, with or without enclosing quotes. Enclosing quotes are ignored.

Address and command are separated by at least one space.

Commands should be separated by a new line, or carriage return. For example:

```
19 ":init:cont off"  
16 init:imm
```

The front-panel **Enter** key inserts a new line into the field.

The number of GPIB commands that can be entered is limited only by the available memory of the PNA.

See [Z5326A H08 Test Set Commands](#).

## Material Handler I/O

**Enable Control** Enables and disables sending data out the [Material Handler I/O connector](#)

**Ports A, B, C, D** Sends values to the respective Handler I/O port. Although ports C and D are normally bidirectional, ONLY Output mode is allowed using the Interface Control feature. It cannot read from these, or any other, ports.

## Test Set I/O

Note: The PNA has a separate interface for [controlling the E5091A Test Set](#).

**Enable Control** Enables and disables sending data out the [External Test Set I/O connector](#).

**Multi-line edit control** Each line contains a Write command using the following syntax:

```
address.value
```

Where:

**address** any positive integer.

**value** numeric character. Entries that require **alpha** characters should use the [GPIB interface](#).

Address and value are separated by a period. For example:

```
18.2  
27.3
```

Entries should be separated by a new line, or carriage return. The PNA front-panel **Enter** key inserts a new line into the field.

All entries are sent out the Test Set I/O port using the [WriteData Method](#).

The number of entries is limited only by the available memory of the PNA.

## Aux I/O

**Enable Control** Enables and disables sending data out the [Auxiliary I/O connector](#).

**DAC1, DAC2** Sets voltages on the Aux I/O connector pins 2 (DAC1) and pin 3 (DAC2).

**Dwell After Command** Specifies a wait time, in milliseconds, after all commands to all interfaces are sent. Any positive integer is allowed. This is used to allow all external devices to settle before beginning a measurement. An erratic trace could indicate that more settling time is necessary.

**Reset All** Sets ALL fields on ALL channels to their default values.

**Save and Recall** Saves and recalls the contents of this dialog box. If the Interface Control dialog box is populated with settings during an [Instrument State Save](#), the settings are automatically recalled with the Instrument State settings.

Interface control uses an \*.xml file type. An example file is stored on the PNA hard drive. You can recall it into the dialog, or you can open and edit it with a word processor, such as Word Pad.

**OK** Applies the settings and closes the dialog box.

**Cancel** Does not apply changes that were made, and closes the dialog box.

### Z5623A H08 Test Set Commands

The following table lists the commands that are used to control the popular Agilent Z5623A H08 Test Set. These commands can be entered into the [GPIB Interface](#) control.

Connection Path	Test Set Command
Reflection to Port 1	refl_01
Reflection to Port 2	refl_02
Reflection to Port 3	refl_03
Reflection to Port 4	refl_04
Reflection to Port 5	refl_05
Reflection to Port 6	refl_06
Reflection to Port 7	refl_07
Reflection to Port 8	refl_08
Transmission to Port 1	tran_01
Transmission to Port 2	tran_02
Transmission to Port 3	tran_03
Transmission to Port 4	tran_04
Transmission to Port 5	tran_05

Transmission to Port 6	tran_06
Transmission to Port 7	tran_07
Transmission to Port 8	tran_08
Reset	*rst
Reflection Termination	*r_term
Transmission Termination	*t_term
All Termination	*all_term

---

Last Modified:

4-Sep-2008    Removed legacy content



## Auxiliary I/O Connector

### General Description

This DB-25 male connector provides a variety of analog I/O, digital I/O, timing I/O, and supply lines. You can change the settings on the Auxiliary IO connector through [SCPI](#) and [COM](#) programming commands. The settings are NOT accessible through the front-panel keys or display menu.

**Note:** This connector does NOT exist on the PNA-X model. Instead, most of this functionality is present on the [PNA-X Power IO connector](#).

**Note:** The AUX IO configuration settings REMAIN after an [Instrument Preset](#) and [Hibernation](#). However, Preset will clear the [DAC](#) values. The settings will revert to their default settings ONLY after the PNA is restarted, or until they are changed by you. AUX IO settings are [saved and recalled with Instrument State](#).



Pin	Name	Description
1	<a href="#">ACOM</a>	Ground reference for analog signals
2	<a href="#">Analog Out 2</a>	-10 to +10Vdc output, 10mA max
3	<a href="#">Analog Out 1</a>	-10 to +10Vdc output, 10mA max
4	no connect	for future enhancements
5	<a href="#">DCOM</a>	Ground reference for digital signals
6	reserved	for future enhancements
7	reserved	for future enhancements
8	reserved	for future enhancements
9	<a href="#">+5V</a>	+5Vdc output, 100mA max.
10	<a href="#">Pass/Fail Write Strobe</a>	Indicates pass/fail line is valid (active low)
11	<a href="#">Sweep End</a>	Indicates sweep is done (programmable modes)

12	<a href="#">Pass/Fail</a>	Indicates pass/fail (programmable logic, modes and scope)
13	<a href="#">Output Port Write Strobe</a>	Writes I/O port data (active low)
14	<a href="#">Analog In</a>	-10 to +10VDC analog input
15	<a href="#">ACOM</a>	Ground reference for analog signals
16	<a href="#">Power Button In</a>	Grounding replicates front panel power button press
17	<a href="#">DCOM</a>	Ground reference for digital signals
18	<a href="#">Ready for Trigger</a>	Indicates ready for external trigger (active low)
19	<a href="#">External Trigger In</a>	Measurement trigger input (programmable to be active high or low)
20	<a href="#">Footswitch In</a>	Active low input latches a user-readable status bit.
21	<a href="#">+22V</a>	+22Vdc output, 100mA max.
22	<a href="#">In/Out port C0</a>	General purpose input / output
23	<a href="#">In/Out port C1</a>	General purpose input / output
24	<a href="#">In/Out port C2</a>	General purpose input / output
25	<a href="#">In/Out port C3</a>	General purpose input / output

### ACOM (pins 1, 15)

#### Description

Analog common (ground) - To be used with the Analog Out and Analog In lines.

ACOM and DCOM are connected to system ground at a star ground point inside the analyzer.

### Analog Out 1, 2 (pins 2, 3)

#### Description

Two analog outputs programmable to  $\pm 10\text{V}$ ;  $I_{\text{out}} < 10\text{mA}$ ;  $R_{\text{out}} = 100\ \Omega$

12-bit DACs with voltage resolution of approximately  $5\text{mV/count}$ .

The DACs are set to constant values using [SCPI](#) or [COM](#), and can be read using [SCPI](#) or [COM](#) commands.

Preset state for both pins is 0 volts.

#### HW Details

Looking into this output pin is a 100-ohm series resistor followed by two diodes tied to  $\pm 15\text{V}$  for static protection, then the output or an op-amp.

The voltage output is provided by a 12-bit DAC with an op amp buffer.

Specifics:

- Maximum output current = 10mA
- Settling time = 3us

#### Timing

The DACs are set after the last data point is measured, during retrace. If the analyzer is in single sweep mode, the DACs are set as part of the presweep process, before the sweep is triggered.

---

#### DCOM (pins 5, 17)

##### Description

Digital common (ground).

Used with the digital input and output lines.

ACOM and DCOM are connected to system ground at a star ground point inside the analyzer.

---

#### Pins 6, 7, 8

##### Description

Reserved

---

#### +5V (pin 9)

##### Description

+5V nominal output (100mA max).

Protected by self-healing fuse:

---

#### Pass/Fail Write Strobe (pin 10)

##### Description

See [Handler IO connector](#).

---

#### Sweep End (pin 11)

##### Description

See [Handler IO connector](#).

---

### Pass/Fail (pin 12)

#### Description

See [Handler IO connector](#).

---

### Output Port Write Strobe (pin 13)

#### Description

See [Handler IO connector](#).

---

### Analog In (pin 14)

Beginning with PNA Rev. A.07.21, voltages on this pin can be read using ADC receiver measurements. [Learn more](#).

#### Description

Analog input, +/-10V range,  $R_{in}=100k\ \Omega$

Bandwidth = 40kHz (2-pole lowpass filter).

This analog input may be read using the [SCPI](#) or [COM](#) commands.

#### HW Details

Looking into this pin there is 1k-ohm series resistor followed by 100k-ohm resistor to ground, static protection diodes after the 1k resistor limit the signal to +/-15V, then a high impedance buffer and active filter limiting the bandwidth to 40kHz with a lowpass filter.

---

### Power Button In (pin 16)

#### Description

Short this pin to ground to replicate a front panel power button key press.

#### HW Details

Looking into the pin there is a 215-ohm series resistor followed by a 10k pull-up to the 3V standby supply, static protection diodes to the 0V/5V and then connects to the front panel power key circuit.

CAUTION: Because this line is internally pulled up to 3V, it should not be driven by a TTL driver.

#### Timing

Grounding this line for 1us to 2 seconds will simulate pressing the front panel power button.

Grounding this line for >4 seconds will perform a hard reset (similar to a personal computer) and is not recommended.

---

### Ready for Trigger (pin 18)

#### Description

TTL output.

Active Low signal indicates that system is ready for an external trigger.

Remains High if system is not in External Trigger mode.

Goes High after an External Trigger is acknowledged.

Goes Low after the system has finished with its measurements, the source has been set up, and the next data point is ready to be measured.

#### HW Details

Looking into this pin there is a 215-ohm series resistor followed by a 10k pullup, diodes to 0V/5V for static protection, then the output of an "ABT" TTL buffer.

This line is enabled only when the analyzer is in External Trigger mode.

Refer to External Trigger In (following pin) for more information.

#### Timing

Refer to [External Trigger In](#) (following pin)

---

### External Trigger In (pin 19)

#### Description

This input accepts level trigger signals (High / Low) on all PNA models, or edge trigger signals on some PNA models.

The external trigger configuration is set from the front panel, [SCPI](#) or [COM](#).

For more information, see [External triggering](#).

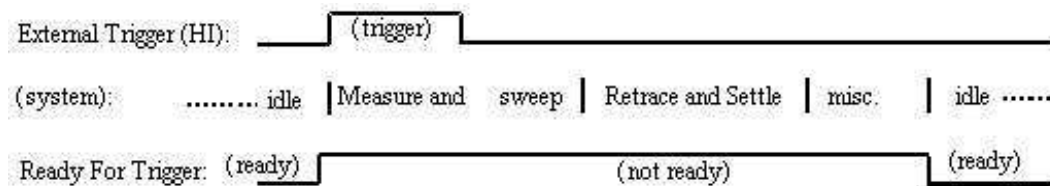
A single trigger is achieved by asserting the external trigger for a period from 1us to 50us. Continuous triggering is achieved by holding the external trigger in the "asserted" mode (either Low or High).

#### HW Details

Looking into this pin is a 215-ohm series resistor followed by a 4.64k pullup, 1000pF to ground and then a "FAST" TTL buffer input.

#### Timing

A level trigger width should be between 1us and 50us.



### Footswitch In (pin 20)

#### Description

TTL input.

A Low level input such as shorting this line to ground using a footswitch (where the input stays low for >1us) will be latched.

The latched status may be read using the [SCPI](#) or [COM](#) commands.

Only one footswitch press can be latched (remembered) by the system.

Reading the latch status will reset it if Footswitch In has returned to a high level.

## HW Details

Looking into this pin is a 215-ohm series resistor followed by a 4.64k pullup to 5V and 1000pF to ground. This line is an input to a "FAST" TTL buffer.

## Timing

Footswitch In must be Low for at least 1us.

---

## +22V (pin 21)

### Description

+22V nominal output (100mA max).

Protected by self-healing fuse.

---

## In/Out Port C0-C3 (pins 22-25)

### Description

See [Handler IO connector](#)

---

Last Modified:

23-Mar-2009    Added links to ADC Meas from pin 14

# External Test Set I/O Connector

## General Description

This DB-25 female connector is used to control external test sets. The external test set bus consists of 13 multiplexed address and data lines, three control lines, and an open-collector interrupt line. The Test Set IO is not compatible with the 8753 test sets.

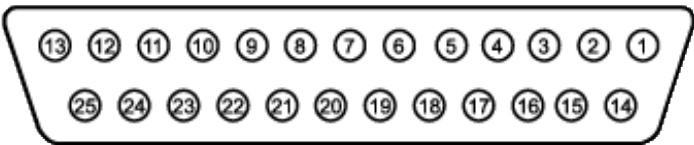
You can change the settings on the External Test Set IO connector through [SCPI](#) and [COM](#) programming commands. The settings are NOT accessible through the front-panel keys or display menu.

### Notes:

- The External Test Set pin settings are NOT affected by [Instrument State Save/Recall](#) or [Instrument Preset](#).
- At PNA Power Up and return from Hibernation, the External Test Set bus data lines, address lines, and control lines are set HIGH, and no strobe lines are pulsed.

**Caution:** Do not mistake this connector with a Parallel Printer port. A printer may be damaged if connected to this port.

## Other System Configuration Topics



Pin	Name	Description
1	<a href="#">SEL0</a>	Test set select bit 0; tied to GND
2	<a href="#">Sweep Holdoff In</a>	TTL input - state may be read with SCPI or COM command
3	<a href="#">AD12</a>	Address and latched data
4	<a href="#">AD10</a>	Address and latched data
5	<a href="#">AD9</a>	Address and latched data
6	<a href="#">AD8</a>	Address and latched data
7	<a href="#">GND</a>	0V
8	<a href="#">LAS</a>	TTL output    Low = Address Strobe

9	<a href="#">AD4</a>	Address and latched data
10	<a href="#">AD3</a>	Address and latched data
11	<a href="#">AD2</a>	Address and latched data
12	<a href="#">GND</a>	0V
13	<a href="#">Interrupt In</a>	TTL input - state may be read with a SCPI or COM command
14	<a href="#">No connect</a>	<b>CAUTION:</b> Older PNAs have +22v on this line; this will damage a printer.
15	<a href="#">SEL1</a>	Test set select bit 1; tied to GND
16	<a href="#">SEL2</a>	Test set select bit 2; tied to GND
17	<a href="#">AD11</a>	Address and latched data
18	<a href="#">SEL3</a>	Test set select bit 3; tied to GND
19	<a href="#">AD7</a>	Address and latched data
20	<a href="#">AD6</a>	Address and latched data
21	<a href="#">AD5</a>	Address and latched data
22	<a href="#">AD0</a>	Address and latched data
23	<a href="#">AD1</a>	Address and latched data
24	<a href="#">LDS</a>	TTL output - active low data strobe
25	<a href="#">RLW</a>	TTL output - high-read, low write

### SEL0-SEL3 (pins 1,15,16,18)

#### Description

Selects addresses of test sets that are "daisy chained" to this port. The select code is set to zero at the PNA connector and is incremented by one as it goes through each successive external test set. Therefore, the first test set in the chain has address zero and so on, for up to 16 test sets.

#### HW Details

Connected to ground inside the PNA.

#### Timing

None

### Sweep Holdoff In (pin 2)

#### Description



Input line used by the test set for holding off a sweep. Holding off a sweep is one way of introducing a delay that allows an external device to settle before the PNA starts taking data. You must write a program that will query the line and perform the delay. The program needs to query the line and keep PNA from sweeping while the line remains low. When a subsequent query detects that the line went high the program would then trigger the PNA to start the sweep.

Use either Single or External trigger mode to control the PNA sweep.

#### HW Details

This pin has a series 215-ohms resistor followed by 4.7k-ohm pull-up and then an "ABT" TTL buffered register.

#### Timing

This input is not latched by the PNA hardware. Therefore the input level must be held at the desired state by the test set until it's read by your program.

---

### AD0-AD12 (pins 3-6, 9-11, 17, 19-23)

#### Description

Thirteen lines are used to output data addresses or input / output data. Several [SCPI](#) and [COM](#) commands are available for reading and writing to these lines. You can choose to use commands where the PNA provides the appropriate timing signals needed for strobing the addresses and data. Or you can choose to control the timing signal directly. The timing signals are RLW, LAS and LDS. If you decide to do direct control refer to the corresponding SCPI and COM command details. Close attention to detail is needed to insure the desired results.

After a write command, lines AD0-AD12 are left in the state they were programmed. Default setting for Mode is Read / Input).

After a read command, lines AD0-AD12 are left in input mode. While in this mode an external test set attached to the IO is free to set the level on each line.

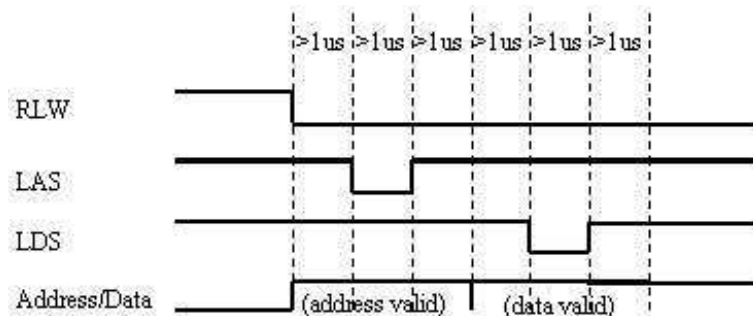
#### HW Details

Each of these I/O pins has a series 215-ohm resistor followed by 4.7k-ohm pull-up resistor.

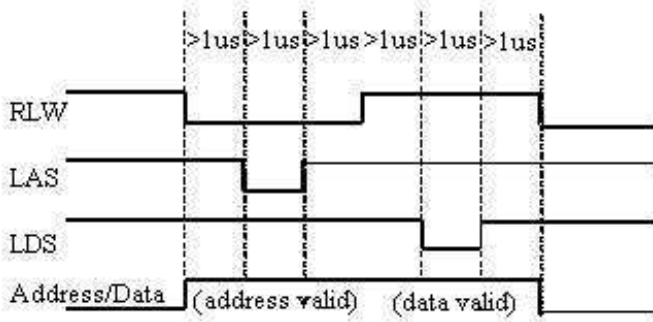
Write/Read is implemented by an output tri-state TTL buffer / latch for latching and enabling write data in parallel with a TTL input buffer for reading.

#### Timing

Output Address and data setup and hold times are 1us minimum.



#### Address & Data I/O Write



**Address & Data I/O Read** - Data must be valid for 1us before and after strobe

---

## GND (pins 7, 12)

### Description

Two ground pins used as ground references by the test set.

### HW Details

Connected to digital ground.

### Timing

None.

---

## LAS (Low Address Strobe) (pin 8)

### Description

This line has two behaviors that are command dependent. Refer to the [SCPI](#) and [COM](#) commands for further details.

In one behavior LAS is one of the lines used by the PNA to provide appropriate timing for writing Address and Data to the Test Set. In this case LAS is controlled automatically by the PNA and is intended to be used as the strobe for the Address. When LAS is low, lines AD0 - AD12 represent the Address. LAS will return to its normally high state when the transaction is finished.

In the second behavior the PNA will NOT provide appropriate timing. In this case LAS is controlled directly by the user through a [SCPI](#) or [COM](#) command. When the transaction is finished LAS is left set to the state it was programmed to until another command changes it. (Default for LAS is TTL High).

### HW Details

This output pin is driven by a TTL latched buffer with a series 215-ohm resistor followed by 2.15k-ohm pull-up.

### Timing

Strobe length, setup and hold times are all 1us minimum.

See the description for [AD0-AD12](#) for more timing information.

---

## Interrupt In (pin 13)

### Description

Query this line with a [SCPI](#) or [COM](#) command.

### HW Details

This line is a non-latched TTL input, has series 215-ohms followed by 4.64k-ohm pullup.

#### Timing

The Test Set must maintain at the desired TTL level until its read.

---

#### (pin 14) No Connect (previously +22V )

**WARNING:** Early versions of the PNA had +22v on this pin. **Connecting a printer to this port will usually damage the printer.**

#### Description

+22V, 100mA max. The 25-pin D connector is the same as a computer parallel printer port connector. Pin (14) corresponds to a printer's "autofeed" line. **Connecting a printer to this port will damage the printer if +22v is present** since printers requires less than 5V on all control lines.

#### HW Details

No connect

#### Timing

None

---

#### LDS (Low Data Strobe) (pin 24)

##### Description

This line has two behaviors that are command dependent. Refer to the External Test Set IO SCPI and COM commands for further details. (Default setting for LDS is TTL High)

In one behavior LDS is one of lines used by the PNA to provide appropriate timing for writing Address and Data to the Test Set. In this case LDS is controlled automatically by the PNA and is intended to be used as the strobe for the Data. When LDS is low, lines AD0 - AD12 represents Data. LDS will return to its normally high state when the transaction is finished.

In the second behavior the PNA will NOT provide appropriate timing. In this case LDS is controlled directly by the user through a SCPI or COM command. When the transaction is finished the LDS is left set to the state it was programmed to.

##### HW Details

This output pin is driven by a TTL latched buffer with a series 215-ohm resistor followed by 2.15k-ohm pull-up.

##### Timing

Strobe length, setup and hold times are all 1us minimum.

See the description for [AD0-AD12](#) for more timing information.

---

#### RLW (pin 25)

##### Description

This line is the output for the Read Write signal. It has two behaviors that are command dependent. Refer to the External Test Set IO SCPI and COM commands for further details. (Default setting for RLW is TTL High)

In one behavior RWL is controlled automatically by the PNA during a Read Write operation. When RLW is low, lines AD0 - AD12 represent output Data. When RLW is high, the lines represent input Data.

In the second behavior the PNA does NOT provide the timing. The user must control it directly through the SCPI or COM command. In this case the line is left set to the state it was programmed to.

#### **HW Details**

This pin is a TTL latched output with a series 215-ohm resistor followed by 2.15k-ohm pull-up resistor.

#### **Timing**

Strobe length, setup and hold times are all 1us minimum.

See the description for [AD0-AD12](#) for more timing information.

---

## Material Handler I/O Connector

---

This [rectangular 36-pin female connector](#) provides communication signals between the PNA and a material parts handler. You can change the settings on the Material Handler IO connector using [SCPI](#) and [COM](#) commands. The settings are NOT accessible through the front-panel keys or display menu.

- [Overview - Controlling a Material Handler](#)
- [Pin Assignments](#)
- [Pin Descriptions](#)
- [Timing Diagrams](#)
- [Input Output Electrical Characteristics](#)

**Note:** On early PNAs this connector is labeled "GPIO". It is covered to indicate that the connector is not functional.

### Overview - Controlling a Material Handler

The PNA is capable of interacting with an external material handler or part handler. This allows the PNA to be used in an automated test environment, where devices to be tested are inserted into a test fixture by a part handler, and sorted into pass/fail bins by the handler after testing is complete. By connecting the part handler to the PNA Auxiliary or Material Handler I/O ports, the PNA and part handler can synchronize their activities in a way that makes automated testing possible.

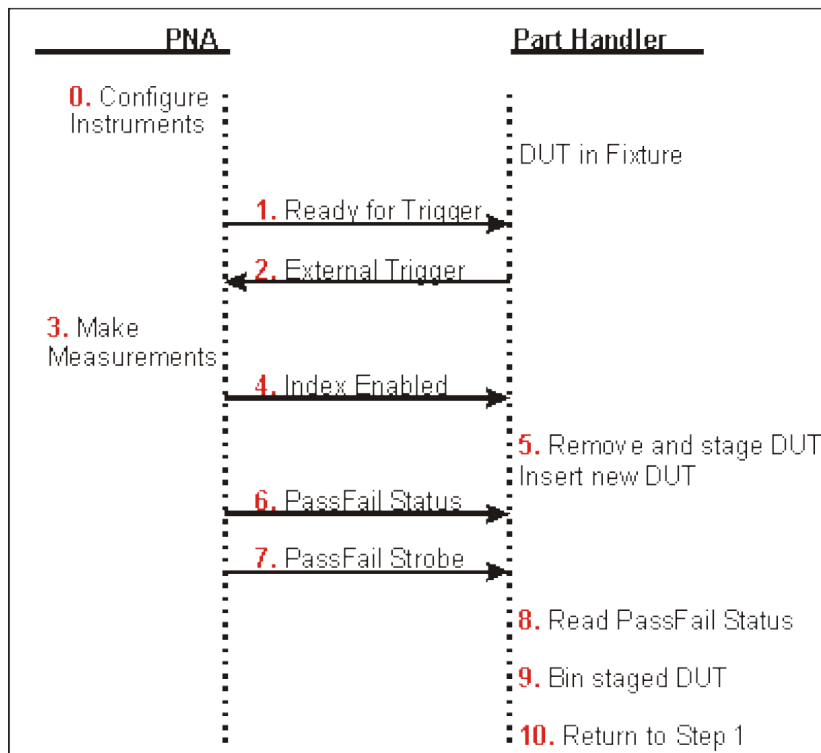
### PNA and Part Handler Preparation

1. [Define the measurements](#) you want to make.
2. [Define limits](#) for each of the measurements.
3. Configure the PNAs Material Handler port so that it is compatible with your part handler. This usually involves setting the [handler logic](#), [pass/fail logic](#), [pass/fail scope](#), and [pass/fail mode](#). These settings are made remotely using [SCPI](#) or [COM](#) commands.
4. Use a cable to connect the PNA to your part handler.
5. Put the PNA in [External Trigger](#) mode.
6. Load parts in handler per manufacturer instructions.

**Note:** The Material Handler configuration settings REMAIN after an [Instrument Preset](#) and [Hibernation](#). The settings will revert to their default settings ONLY after the PNA is restarted, or until they are changed by you. Material Handler settings are [saved and recalled with Instrument State](#).

### Flow Diagram

The following diagram and descriptions summarizes the events that occur during automated testing. 'DUT' refers to Device Under Test.

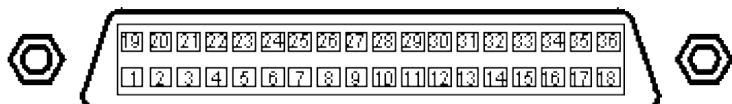


### Text Descriptions

0. (Optional). The PNA sends values out the Material Handler and/or [Auxiliary I/O](#) connectors to configure external instruments. The A,B,C, and D ports of the Material Handler can be used to control devices used in testing, such as step attenuators, part handlers, or even the DUT itself. Also, the DAC1 and DAC2 lines on the Aux I/O connector can be used to provide bias voltages for devices and instruments. If you wish to use the Material Handler or Aux connectors for testing, you will need to write a program to send values out the various lines and ports, as there is no activity on these lines by default.
1. The part handler receives a [Ready for Trigger](#) signal from the PNA. This indicates that the PNA is properly configured and ready to take a measurement.
2. The part handler sends an [External Trigger](#) signal to the PNA. This signals that the part handler has settled, and allows the PNA to begin taking measurements.
3. The PNA takes measurements on all triggerable channels.
4. The [Index line](#) on the material handler goes to a Low state, which means that all required data has been collected by the PNA.
5. The part handler removes the DUT from the test fixture, and inserts a new DUT into the fixture. This operation is often referred to as part handler indexing. The device just tested is staged (removed from the fixture and prepared for binning), and the next part to be tested is put into the fixture. The removed DUT cannot be assigned to a Pass/Fail bin yet, as the Pass/Fail status is not available.
6. The PNA sends the [Pass/Fail Status](#).
7. The PNA sends the [Pass/Fail Strobe](#) meaning that the Pass/Fail status has been determined.

8. The part handler reads the Pass/Fail Status line.
9. The part handler bins the staged part based on the Pass/Fail Status.
10. The test process repeats at step 1, waiting for Ready for Trigger from the PNA.

## Material Handler IO Pin Assignments



There are three different Handler IO pin assignment configurations depending on the PNA model:

- **Type 1** - All E835xA, E880xA, N338xA models. You can change the pinout configuration to Type 2 on these models. This requires opening the instrument and changing a connector internally. Refer to the procedure in the Service Guide, Chapter 7. You can download the Service Guide for your PNA model from our website: <http://na.tm.agilent.com/pna> **Caution:** Changing this connection should be done by qualified service personnel.
- **Type 2** - E8362x, E8363x, E8364x and N5230 Opts 220, 225, 420, 425, 520, 525;
- **Type 3** - All PNA-X and N5230 Opts 020, 025, 120, 125, 140, 145, 146, 240, 245, 246.

[See PNA models and options.](#)

**Shaded/bold** indicates changes from Type 1

**Note:** A slash (/) preceding the signal names indicates that the signal uses negative (active low) logic. A low pulse is a logical 1.

Pin	Type 1	Type 2	Type 3
1	Ground	Ground	Ground
2	/ INPUT1	/ INPUT1	/ INPUT1
3	/ OUTPUT1	/ OUTPUT1	/ OUTPUT1
4	/ OUTPUT2	/ OUTPUT2	/ OUTPUT2
5	/ Output port A0	/ Output port A0	/ Output port A0
6	/ Output port A1	/ Output port A1	/ Output port A1
7	/ Output port A2	/ Output port A2	/ Output port A2
8	/ Output port A3	/ Output port A3	/ Output port A3
9	/ Output port A4	/ Output port A4	/ Output port A4
10	/ Output port A5	/ Output port A5	/ Output port A5

11	/ Output port A6	/ Output port A6	/ Output port A6
12	/ Output port A7	/ Output port A7	/ Output port A7
13	/ Output port B0	/ Output port B0	/ Output port B0
14	/ Output port B1	/ Output port B1	/ Output port B1
15	/ Output port B2	/ Output port B2	/ Output port B2
16	/ Output port B3	/ Output port B3	/ Output port B3
17	/ Output port B4	/ Output port B4	/ Output port B4
18	no connect	/ Output port B5	/ Ext. Trigger
19	/ Output port B5	/ Output port B6	/ Output port B5
20	/ Output port B6	/ Output port B7	/ Output port B6 -or / Index Signal <a href="#">Learn more</a>
21	/ Output port B7	/ In/Out port C0	/ Output port B7 or / Ready for Trigger <a href="#">Learn more</a>
22	/ In/Out port C0	/ In/Out port C1	/ In/Out port C0
23	/ In/Out port C1	/ In/Out port C2	/ In/Out port C1
24	/ In/Out port C2	/ In/Out port C3	/ In/Out port C2
25	/ In/Out port C3	/ In/Out port D0	/ In/Out port C3
26	/ In/Out port D0	/ In/Out port D1	/ In/Out port D0
27	/ In/Out port D1	/ In/Out port D2	/ In/Out port D1
28	/ In/Out port D2	/ In/Out port D3	/ In/Out port D2
29	/ In/Out port D3	Port C Status	/ In/Out port D3
30	Port C Status	Port D Status	Port C Status
31	Port D Status	/ Output Port Write Strobe	Port D Status
32	/ Output Port Write Strobe	no connect	/ Output Port Write Strobe
33	/ Pass/Fail	/ Pass/Fail	/ Pass/Fail
34	/ Sweep End	+5V	/ Sweep End



35	+5V	/ Sweep End	+5V
36	/ Pass/Fail Write Strobe	/ Pass/Fail Write Strobe	/ Pass/Fail Write Strobe

### Pin Descriptions

#### Input1

When this Input line receives a Low pulse from the material handler, data is latched on the [OUTPUT1 and OUTPUT2](#) lines. See [OUTPUT1|2 Data Output Write Timing](#)

**Note:** [Type 1 and Type 2 Behavior](#): The Input line responds to a High (rising edge) pulse.

The Input Line activity can be read:

SCPI	COM
<a href="#">CONTrol:HANDler:INPut?</a>	<a href="#">get_Input1 Method</a>

#### Output1, Output2

See [OUTPUT1|2 Data Output Write Timing](#)

The **current** state of these latched TTL outputs may be set High or Low (Default setting) using the (non-user) [SCPI put\\_Output \(COM\)](#) commands.

The **next** state (following a negative edge on the INPUT1 line) may be pre-loaded to High or Low (Default setting) using the user commands.

For example, on the next negative pulse on the INPUT1 line, you want the OUTPUT1 line to go from 0 to 1. To do this:

```

CONT:HAND:OUTP1:DATA 0 'Force the OUTPUT1 line to 0
CONT:HAND:OUTP1:USER 1 'Set the OUTPUT1:USER buffer to 1, indicating the next state

```

	SCPI	COM
Write User Data	<a href="#">CONT:HAND:OUTP&lt;pin&gt;:USER</a>	<a href="#">put_Output Method</a>
Read last value written	<a href="#">CONT:HAND:OUTP&lt;pin&gt;:USER</a>	<a href="#">get_Output Method.</a>
Write non-user data	<a href="#">CONT:HAND:OUTP&lt;pin&gt;:DATA</a>	<a href="#">put_Output Method</a>
Read last value written	<a href="#">CONT:HAND:OUTP&lt;pin&gt;:DATA</a>	<a href="#">get_Output Method</a>

## Output Ports A and B

These two general purpose, 8-bit output ports are used to write data to the material handler. When any line changes state, all output lines are latched to the I/O connector as the [Output Write Strobe](#) goes Low.

The default state for data is Low.

[See Data Output Write Timing Diagram](#)

### Set Port Logic:

The logic for the data lines can be set to either: Positive (1 = High) or Negative (1 = Low). This setting affects all data ports. They cannot be set independently.

#### SCPI

[CONTrol:HANDler:LOGic](#)

#### COM

[PortLogic Property](#)

### Combine to read or write data to Port F:

Ports A and B can be virtually combined to write data to one 16-bit I/O port F.

#### SCPI

[CONTrol:HANDler:F <num>](#)

#### COM

[put Port \(F\)](#)

## Input/Output Ports C and D

These two general purpose 4-bit Input/Output ports are used to write data (Output) or read data (Input). These lines could be used to write to an external device such as a step attenuator.

When any line changes state, all output lines are latched to the I/O connector as the [Output Write Strobe](#) goes Low. [See Data Output Write Timing](#)

The four lines of Port C are connected internally to the [Auxiliary IO](#) connector.

### Set Input | Output Mode:

Each port may be independently defined as Output or Input.

#### SCPI

[CONTrol:HANDler:C:MODE](#)

[CONTrol:HANDler:D:MODE](#)

#### COM

[PortMode Property](#)

### Set Port Logic:

The logic for the data lines can be set to either: Positive (1 = High) or Negative (1 = Low). This setting affects all data ports. They cannot be set independently.

#### SCPI

[CONTRol:HANDler:LOGic](#)

#### COM

[PortLogic Property](#)

### Read or write data:

Ports C and D can be virtually combined to read or write data to one 8-bit I/O port **E**. When combined, **both** C and D ports must be set to either INPUT or OUTPUT mode.

#### SCPI

[CONTRol:HANDler:<port>\[:DATA\]>](#)

#### COM

[get\\_Port\(x\)](#)

[put\\_Port \(x\)](#)

### Port C Status, Port D Status

These two output lines indicate the Read / Write mode of the C and D ports.

- A Low level indicates that the associated port is in **INPUT** mode (read only).
- A High level indicates that the associated port is in **OUTPUT** mode (write only).

These logic of these status outputs cannot be changed.

See [Input/Output Ports C and D](#) to learn how to set I/O Mode

[See Data Output Write Timing](#)

### Output Port Write Strobe

This Output line goes Low to write data from [Ports A and B](#) and [Ports C and D](#) when a change is detected on any of the data lines.

These logic of this strobe output cannot be changed.

This line is shared with Auxiliary IO connector.

[See Data Output Write Timing](#)

### External Trigger

When trigger source is set to external, this Input line accepts a trigger signal from the material handler. This usually means that a part is in place and ready to be tested.

[See Trigger Timing Diagram](#)

### Index

A Low signal on this Output line indicates to the material handler that the measurement is complete. This usually means that the handler can connect the next device. However, measurement data is not available until data is calculated. [See Trigger Timing Diagram](#).

#### Set Function:

This line also serves as a data line. Set the function using the following commands:

#### SCPI

[CONTrol:HANDler:INDex:STATe](#)

#### COM

[IndexState](#)

### Ready for Trigger

When this output line goes low, it indicates to the material handler that the PNA is ready for a trigger signal. Available only on Type 3 PNA Models (Pin 17). See also [Aux IO connector](#).

[See Trigger Timing Diagram](#)

[See Pass/Fail Timing Diagram](#)

#### Set Function:

This line also serves as a data line. Set the function using the following commands:

#### SCPI

[CONTrol:HANDler:RTRigger:STATe](#)

#### COM

[ReadyForTriggerState](#)

## Pass/Fail State

This Output line indicates to the handler whether the limit test has passed or failed.

Pass/Fail state is valid only when the [limit test](#) function is ON and while [Pass/Fail strobe](#) line is Low. [See Pass/Fail Timing Diagram](#)

This line is shared with the Auxiliary IO connector.

### Set Pass / Fail Logic:

- Positive Logic: High=Pass, Low=Fail. (Default setting)
- Negative Logic: High=Fail, Low=Pass.

#### SCPI

[CONTrol:HANDler:PASSfail:LOGic](#)

#### COM

[PassFailLogic Property](#)

### Set Default Conditions:

- **PASS**- the line stays in PASS state. When a device fails, then the line goes to fail after the Sweep End line is asserted.
- **FAIL**- the line stays in FAIL state. When a device passes, then the line goes to PASS state after the Sweep End line is asserted.
- **No Wait**- the line stays in PASS state. When a device fails, then the line goes to fail IMMEDIATELY. (Default setting)

#### SCPI

[CONTrol:HANDler:PASSfail:MODE](#)

#### COM

[PassFailMode Property](#)

### Set Pass / Fail Scope:

- **Channel scope**: The line resets to the default state after the measurements on a channel have completed.
- **Global scope**: The line resets to the default state after the measurements on all triggerable channels have completed. (Default setting)

#### SCPI

[CONTrol:HANDler:PASSfail:SCOPE](#)

#### COM

[PassFailScope Property](#)

### Pass/Fail Write Strobe

A Low pulse indicates that [Pass/Fail](#) line is valid and the Pass / Fail State is output to the material handler.

This line is shared with the Auxiliary IO connector.

The Pass/Fail Strobe is fixed in duration and timing. However, when the strobe occurs depends on the Pass/Fail Mode and Pass/Fail Scope (Channel or Global) settings. [See Pass/Fail State](#)

[See Pass/Fail Timing Diagram](#)

### +5V

+5V nominal output (100mA max).

Protected by self-healing fuse.

### Sweep End

This output line indicates the status of the PNA sweep. The sweep includes sweeping the source and taking data.

- **Low** (falling edge) indicates that the specified sweep event has finished. This does NOT indicate that all calculations have finished.
- **High** indicates that the specified sweep event is active.

[See Trigger Timing Diagram](#)

This line is shared with the Auxiliary IO connector.

#### Set Sweep Event Mode:

- **Sweep**: indicates that a single source sweep has finished. (Default setting)
- **Channel**: indicates that a single channel has finished.
- **Global**: indicates that all enabled channels have finished.

SCPI

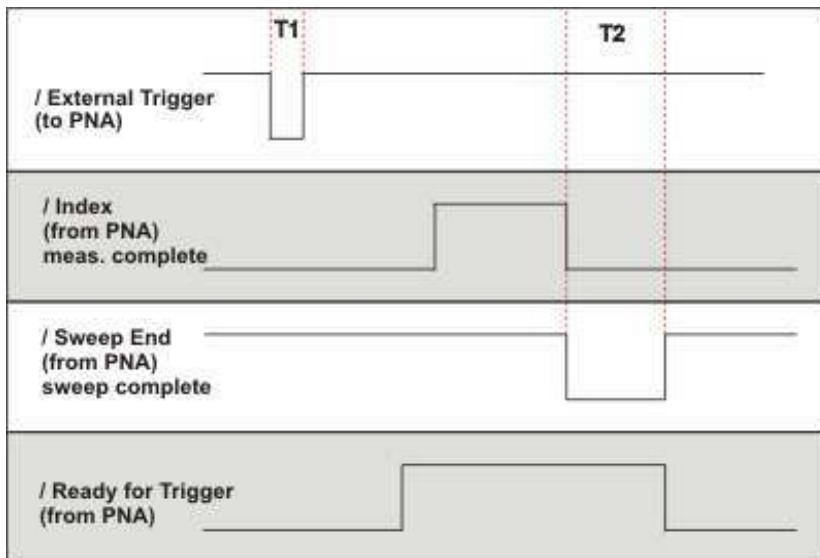
[CONTrol:HANDler:SWEepend](#)

COM

[SweepEndMode Property](#)

## Timing Diagrams

### Trigger Timing

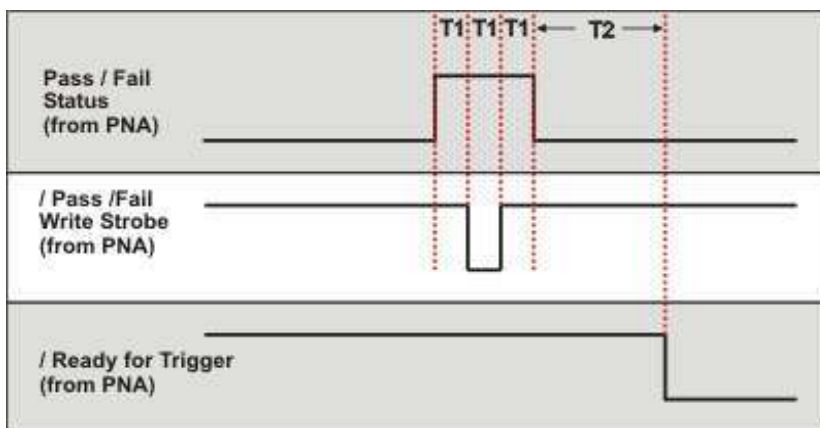


All signals are active low.

**T1 = 1 s** External Trigger pulse width

**T2 > 10 s** Sweep End pulse width (both High and Low)

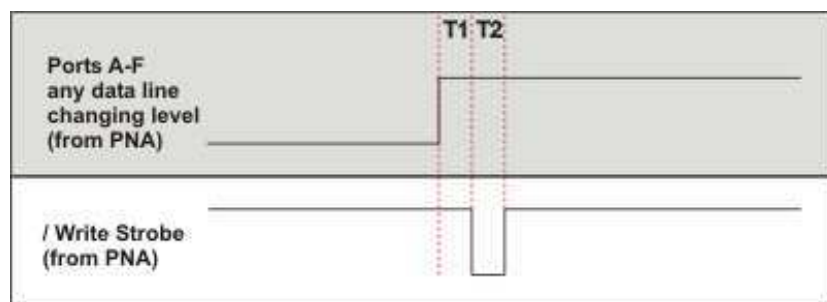
### Pass / Fail Timing



**T1 = 1 s** Pulse width and response time of Pass / Fail Strobe

**T2 > 10 s** Ready for Trigger lag

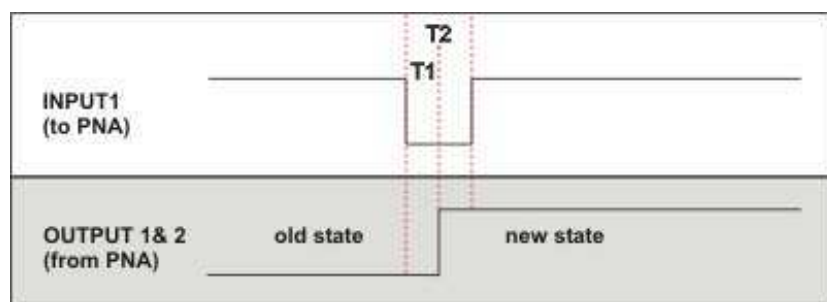
## Ports A-F Data Output Write Timing



**T1 = 1 s** Write Strobe response time

**T2 = 1 s** Write Strobe pulse width

## OUTPUT1|2 Data Output Write Timing



The old state to new state transition can be either low to high (as shown) or high to low.

**T1 = .6 s** [Output1|2](#) response time

**T2 = 1 s** [Input1](#) Strobe pulse width

## Input / Output Electrical Characteristics

All Material Handler I/O Input and Output lines are TTL compatible.

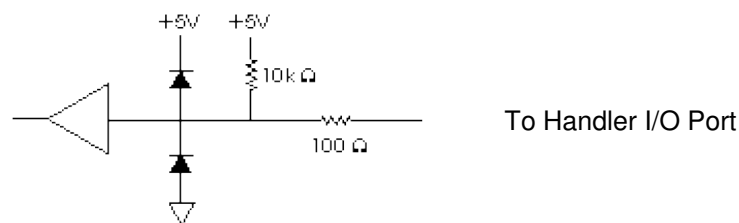
### Input and Input/Output lines

Lines carrying information IN (or bidirectional) to the PNA from the material handler.

Maximum Input Voltages:	-0.5 V to 5.5 V
TTL High level:	2.0 V to 5.0 V
TTL Low level:	0 V to 0.5 V



**PNA Input and Input/Output Circuit Diagram**



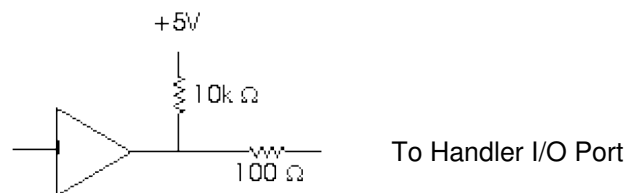
**Note:** The [INPUT1](#) line does NOT have the 10K pullup resistor.

**Output Lines**

Lines carrying information OUT of the PNA to the material handler.

Maximum Output Current:		-10 mA to 10 mA
Output Current	TTL High level:	-5 mA
	TTL Low level:	3 mA
Output Voltage	TTL High level:	2.0 V to 3.3 V
	TTL Low level:	0 V to 0.8 V

**PNA Output Circuit Diagram**



Last modified:

6-Apr-2009    Replaced N5242A with PNA-X

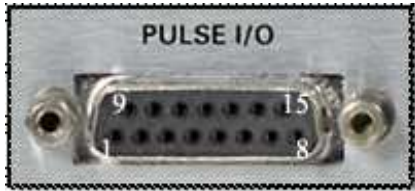
8-Feb-2008    Clarified Types

Nov 3, 2006    Added link from pin 20

## Pulse I/O Connector

New with the PNA-X, the 15 pin D connector provides access to Pulse Modulators and Generators.

- [See IF Path Configuration Dialog and block diagram](#), which includes the Pulse Modulators and Generators.
- [See the Pulsed Application \(Opt H08 / 008\)](#)



Pin	Name	Description
1	IFGateAIn	IF pulse gate input A (TTL)
2	IFGateBIn	IF pulse gate input B (TTL)
3	IFGateCIn	IF pulse gate input C (TTL)
4	IFGateDIn	IF pulse gate input D (TTL)
5	IFGateRIn	IF pulse gate input R (TTL)
6	DCOM	Ground
7	PulseSyncIn	Pulse gen. synchronization trigger input (TTL)
8	RFPulseModIn	RF source pulse modulation drive input (TTL)
9	DCOM	Ground
10	Pulse1Out	Hardwired pulse train output #1 (TTL)
11	Pulse2Out	Hardwired pulse train output #2 (TTL)
12	Pulse3Out	Hardwired pulse train output #3 (TTL)
13	Pulse4Out	Hardwired pulse train output #4 (TTL)
14	N.C.	No connect -- for future use
15	DCOM	Ground

See Pulse [SCPI](#) and [COM](#) commands

## N1966A Pulse I/O Adapter



This D connector to RF adapter makes accessing the Pulse I/O connector more convenient.

---

Last Modified:

16-Jul-2007 Clarification

18-Jan-2007 MX New topic

## Power I/O Connector

New on the PNA-X, the 9-pin D connector replaces much of the functionality of the [AUX I/O connector](#) on older PNA models. [See PNA-X Rear Panel](#)



Pin	Name	Description
1	+15V	+15V @ 400 mA
2	-15V	-15V @ 400 mA
3	AnalogOut1	Analog Output Voltage Programmable +/-10V @100 mA out Nominally 0 ohms 2.44mV typical resolution 1MHz BW Read and write voltage programmatically using: <ul style="list-style-type: none"><li>• <a href="#">CONT:AUX:OUTP1:VOLT</a> (SCPI - read and write)</li><li>• <a href="#">get OutputVoltage Method</a> (COM - read)</li><li>• <a href="#">put OutputVoltage Method</a> (COM - write)</li></ul>
4	AnalogOut2	Analog Output Voltage Programmable +/-10V @100 mA out Nominally 0 ohms 2.44mV typical resolution 1MHz BW Read and write voltage programmatically using: <ul style="list-style-type: none"><li>• <a href="#">CONT:AUX:OUTP2:VOLT</a> (SCPI - read and write)</li><li>• <a href="#">get OutputVoltage Method</a> (COM - read)</li><li>• <a href="#">put OutputVoltage Method</a> (COM - write)</li></ul>
5	ACOM	System ground
6	GndSense	Ground sense for Analog In & Out

		<p>Connected with 51.1-ohm to ACOM</p> <p>Read voltage programmatically using:</p> <ul style="list-style-type: none"> <li>• <a href="#">InputVoltageEX Property</a> (COM)</li> <li>• <a href="#">CONT:AUX:OUTP3:VOLT</a> (SCPI)</li> </ul>
7	AnalogIn1	<p>Analog input:</p> <p>+/-10V @ 1.22mV typical resolution</p> <p>Rin &gt;1 M-ohm</p> <p>BW ~ 1 MHz</p> <p>ADC conversion time &lt; 1 us typical</p> <p>Read voltage programmatically using:</p> <ul style="list-style-type: none"> <li>• <a href="#">CONT:AUX:INP1:VOLT</a> (SCPI)</li> <li>• <a href="#">InputVoltageEX Property</a> (COM)</li> </ul> <p>Voltages on this pin can also be read using ADC receiver measurements. <a href="#">Learn more.</a></p>
8	AnalogIn2	<p>Analog input:</p> <p>+/-10V @ 1.22mV typical resolution</p> <p>Rin &gt;1 M-ohm</p> <p>BW ~ 1 MHz</p> <p>ADC conversion time &lt; 1 us typical</p> <p>Read voltage programmatically using:</p> <ul style="list-style-type: none"> <li>• <a href="#">CONT:AUX:INP2:VOLT</a> (SCPI)</li> <li>• <a href="#">InputVoltageEX Property</a> (COM)</li> </ul> <p>Voltages on this pin can also be read using ADC receiver measurements. <a href="#">Learn more.</a></p>
9	Power Button	<p>Open collector input</p> <p>Active low replicates power button key press.</p>

Last Modified:

23-Mar-2009	Added links to ADC meas.
13-Aug-2008	Edited for InputVoltageEx
10-Jul-2007	Added COM commands
18-Jan-2007	MX New topic

## New Programming Commands

The following are new programming commands for **PNA release A.09.22** See What's New

### Port Mapping - Noise Figure Opt 028

Write port mapping	SENSe:NOISe:PMAP	SetPortMap
Read input port mapping	SENSe:NOISe:PMAP:INPut?	DeviceInputPort
Read output port mapping	SENSe:NOISe:PMAP:OUTPut?	DeviceOutputPort

The following are new programming commands for **PNA release A.09.20** See What's New

### Configure Pulse Measurements

Pulse Meas Mode	SENS:SWE:PULS:MODE	PulseMeasMode
Autodetect Pulse mode	SENS:SWE:PULS:DETECTmode	AutoDetection
Set Pulse Mode (Narrow   Wide)	SENS:SWE:PULS:WIDeband	WideBandDectionState
Autoselect IFBW	SENS:SWE:PULS:IFBW	AutoIFBandWidth
Autoselect IF Gain	SENS:SWE:PULS:IFGain	AutoIFGain
Autoselect Pulse clock period	SENS:SWE:PULS:PRF	AutoOptimizePRF
Autoselect Width and Delay	SENS:SWE:PULS:TIMing	AutoPulseTiming
Autoselect Pulse Gens	SENS:SWE:PULS:DRIVe	AutoSelectPulseGen

### External Pulse Configuration

PulseSyncIn Trigger Polarity	Sense:Pulse:TPolarity	TriggerInPolarity
PulseSyncIn Trigger Type	Sense:Pulse:TType	TriggerInType

## PSAT Marker Search

Initiate a PSAT search	CALC:MARK:PSAT:BACKoff	SearchPowerSaturation
Set and read PSAT backoff	CALC:MARK:PSAT:BACKoff	PMaxBackOff
Read PSat Out	CALC:MARK:PSAT:POUT?	POut
Read PSat In	CALC:MARK:PSAT:PIN?	Pin
Read PMax Out	CALC:MARK:PSAT:POUT:MAXimum?	PMaxOut
Read PMax In	CALC:MARK:PSAT:PIN:MAXimum?	PMaxIn
Read Gain Sat	CALC:MARK:PSAT:GAIN?	GainSaturation
Read Gain Max	CALC:MARK:PSAT:GAIN:MAXimum?	GainMax
Read Gain Linear	CALC:MARK:PSAT:GAIN:LINear?	GainLinear
Read Comp Sat	CALC:MARK:PSAT:COMP:SAT?	CompressionSaturation
Read Comp Max	CALC:MARK:PSAT:COMP:MAX?	CompressionMax

## PNOP Marker Search

Initiate a PNOP search	CALC:MARK:PNOP:BACKoff	SearchPowerNormalOperatingPoint
Set and read PNOP backoff	CALC:MARK:PNOP:BACKoff	BackOff
Set and read PNOP Power Offset	CALC:MARK:PNOP:POFFset	PinOffset
Read Pnop Out	CALC:MARK:PNOP:POUT?	POut
Read Pnop in	CALC:MARK:PNOP:PIN?	Pin
Read Pnop Gain	CALC:MARK:PNOP:GAIN?	Gain
Read Pnop Comp	CALC:MARK:PNOP:COMPression?	Compression
Read PMax Out	CALC:MARK:PNOP:POUT:MAXimum?	PMaxOut
Read PMax In	CALC:MARK:PNOP:PIN:MAXimum?	PMaxIn
Read Gain Max	CALC:MARK:PNOP:GAIN:MAXimum?	GainMax
Read Comp Max	CALC:MARK:PNOP:COMPression:MAXimum?	CompressionMax



Read PBO Out	CALC:MARK:PNOP:BACKoff:POUT?	BackOffPout
Read PBO In	CALC:MARK:PNOP:BACKoff:PIN?	BackOffPin
Read PBO Gain	CALC:MARK:PNOP:BACKoff:GAIN?	BackOffGain

### Include Phase in SMC measurements

Enable Phase	SENS:MIX:PHASe	EnablePhase
Set normalize point	SENS:MIX:NORMalize	NormalizePoint
Choose known delay or S2P	SENS:CORR:COLL:SESS:SMC:PHASe:METHod	DelayCalculationMethod
Set known delay	SENS:CORR:COLL:SESS:SMC:PHASe:DELay	FixedDelay
Set Cal Mixer Char S2P filename	SENS:CORR:COLL:SESS:SMC:PHASe:MIXer	MixerCharacterizationFile

### Group Delay Aperture

Set aperture using frequency	CALC:GDELay:FREQ	Frequency
Set aperture using percent of span	CALC:GDELay:PERCent	Percent
Set aperture using fixed number of points	CALC:GDELay:POINTs	Points
Set Preference to 2 points	SYST:PREF:ITEM:GDEL:TWOP	TwoPointGroupDelayAperture

### Calibrations

Returns the names of the mechanical cal kits for unguided calibrations.	SENSe:CORR:COLL:CKIT:CAT?	CalKitTypes
Set/get by name which cal kit is currently selected for use by unguided cal.	SENSe:CORR:COLL:CKIT:PORT	CalKitType
Read pass fail status of tolerance limits on the target cal power	Not needed - read from SCPI error queue	LastCalPassedTolerance
Gets the power correction dB values from the prior iteration of the source power cal.	SOURce:POWER:CORR:DATA:PRIor	getSourcePowerCalDataEx getSourcePowerCalDataScalarEx

## Fixturing

Compensate source power	CALC:FSIM:SEND:POW:PORT:COMP	EnablePowerCompensation
Create Cal Set with De-embedded fixture removed.	CSET:FIXTure:DEEMbed	None
Create Cal Set with Matching Network included.	CSET:FIXTure:EMBed	None

## Noise Receiver (Opt 028)

Select Noise Receiver	SENSe:NOISe:RECeiver	NoiseReceiver
-----------------------	----------------------	---------------

## Miscellaneous

Include SC12 Sweep	Sens:Mixer:Reverse	IncludeReverseSweep
Active Window Background	Disp:Color:ABACKground	ActiveBackground
Sweep Delay	SENS:SWEep:DWELI:SDELay	SweepDelay
<b>Preference:</b> On PRESET always turn power ON	SYST:PREF:ITEM:PRESet:POWer	PresetPowerState
Send and return Instrument state file to remote computer	SYSTem:SET	None
Read the display image into remote computer	HCOPy:SDUMp:DATA?	None
Set format of display image	HCOPy:SDUMp:DATA:FORM	None
Set and return LXI status	LXI:IDEN	LXIDeviceIDState
GCA Safe mode - max power	Sens:GCS:SAFE:MLIM	SafeSweepMaximumLimit
Set Line type of the window grid	DISP:WIND:TRAC:GRAT:GRID:LTYPE	GridLineType
Save Data	MMEM:STOR:DATA	SaveData

The following are new programming commands for **PNA release 9.10** See What's New

## NFX

Sets the power sensor connector	SENS:CORR:COLL:GUID:PSEN:CONN	PowerSensorConnectorType
Sets the power sensor calkit	SENS:CORR:COLL:GUID:PSEN:CKIT	PowerSensorCalKitType
Sets power level for source power cal	SENS:CORR:COLL:GUID:PSEN:POW:LEV	PowerCalibrationLevel
Sets auto orientation state for noise tuner	SENS:NOISe:TUNer:ORlent	AutoOrientTuner
Sets LO power calibration state	SENS:CORR:COLL:NOIS:LO:PCAL:STATe	EnableLOPowerCal
Sets the source pull technique to compute DUT S-parameters	SENS:NOISe:PULL	SourcePullForSParameters
Sets the state of ENR adapter de-embedding.	SENS:CORR:COLL:NOIS:ENR:ADAP:DEEM	ForceDeEmbedENRAdapter
Sets the state of Power Sensor adapter de-embedding.	SENS:CORR:COLL:NOIS:PSEN:ADAP:DEEM	ForceDeEmbedSensorAdapter

## Help About commands

DSP Revision	SYST:CONF:REVision:DSP?
DSP FPGA	SYST:CONF:REVision:DSPFpga?
CPU Speed	SYST:CONF:REVision:CPU?
Hostname	SYST:COMM:LAN:HOSTname?

## Miscellaneous

Perform and apply Response (Normalization) cal	SENS:CORR:COLL:METHod	DoResponseCal
PulseSyncIn Trigger Polarity	Sense:Pulse:TPolarity	TriggerInPolarity
PulseSyncIn Trigger Type	Sense:Pulse:TType	TriggerInType

---

The following are new programming commands for **PNA release 9.0** See What's New

## Configure External Devices

Adds an external device to the system.	SYST:CONF:EDEV:ADD	Add (External Device)
Returns names of all configured devices	SYST:CONF:EDEV:CAT?	Items
Set driver for the external device.	SYST:CONF:EDEV:DRIV	Driver
Set type of device.	SYST:CONF:EDEV:DTYP	DeviceType
Configuration path for external device.	SYST:CONF:EDEV:IOC	IOConfiguration
Enable or disable communication with device.	SYST:CONF:EDEV:IOEN	IOEnable
Activation state of the device.	SYST:CONF:EDEV:STAT	Active
Time out value for external device.	SYST:CONF:EDEV:TOUT	TimeOut

## External Source Config

Set Dwell per Point	SYST:CONF:EDEV:SOUR:DPP	DwellPerPoint
Set Trigger Mode	SYST:CONF:EDEV:SOUR:TMOD	Trigger Mode
Set Trigger Port	SYST:CONF:EDEV:SOUR:TPORT	TriggerPort

## Power Meter As Receiver (PMAR) Config

Enable min and max freqs	SYST:CONF:EDEV:PMAR:FLIM	LimitFrequency
Set Max freq	SYST:CONF:EDEV:PMAR:FMAX	MaximumFrequency
Set Min freq	SYST:CONF:EDEV:PMAR:FMIN	MinimumFrequency
Set max number of PM readings	SYST:CONF:EDEV:PMAR:READ:COUN	ReadingsPerPoint
Set tolerance level	SYST:CONF:EDEV:PMAR:READ:NTOL	ReadingsTolerance
Select sensor	SYST:CONF:EDEV:PMAR:SENS	SensorIndex
Set Cal Factor data	SYST:CONF:EDEV:PMAR:TABL:CFAC:DATA	ReferenceCalFactor
Set Cal Factor frequencies	SYST:CONF:EDEV:PMAR:TABL:CFAC:FREQ	Frequency

Set Power loss data	SYST:CONF:EDEV:PMAR:TABL:LOSS:DATA	Loss
Set Power loss frequencies	SYST:CONF:EDEV:PMAR:TABL:LOSS:FREQ	Frequency
Enable Power loss data	SYST:CONF:EDEV:PMAR:TABL:LOSS:STAT	UsePowerLossSegments
Set reference cal factor	SYST:CONF:EDEV:PMAR:TABL:RFAC	ReferenceCalFactor

## Power Limit

Set power limit	SYST:POWEr:LIMit	Limit
Power limit ON/OFF	SYST:POWEr:LIMit:STATe	State
Power limit UI lock	SYST:POWEr:LIMit:LOCK	Lock

## Scale Coupling

Set method	DISP:WIND:TRAC:Y:COUP:METH	ScaleCouplingMethod
Enable window	DISP:WIND:TRAC:Y:COUP	ScaleCouplingState

## Display and Print Colors

Set background color	DISP:COL:BACK	Background
Set labels and grid frame colors	DISP:COL:GRAT1	ActiveLabels
Set inner lines of all grids in all windows colors	DISP:COL:GRAT2	Grid
Set Inactive window label colors	DISP:COL:ILAB	InactiveLabels
Set limit line colors	DISP:COL:LIM1	FailedTraces
Set trace data and Limit Line colors	DISP:COL:TRAC:DATA	DataAndLimits
Set data trace marker colors	DISP:COL:TRAC:MARK	Markers
Set memory trace colors	DISP:COL:TRAC:MEM	Memory
Set memory trace marker colors	DISP:COL:TRAC:MMAR	MemoryMarkers

Load a color theme	DISP:COL:LOAD	LoadTheme
Saves the current color theme.	DISP:COL:STOR	StoreTheme
Resets to the default PNA colors.	DISP:COL:RES	ResetTheme

#### IMD and IMS Limited Port Mapping

Set port map	SENS:IMD:PMAP SENS:IMS:PMAP	SetPortMap
Read Input	SENS:IMD:PMAP:INP? SENS:IMS:PMAP:INP?	DeviceInputPort
Read Output	SENS:IMD:PMAP:OUTP? SENS:IMS:PMAP:OUTP?	DeviceOutputPort

#### ECal User Char to Disk (new and modified commands)

Returns the factory defined and user-defined connectors.	SENSe:CORR:CKIT:ECAL:CHAR:CONN:CAT?	None
Initiate a User Char - optionally check module memory.	SENSe:CORR:CKIT:ECAL:CHAR:INIT	InitializeEx
Saves a new characterization to disk memory	SENSe:CORR:CKIT:ECAL:CHAR:DMMEM:SAVE	SaveToDiskMemory
Delete disk memory characterizations.	SENSe:CORR:CKIT:ECAL:DMMEMory:CLEar	None
Saves a disk memory characterization to an archive file.	SENS:CORR:CKIT:ECAL:EXPort	None
Imports the ECal characterization from the specified archive file.	SENSe:CORR:CKIT:ECAL:DMMEMory:IMPort	None
Reads the user-characterization info from ECal module or PNA disk memory.	SENS:CORR:CKIT:ECAL:KNAM:INF?	None

## Gain Compression Analysis

Enable a compression analysis trace	CALC:GCM:ANAL:ENAB	AnalysisEnable
Set CW frequency	CALC:GCM:ANAL:CWFR	AnalysisCWFreq
Set to discrete or interpolated CW frequencies	CALC:GCM:ANAL:ISD	AnalysisIsDiscreteFreq
Sets X-axis display	CALC:GCM:ANAL:XAX	AnalysisXAxis

## Miscellaneous

Red Segment on Limit Line failures.	SYST:PREF:ITEM:RTOF	RedTraceOnFail
Returns the X-axis values for the selected trace.	CALC:X?	
Saves a Cal Kit to a file.	SENS:CORR:CKIT:EXP	
Returns the list of cal kits that use the specified connector.	SENSe:CORR:COLL:GUID:CKIT:CAT?	GetCompatibleCalKits
Gain Compression Saturation level	SENS:GCS:COMP:SAT:LEV	SaturationLevel
Set Cal FOM Range Preference	SENS:CORR:PREF:CAL:RANG	FrequencyOffsetRangeForCalComputations
Return the Calset X-axis FOM frequency range	SENS:CORR:CSET:STIMulus?	StimulusValues
SMC - perform separate power calcs	SENS:CORR:COLL:SESS:SMC:PWRC:SEP	SeparatePowerCal

---

The following are new programming commands for **PNA release 8.60** See What's New

## Miscellaneous

Reads or writes the segment sweep table.	SENSe:SEGM:LIST	GetAllSegments
Optional arguments for...	Sens:Corr:Ckit:CLEar Sens:Corr:Ckit:Init	

---

The following are new programming commands for **PNA release 8.55** See What's New

Miscellaneous	SCPI	COM
Enable subpoint triggering	Sens:pulse:SUBPointTrig	SubPointTrigger
<b>Description</b>		
<b>IMD-X for Converters</b>		
<b>Create or Change a Custom (Application) Measurement</b>		
Create a custom measurement		
Change a custom measurement		
Return handle to a converter object		
<b>Configure a Mixer</b>		
Calculate Input and Output frequencies		
X-axis display		
Discard Changes		
Load a mixer setup		
Save a mixer setup		
Apply mixer settings		
Assign a source to mixer input or LO.		
Read all assigned roles		
Read the source assigned to a role.		
<b>Set Input</b>		
Input start frequency		
Input stop frequency		
Input power level		
Numerator - Input Frac.Mult		



Denominator - Input Frac.Mult

Input to Swept or fixed

Input fixed frequency

### **Set LO**

Recall a previously-configured external source. This is the only external LO configuration command.

LO freq sweep mode (fixed or swept)

LO fixed frequency

LO start frequency

LO stop frequency

LO power

LO power start

LO power stop

Numerator - LO Frac.Mult

Denominator - LO Frac.Mult

Input Greater / Less than LO

### **Set IF**

Sideband (high or low)

### **Set Output**

Sideband (high or low)

Output start frequency

Output stop frequency

Output to swept or fixed

Output fixed frequency

### **IMDx Cal**

Enable LO Power cal

The following are new programming commands for **PNA release 8.50** See What's New

### Compression Marker

Compression Marker level found.	CALC:MARKer:COMPression:LEVel	CompressionLevel
Read Compression Marker Input power	CALC:MARKer:COMPression:PIN	CompressionPin
Read Compression Marker Output power	CALC:MARKer:COMPression:POUT	CompressionPout
New Search function	CALC:MARKer:FUNCTion:SEL	SearchCompressionPoint
New Execute function	CALC:MARKer:FUNCTion:EXEC	SearchCompressionPoint

### Port Extensions

Port Ext in distance	SENS:CORR:EXT:PORT:DIST	PortDistance
Set distance units	SENS:CORR:EXT:PORT:UNIT	PortDistanceUnit
Set Media per port	SENS:CORR:EXT:PORT:MEDium	PortMedium
Set waveguide cutoff freq per port	SENS:CORR:EXT:PORT:WGCutoff	PortWGCutoffFreq
Set Velocity Factor per port	SENS:CORR:EXT:PORT:VELF	PortVelocityFactor
Couple to system Velocity Factor	SENS:CORR:EXT:PORT:SYSV	PortCoupleToSystemVelocity
Couple to system Media type	SENS:CORR:EXT:PORT:SYSM	PortCoupleToSystemMedia

### Electrical Delay

Delay in distance	CALC:CORR:EDELay:DISTance	ElecDistanceDelay
Set units for distance	CALC:CORR:EDELay:UNIT	ElecDistanceDelayUnit

## Receiver Leveling

Rx Level ON OFF	SOUR:POW:ALC:MODE:REC	State
Select the reference receiver	SOUR:POW:ALC:MODE:REC:REF	ReferenceReceiver
Set maximum iterations	SOUR:POW:ALC:MODE:REC:ITER	IterationNumber
Set tolerance	SOUR:POW:ALC:MODE:REC:TOL	Tolerance
Set offset value.	SOUR:POW:ALC:MODE:REC:OFFS	PowerOffset
Separate IFBW	SOUR:POW:ALC:MODE:REC:FAST	FastMode
Set Rx IFBW	SOUR:POW:ALC:MODE:REC:IFBW	LevelingIFBW
Safe mode ON OFF	SOUR:POW:ALC:MODE:REC:SAFE	SafeMode
Safe mode Max power	SOUR:POW:ALC:MODE:REC:SAFE:MAX	PowerMax
Safe mode Min power	SOUR:POW:ALC:MODE:REC:SAFE:MIN	PowerMin

## Phase Sweep

Enable Phase sweep	CALC:FSIM:BAL:PHAS:SWE:STAT	PhaseSwpState
Start Phase port 1	CALC:FSIM:BAL:BPOR:SWE:PHAS:STAR	BalPort1StartPhase
Start Phase port 2	CALC:FSIM:BAL:BPOR:SWE:PHAS:STAR	BalPort2StartPhase
Stop Phase port 1	CALC:FSIM:BAL:BPOR:SWE:PHAS:STOP	BalPort1StopPhase
Stop Phase port 2	CALC:FSIM:BAL:BPOR:SWE:PHAS:STOP	BalPort2StopPhase
Enable as fixture offset	CALC:FSIM:BAL:FIXT:SWE:PHAS	PhaseSwpAsFixture

## Other

ASYN argument added to Cal ACQUIRE commands	<a href="#">Learn more.</a>	Not Applicable
Returns ECal orientation.	SENS:CORR:CKIT:ECAL:ORI?	None

---

The following are new programming commands for **PNA release 8.35** [See What's New](#)

## FIFO Data Buffer

FIFO ON OFF	SYST:FIFO[:STATe]	State
Read number of data points	SYST:FIFO:DATA:COUNt?	DataCount
Read data	SYST:FIFO:DATA?	Data
Read data compact form	None	DataInCompactForm
Clear data	SYST:FIFO:DATA:CLEAr	Clear

## Other N5264A Commands

FastCW	SENS:SWE:TYPE:FACW	FastCWPointCount
Enable Point Averaging	SENS:AVER:MODE	AverageMode
Enable Point Sweep	SENS:SWE:GEN:POINTsweep	PointSweepState
Set Trace Sweep	SENS:SWE:TRIG:MODE	Trigger Mode

---

The following are new programming commands for **PNA release 8.33** See What's New

## Miscellaneous

Set display format units	Calc:Format:Unit	FormatUnit
Perform trace max	Disp:TMAX	TraceMax
Fast sweep mode	SENS:SWE:SPE	SweepSpeedMode
Launch Cal Wizard for apps (new behavior)	SYST:CORR:WIZ	LaunchCalWizard
Queries the TCP/IP port number for a TCP/IP socket connection.	SYST:COMM:TCPIP:CONT	None
Set CWFreq to Marker location	CALC:MARK:SET	SetCWFreq

Returns a list of channel numbers	SYST:CHAN:CAT?	Not new
Returns measurement numbers	SYST:MEAS:CAT?	Not new
Returns trace number	SYST:MEAS<n>:TRACe?	Not new
Returns Meas name	SYST:MEAS:NAME?	Not new
Return window number	SYST:MEAS<n>:WINDow?	Not new
Returns window numbers	SYST:WIND:CAT?	Not new
Same as calc:par:sel except takes a meas number	CALC:PAR:MNUM:[SEL]	Not new
Returns Limit line pass/fail status	CALC:LIMIT:FAIL?	Not new
Deletes the current limit line?	CALC:LIMIT:DATA:DELeTe	Not new

## Swept IMD

Create a measurement	Calc:Custom:Define	CreateCustomMeasurementEx
Set sweep type	SENS:IMD:SWE:TYPE	SweepType
Set DeltaF	SENS:IMD:DFR	DeltaFrequency
Set center freq	SENS:IMD:FREQ:FCEN	FrequencyCenter
Start for center freq sweep	SENS:IMD:FREQ:FCEN:STAR	FrequencyCenterStart
Stop for center freq sweep	SENS:IMD:FREQ:FCEN:STOP	FrequencyCenterStop
Center for center freq sweep	SENS:IMD:FREQ:FCEN:CEN	FrequencyCenterCenter
Span for center freq sweep	SENS:IMD:FREQ:FCEN:SPAN	FrequencyCenterSpan
Start for DeltaF sweep	SENS:IMD:FREQ:DFR:STAR	DeltaFrequencyStart
Stop for DeltaF sweep	SENS:IMD:FREQ:DFR:STOP	DeltaFrequencyStop
Set F1 for CW and Power sweep	SENS:IMD:FREQ:F1	F1Frequency
Set F2 for CW and Power sweep	SENS:IMD:FREQ:F2	F2Frequency
Set main tone IFBW	SENS:IMD:IFBW:MAIN	MainToneIFBandwidth

Set product tones IFBW	SENS:IMD:IFBW:IMT	IMToneIFBandwidth
Enables power coupling for F1 and F2	SENS:IMD:TPOW:COUP	CoupleTonePower
Set power level for F1 tone	SENS:IMD:TPOW:F1	TonePower
Set power level for F2 tone	SENS:IMD:TPOW:F2	TonePower
F1 start for power sweep	SENS:IMD:TPOW:F1:STAR	TonePowerStart
F1 stop for power sweep	SENS:IMD:TPOW:F1:STOP	TonePowerStop
F2 start for power sweep	SENS:IMD:TPOW:F2:STAR	TonePowerStart
F2 stop for power sweep	SENS:IMD:TPOW:F2:STOP	TonePowerStop
Read highest product allowed	None	HighestOrderProduct

#### For CTB, CSO, and XMod parameters

Normalization Mode	SENS:IMD:NORM:MODE	CompositeNormalizationMode
Normalized CSO power	SENS:IMD:CSO:NORM:POW	CompositeNormalizedCSOPower
CSO Offset	SENS:IMD:CSO:OFFS	CSOOffset
CSO Number of Distortion products	SENS:IMD:CSO:NDPR	CSONumDistortionProducts
Normalized CTB power	SENS:IMD:CTB:NORM:POW	CompositeNormalizedCTBPower
CTB and XMod Number of carriers	SENS:IMD:CTB:NCAR	CTBXMODNumCarriers
CTB Offset	SENS:IMD:CTB:OFFS	CTBOffset

## IMD Calibration

Set Cal frequencies	SENS:CORR:IMD:CAL:FREQ	CalibrationFrequencies
Max Products	SENS:CORR:IMD:MPR	MaxProduct
Set power	SENS:CORR:IMD:POW	PowerLevel
Sensor Cal Kit	SENS:CORR:IMD:SENS:CKIT	PowerSensorCalKitType
Sensor connector	SENS:CORR:IMD:SENS:CONN	PowerSensorConnectorType

The following are new programming commands for **PNA release 8.2** See What's New

## iTMSA

Set Stimulus Mode	CALC: FSIM:BAL:STIM:MOD	Mode
Set Phase Offset	CALC: FSIM:BAL:BPOR:OFFS:PHAS	BalPort1PhaseOffset BalPort2PhaseOffset
Set Phase Offset as fixture	CALC: FSIM:BAL:FIXT:OFFS:PHAS	PhaseAsFixture
Set Power Offset	CALC: FSIM:BAL:BPOR:OFFS:POW	BalPort1PowerOffset BalPort2PowerOffset
Set Power Offset as fixture	CALC: FSIM:BAL:FIXT:OFFS:POW	PowerAsFixture
Set Source power for balanced ports	SOUR:POW	TestPortPower
Returns the number of source ports.	N/A	chan.SourcePortCount
Returns the string names of source ports.	N/A	chan.SourcePortNames
Returns the port number for the specified string port name.	N/A	chan.GetPortNumber

## Uncoupled Power Sweep

Set Start power for uncoupled power sweep	SOUR:POW:PORT:START	StartPowerEx
---	---------------------	--------------

Set Start power for uncoupled power sweep	SOUR:POW:PORT:STOP	StopPowerEx
---	--------------------	-------------

## Choose FCA ports

Map PNA to DUT ports	SENS:MIX:PMAP	SetDutPorts
----------------------	---------------	-------------

Read Input port number	SENS:MIX:PMAP:INP?	DeviceInputPort
------------------------	--------------------	-----------------

Read Output port number	SENS:MIX:PMAP:OUTP?	DeviceOutputPort
-------------------------	---------------------	------------------

## LXI

Returns <b>Structured</b> status of the PNA networking configuration.	None	GetIPConfigurationStruct
---	------	--------------------------

Returns <b>string</b> status of the PNA networking configuration.	None	LANConfiguration
---	------	------------------

Resets the PNA LAN configuration.	None	LANConfigurationInitialize
-----------------------------------	------	----------------------------

Modifies settings of the PNA computer networking configuration.	None	SetIPConfiguration
---	------	--------------------

Displays the LAN Status dialog with LAN Status Indicator showing IDENTIFY.	None	LXIDeviceIDState
--	------	------------------



## Miscellaneous

Reset Preference Defaults	SYST:PREF:DEF	cap.RestoreDefaults
Returns the Measurement Class name	Sens:Class:Name?	Get_MeasurementClass
GCA - Returns number of iterations required in a SMART Sweep	CALC:GCDAT:ITER	TotalIterations

The following are new programming commands for **PNA release 8.0** See What's New

## Gain Compression Setup

Number of frequency points	SENS:GCS:SWE:FREQ:POIN	NumberOfFrequencyPoints
Number of power points	SENS:GCS:SWE:POW:POIN	NumberOfPowerPoints
Maximum number of points	None	MaximumNumberOfPoints
Total number of points	None	TotalNumberOfPoints
Acquisition mode	SENS:GCS:AMOD	AcquisitionMode
Smart tolerance	SENS:GCS:SMAR:TOL	SmartSweepTolerance
Smart Iterations	SENS:GCS:SMAR:MIT	SmartSweepMaximumIterations
Smart settling time	SENS:GCS:SMAR:STIM	SmartSweepSettlingTime
Smart show iterations	SENS:GCS:SMAR:SIT	SmartSweepShowIterations
Read compression failures	SENS:GCS:SFA?	SearchFailures
Write port map	SENS:GCS:PORTM	SetPortMap
Read Port Map (Input)	SENS:GCS:PORT	DeviceInputPort
Read Port Map (Output)	SENS:GCS:PORT	DeviceOutputPort
End of Sweep	SENS:GCS:EOS	EndOfSweepOperation
Linear input power	SENS:GCS:POW:LIN:INP:LEV	InputLinearPowerLevel
Reverse Power	SENS:GCS:POW:REV:LEV	ReverseLinearPowerLevel

Start power	SENS:GCS:POW:STAR:LEV	chan.Start Power
Stop power	SENS:GCS:POW:STOP:LEV	chan.Stop Power
Compression algorithm	SENS:GCS:COMP:ALG	CompressionAlgorithm
Compression Level	SENS:GCS:COMP:LEV	CompressionLevel
Backoff Level	SENS:GCS:COMP:BACK:LEV	CompressionBackoff
X Delta	SENS:GCS:COMP:DELT:X	CompressionDeltaX
Y Delta	SENS:GCS:COMP:DELT:Y	CompressionDeltaY
Interpolation	SENS:GCS:COMP:INT	CompressionInterpolation
Safe Sweep enable	SENS:GCS:SAFE:ENAB	SafeSweepEnable
Safe Sweep coarse	SENS:GCS:SAFE:CPAD	SafeSweepCoarsePowerAdjustment
Safe Sweep fine	SENS:GCS:SAFE:FPAD	SafeSweepFinePowerAdjustment
Safe Sweep threshold	SENS:GCS:SAFE:FTHR	SafeSweepFineThreshold
Read all GCA data	CALC:GCDData:DATA	GetRaw2DDData
Read real GCA data	CALC:GCDData:REAL	GetDataIm
Read imaginary GCA data	CALC:GCDData:IMAG	GetDataRe

## Noise Figure Setup

Create Noise figure meas	Calc:Cust:Def	CreateCustomMeasurementEx
Sets the number of impedance states to use	SENS:NOIS:IMP:COUN	ImpedanceStates
Noise averaging ON and OFF	SENS:NOIS:AVER:STAT	NoiseAverageState
Set averaging of noise receiver.	SENS:NOIS:AVER	NoiseAverageFactor
Set bandwidth of noise receiver.	SENS:NOIS:BWID	NoiseBandwidth
Set gain state of noise receiver.	SENS:NOIS:GAIN	NoiseGain
Sets noise tuner identifier	SENS:NOIS:TUN:ID	NoiseTuner

Sets the port identifier of the ECal noise tuner that is connected to the PNA Source.	SENS:NOIS:TUN:INP	NoiseTunerIn
Sets the port identifier of the ECal noise tuner that is connected to the DUT.	SENS:NOIS:TUN:OUTP	NoiseTunerOut
Set the excess noise source ON or OFF.	CONTrol:NOISe:SOURce <b>or</b> OUTPut:MANual:NOISe[:STATe]	NoiseSourceState
Set mechanical switches	SENS:PATH:CONF:ELEM	PathConfiguration
Sets the default setting for the Noise Tuner switch.	SYST:PREF:ITEM:SWIT:DEF	Port1NoiseTunerSwitchPresetsToExternal

### Noise Figure Cal

Create Noise Cal object	N/A	CreateCustomCalEx
Set Noise Calibration method	SENS:NOIS:CAL:METH	CalMethod
Noise source ENR filename	SENS:NOIS:ENR:FIL	ENRFile
Set noise source Cal Kit type	SENS:NOIS:SOUR:CKIT	NoiseSourceCalKitType
Set ambient temperature	SENS:NOIS:TEMP:AMB	AmbientTemperature
Sets noise source connector type	SENS:NOIS:SOUR:CONN	NoiseSourceConnectorType
Set Noise source temperature	SENS:CORR:TCOL:USER:VAL	NoiseSourceCold

## Noise Figure ENR File Data Management

Set ENR calibration data.	SENS:CORR:ENR:CAL:TABL:DATA	PutENRData
Read ENR calibration data.	SENS:CORR:ENR:CAL:TABL:DATA?	GetENRData
Get/set ID of ENR table.	SENS:CORR:ENR:CAL:TABL:ID:DATA	ENRID
Get/set serial number of noise source.	SENS:CORR:ENR:CAL:TABL:SERial:DATA	ENRSN
Load ENR table from file.	MMEMory:LOAD:ENR	LoadENRFile
Save ENR table to file.	MMEMory:STORe:ENR	SaveENRFile

## Custom Cal Window

Turn ON   OFF Custom Cal window.	SENS:CORR:COLL:DISP:WIND	DisplayNAWindowDuringCalAcquisition
Show NO Custom Cal windows.	SENS:CORR:COLL:DISP:WIND:AOff	DisplayOnlyCalWindowDuringCalAcquisition
Specify channel to sweep before Cal acquisition.	SENS:CORR:COLL:SWE:CHAN	AllowChannelToSweepDuringCalAcquisition
Sweep NO channel before Cal acquisition.	SENS:CORR:COLL:SWE:CHAN:AOff	SweepOnlyCalChannelDuringCalAcquisition
Preview sweep before remote Cal acquisition.	SENS:CORR:COLL:GUID:PACQuire	SetupMeasurementsForStep

## Miscellaneous

Set Trigger sweep mode	SENS:SWE:TRIG:MODE	Trigger Mode
Copy a Cal Set	SENSe:CORR:CSET:COPY	

---

## COM versus SCPI

---

There are two methods you can use to remotely control the PNA: COM and SCPI. The following topics can help you choose the method that best meets your needs:

- [Software Connection](#)
- [Physical Connection](#)
- [Programming Languages](#)

### Other Topics about COM Concepts

#### Software Connection

**COM** uses a binary protocol, allowing you to directly invoke a PNA feature. This is more efficient than SCPI. For example, the following statement calls directly into the PNA, executing the routine GetIDString.

```
PNA.GetIDString()
```

**SCPI** is a text based instrument language. To retrieve the ID string, you would send the following text string to the PNA:

```
IbWrite( "*IDN?")
```

The PNA SCPI parser would first decode this text string to determine that the user has asked for the PNA to identify itself. Then the parser would call the COM method GetIDString().

#### The Physical Connection

##### Internal Control

With either COM or SCPI, the best throughput is attained by using the PNA's internal PC to execute your test code. However, if your test code uses too much system resources (CPU cycles and/or memory), this will slow the PNA's performance.

Using the SICL I/O Libraries, you can also connect to the PNA from a program running on the PNA.

##### External Control

You can control the PNA from a remote PC using either COM or SCPI.

**COM** - (Component Object Model) can be used to access any program like the PNA (835x.exe) or library (.dll) that exposes its features using a COM compliant object model. These programs or libraries are called "servers". Programs (like your remote program on your PC) that connect to and use the features of these servers are called "clients."

With COM, the server and the client do not need to reside on the same machine. DCOM, or distributed COM, makes the location of the server transparent to the client. When you access the PNA from a remote computer, you are using DCOM. In this case, the mechanical transport is a LAN (local area network).

However, using COM can add additional complexity:

- There are some DCOM security issues that may be a problem for you. [Learn more.](#)

- Using the default interface when compiling type libraries results in code that will only run with the latest firmware. [Learn more.](#)

**SCPI** - Using a GPIB interface card in a remote computer, you can connect to the instrument using a GPIB cable. There are some constraints on the length of this cable and the number of instruments that can be daisy-chained together.

Using the Agilent SICL I/O libraries, you can connect to the instrument over a LAN connection.

(LAN or INTERNAL) You can send SCPI commands using COM with the [ScpiStringParser](#) object.

If you have legacy code written in SCPI for another network analyzer, you may be able to leverage that code to control the PNA. However, the PNA uses a different platform than previous Agilent Network Analyzers. Therefore, not all commands have a direct replacement. See the PNA [Code Translator Application](#).

## Programming Languages

You can program the PNA with either COM or SCPI using several languages. The most common include:

**Agilent VEE** - With this language you can send text based SCPI commands and also use automation. VEE 6.0 or later is recommended.

**Visual Basic** - This language has great support for automation objects and can be used to drive SCPI commands. The use of VISA drivers for your GPIB hardware interface will make the task of sending SCPI commands easier.

**C++** - This language can do it all. It is not as easy to use as the above two, but more flexible.

---

Last Modified:

24-Mar-2009    Updated

## Remotely Specifying a Source Port

---

In the 'not-too-distant past', it was a simple task to specify a PNA source port. It was either port 1 or port 2. Now, for the following reasons, it is not so simple:

- **Internal 2nd sources** are now offered on various PNA models. However, some source ports do not have a port number. One example is the second source on the PNA-X 2-port model (option 224). [Learn more about Internal Second Sources](#).
- **External sources** can now be controlled by the PNA as though they are internal sources. External sources do not have a source port number, but use String names as identifiers.
  - **For FCA ONLY:** Once configured using the [Configuration dialog](#), an external source can be selected remotely and controlled by the PNA by specifying the LOName using [SCPI](#) or [COM](#).
  - **All other uses for External sources:** The external source must be configured and selected from the [External Source](#) dialog. You can then save an [Instrument State file](#), then recall that state file remotely.
- **Multiport test sets**...choose between ports 1 through port N, where N is the number of ports on the test set. You still use a port number, but this port number refers to a logical port. The Port mapping feature maps the logical port to a physical port. [Learn more about Multiport test sets](#).
- **iTMSA (Opt 460)** When this option is present, the string names for balanced source ports are returned with the appropriate COM and SCPI commands. For example, "SE Port 1" is used to access 'Single-ended Port 1'.

### Source Port String Names

The PNA User Interface (UI) makes it easy to configure and select the sources and ports. Remotely however, string names are used now, in addition to port numbers, to specify a Source port.

**COM** - The existing COM commands specify source ports as numbers and they are still used. It is necessary to learn the port number from the string using the [GetPortNumber Method](#). Port numbers are assigned dynamically depending on whether [external sources are selected](#) and the number of ports of the PNA.

- [SourcePortNames Property](#)
- [GetPortNumber Method](#)
- [SourcePortCount Property](#).

An example:

```
dim app
set app = CreateObject("Agilentpna835x.application")
dim channel
set channel = app.Channel1
dim portnum
portnum = Channel.GetPortNumber("Src2 Out1")
app.CreateMeasurement 1, "A", portnum
```

**SCPI** - ALL of the existing SCPI commands that specify a source port are extended to also allow the source port to be specified using string names. For example, send the following command to set the power on Src2 Out1:

- [SOUR:POW 5, "Src2 Out1"](#)
- Use [Source:Cat?](#) to list the available source port string names.

---

Last Modified:

23-May-2008    Added iTMSA option

24-Apr-2008    Several edits

5-Nov-2007    MX New topic



## Shut Down or Restart the PNA Remotely

---

To Shut down, send this command:

```
Shutdown -s -t: 00
```

To Restart, send this command:

```
Shutdown -r -t: 00
```

These will wait 00 seconds before executing the command.

1. Paste the command into a Notepad file.
2. Save the file as **shutdownPNA.bat** anywhere on the PNA, preferably in the C:/Windows/system 32 directory.
3. To call the file using GPIB, issue this SCPI command:

```
Diag:batch 'shutdownPNA.bat '
```

If you have placed the file somewhere on the PNA other than the system32 directory, include the full path to the batch file; such as:

```
Diag:batch 'C:/Program Files/Agilent/Network  
analyzer/Documents/shutdownPNA.bat '
```

For variations on this command, on any Windows computer:

Click **Start**, then **Run**, then type **cmd**, then **OK**.

At the prompt, type **shutdown**.

---

Last Modified:

22-May-2009    New topic

## Frequency Converter Application Known Issues

---

To see the current list of known FCA issues, please visit <http://na.tm.agilent.com/fca/> and click the known FCA issues link.

## Frequency Converter Application

---

The Frequency Converter Application ([Option 083](#)) simplifies testing of frequency converting devices.

**Note:** [Option 082](#) allows you to make only SMC calibrations and measurements.

- Advanced calibration techniques that provide exceptional amplitude and phase accuracy.
- Simple setup using PNA models with [Internal Second Source](#).
- Control of external signal sources for use as local oscillators.
- A graphical set-up dialog box that lets you:
  - quickly set up the PNA for single or dual conversion devices.
  - calculate and choose where mixing and image products will fall.

For more information, see the following topics:

- [Using FCA](#)
- [Configure Your Mixer](#)
- [FCA Calibrations](#)
- [Configure an External LO Source](#)
- [SMC with a Booster Amp](#)
- [CharacterizeAdaptor Macro](#)
- [Measure a DUT with an Embedded LO](#)

### Examples

[How to make a VMC Measurement](#)

[How to make an SMC Measurement](#)

#### Notes:

- For a detailed understanding of FCA, see our [Mixer Measurements App Notes](#).
- Please submit FCA issues that you find, as well as enhancement requests, to [fca\\_support@agilent.com](mailto:fca_support@agilent.com). See [Known Issues with the FCA](#)
- FCA is **NOT** supported on PNA-L Models. However, [Opt 082](#) (SMC only) IS supported on PNA-L Models.
- [Copy Channels](#) does NOT work with FCA.

---

Last modified:

2-Sep-2009	Removed FCA not on mmWave
13-Feb-2008	Added note about Copy Channels
Nov 28, 2006	Added clarification to PNA-L note

## Using the Frequency Converter Application (Option 083)

---

- [What's New in FCA](#)
- [Overview](#)
- [How to Create a Measurement](#)
- [FCA Measurements Offered](#)
- [FCA Measurement Settings](#)
  - [Change a Measurement](#)
  - [SMC with Phase](#)
  - [Speed Up SMC Measurements](#)
  - [Use Nominal Incident Power](#)
  - [Select X-axis Display](#)
  - [Save Trace Data](#)
  - [Avoid Spurs](#)

**Examples** (not in this topic)

[How to make a VMC Measurement](#)

[How to make an SMC Measurement](#)

**Note:** Please submit FCA issues that you find, as well as enhancement requests, to [fca\\_support@agilent.com](mailto:fca_support@agilent.com) ([See Known FCA Issues.](#))

Not sure if your analyzer is equipped with Option 083? [Here's how to identify your analyzer.](#)

[Other Frequency Converter Application topics](#)

## What's new in FCA

### Rev A.09.20

- Save SNP data using File, [Save Data As.](#)
- [SMC with Phase](#)
- [Eliminate SC12 \(Reverse\) Sweeps](#)

### Rev A.07.5 and A.08.2

In the [Mixer Configuration dialog](#), select the PNA ports to connect to the DUT.

- SMC - select PNA ports to connect to the DUT input and output.
- VMC - select PNA port to connect to the DUT output. The DUT input must always be connected to PNA port 1 because of the need for a [reference mixer](#) on port 1.

**Rev 7.2** - Support for PNA-X and [Internal Second Source as LO](#)

**Rev 6.2** - [Option 082](#) allows you to make SMC calibrations and measurements. (VMC is NOT available.)

### Rev 6.0:

- Calibrated [Swept LO](#) measurements.
- Create any of the [Mixer measurements](#) that are offered. For example, in the past if you wanted an SC12 measurement, you first had to create an SC21 measurement, and then change it to SC12. You can also create more than one mixer measurement at the same time.
- [FCA calibrations](#) are streamlined for consistency and ease of use.
- Added [SMC Power meter and offset](#) settings.
- [Embed/De-embed networks](#) for Waveguide, in-fixture, or on-wafer measurements.
- [Characterize Adaptor Macro](#) creates S2P files from two 1-port cal Sets.
- **SMC-Forward** and **SMC-Reverse** measurements can now be performed in the same channel. Therefore, we no longer refer to them as separate measurement types.
- Previous [Instrument State files](#) that include an FCA measurement can **NOT** be recalled by Revision 6.0.
- FCA is **NOT** supported when using [External Test Set Control](#).

## Overview

The following is an overview of how to make an FCA measurement:

1. DECIDE to make either a Scalar measurement or Vector measurement. The calibration method is unique to each of these. [See a comparison of these two measurement types.](#)
2. CREATE one or more FCA Mixer measurements.
3. [Setup and CALIBRATE](#) your Scalar or Vector measurement.

### How to Create an FCA Measurement

**Note:** An FCA measurement and a non-FCA measurement can NOT reside on the same channel.

- For [PNA-X](#) and ['C' models](#): First assign a VMC or SMC measurement class to a channel. [Learn how.](#)
- **E836x and PNA-L:** From the [New Trace dialog](#), click **Application**

Then change the default measurement to one you choose by doing the following:

### How to Change an FCA Measurement

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then select parameters

#### Using a mouse with PNA Menus

1. Click **Trace/Chan**
2. then **Measure**
3. then select parameters

**Programming Commands**

### New FCA Measurement dialog box help

Select one or more Scalar Mixer or Vector Mixer measurements.

- SMC and VMC measurements MUST be made on separate channels.
- After you create a mixer measurement, you can [configure the FCA measurement](#) and make other [FCA settings](#).

### FCA Measurements Offered

Learn how to [change the FCA measurement](#).

**Important Note:** Connecting your DUT to the PNA:

**RF** and **IF** terminology is NOT used in FCA because the PNA does not know how the DUT is labeled or how it will be used. Instead, the general terms INPUT and OUTPUT are used.

- **INPUT** - The DUT port being stimulated with frequencies before conversion.
- **OUTPUT** - The DUT port outputting converted frequencies.

INPUT and OUTPUT Frequencies are specified using the [Configure Mixer dialog box](#).

### Vector Mixer/Converter Measurements

The DUT input is always connected to PNA port 1. However, the DUT output can be connected to any other PNA port. [Learn more](#).

- **VC21, VC31, or VC41 Conversion Loss/Gain (default)** - stimulus at Input, response at Output
- **S11** - stimulus and response at Input
- **S22, S33, or S44** - stimulus and response at Output
- **R1** - stimulus at Input, measures absolute power at the R1 receiver (uncorrected)
- **B, C, or D** - stimulus at Input, measures absolute power at the output receiver (uncorrected)
- Reverse conversion loss is NOT offered because of the reference mixer.

See Also: [Measure a DUT with an Embedded LO](#)

### Scalar Mixer/Converter Measurements

**Important Note:** You can connect your DUT input and output to any PNA ports. [Learn more](#).

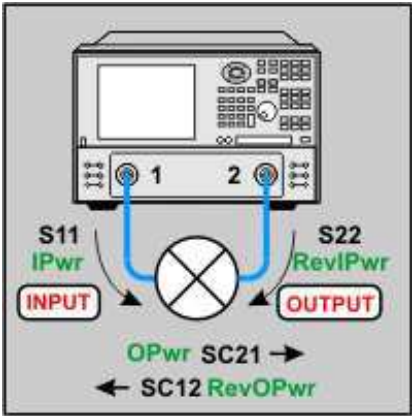
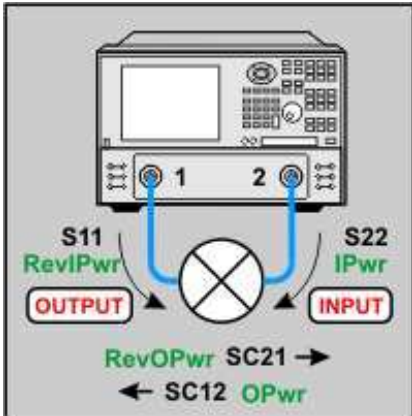
Rather than list all of the measurement possibilities, the following images and descriptions show ONLY a DUT connected to PNA ports 1 and 2 with the DUT Input and Output reversed.

Legend:

**Black** are ratioed measurements

**Green** are unratioed measurements



<p>DUT <b>Input</b> to PNA port 1 DUT <b>Output</b> to PNA port 2</p>	<p>DUT <b>Input</b> to PNA port 2 DUT <b>Output</b> to PNA port 1</p>
	
<p><b>Ratioed</b></p> <ul style="list-style-type: none"> <li>• <b>SC21</b> stimulus at Input, response at Output</li> <li>• <b>SC12</b> stimulus at Output, response at Input</li> <li>• <b>S11</b> stimulus and response at Input</li> <li>• <b>S22</b> stimulus and response at Output</li> <li>• <b>GAV</b> Maximum Available Gain (Pout/Psource)</li> </ul>	<p><b>Ratioed</b></p> <ul style="list-style-type: none"> <li>• <b>SC21</b> stimulus at Output, response at Input</li> <li>• <b>SC12</b> stimulus at Input, response at Output</li> <li>• <b>S11</b> stimulus and response at Output</li> <li>• <b>S22</b> stimulus and response at Input</li> </ul>
<p><b>Unratioed</b> Absolute test port receiver measurements.</p> <ul style="list-style-type: none"> <li>• <b>IPwr</b> (Incident Power) - stimulus and response at Input</li> <li>• <b>RevIPwr</b> (Reverse Incident Power) - stimulus and response at Output</li> <li>• <b>OPwr</b> (Output Power) - stimulus at Input, response at Output</li> <li>• <b>RevOPwr</b> (Reverse Output Power) - stimulus at Output, response at Input</li> </ul>	

See Also: [SMC with a Booster Amp](#)

### Channel / Window Selections

**Channel Number** Select the channel for the new traces.

#### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the PNA [traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See PNA [number of windows limitation](#).

## FCA Measurement Settings

Most of the FCA measurement settings in the remainder of this topic are made using the following menu selection. The choices will be slightly different depending on the active FCA measurement.

### How to select the following FCA measurement settings

1. First [create an FCA measurement](#), then...

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then select setting

#### PNA Menu using a mouse

1. Click **Response**
2. then **Measure**
3. then select setting

◀ Programming Commands ▶

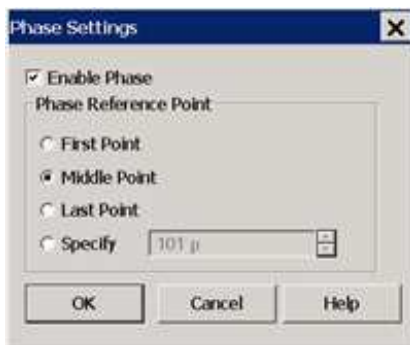
## Include Phase with SMC

Click **Response**, then **Measure**, then **Phase Settings** with an SMC measurement active.

Beginning with PNA Rev.A.09.20, you can optionally measure phase with SMC.

This capability is available only with Option 083 (NOT Option 082).

[Learn more about including Phase with SMC.](#)



## Phase Settings dialog box help

**Include Phase with SMC** Check to perform phase measurements. You can also enable phase measurements during calibration.

### Phase Reference Point

The SMC Phase measurement technique provides for a coherent phase relationship from one frequency to the next in each sweep. However, the phase measurement of the first data point is random from sweep to sweep. This initial phase offset does not impact measurements such as group delay or deviation from linear phase. However, in order to keep a phase trace from appearing random, all phase data in the sweep is normalized against a single point. This results in a stable, normalized phase trace.

For this normalization, select the measurement point that has the best signal-to-noise ratio. The phase at the selected point will always be zero. This selection applies for both the measurement and the calibration sweeps.

[Learn more about including Phase with SMC.](#)

## Speed Up SMC Measurements

Using default SMC settings, any calibrated SMC measurement requires four sweeps. However, you can reduce the number of sweeps required by selecting one or more of the following settings.

- [Apply Cal Set or Cal Type](#)
- [Use Nominal Incident Power](#)

### Notes:

- VMC measurement sweeps can NOT be reduced.
- To speed up a Swept LO measurement when using an external source for the LO, use **Hardware List (BNC)** Trigger setting . [Learn more.](#)

### Use Nominal Incident Power

Click **Response**, then **Measure**, then **Use Nominal Incident Power**

Each data sweep of a fully corrected SMC transmission measurement actually requires FOUR data sweeps. When you select **Use Nominal Incident Power**, the reference receiver (R1 or R2) does not measure incident power. Instead, the incident power is assumed to be at the level that was set with the [Source Power Calibration](#) that is done as part of every SMC measurement. The degradation in accuracy is very negligible if the input or output of your test device is well-matched.

This selection eliminates sweeps ONLY when both **Include Input Match** AND **Include Output Match** is cleared on the Cal Type dialog. [Learn more.](#)

### Select X-axis Display for FCA Measurements

Click **Response**, then **Measure**, then **Select X-axis**, then **Input**, **LO**, or **Output**

You can choose to show the frequency range of any of the swept parameters on the X-axis.

For example, the following image shows an SMC Fixed Output response with the Input frequency range on the X-

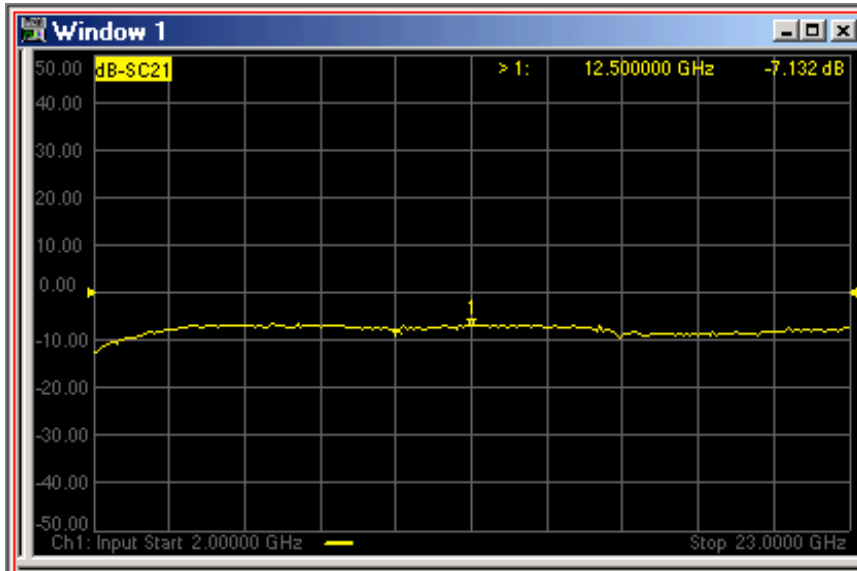
axis:

Output: 100 MHz (data trace)

Input: 2 GHz to 23 GHz (X-axis)

LO: 1.9 GHz to 22.9 GHz (not shown)

Marker annotation shows Output power at Input frequency.



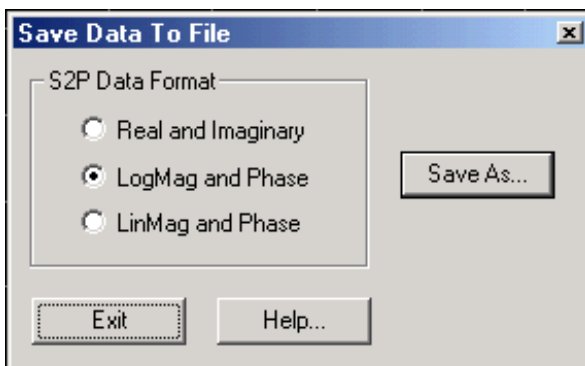
## Save Trace Data

You can save your Frequency Converter measurement data in S2P format to disk.

**Note:** Beginning with PNA rev. A.09.20, you can save \*.S2P data using the **File, [Save Data As](#)** dialog.



Beginning with PNA release 6.03, save FCA .S2P files remotely using the standard [Save SNP](#) programming commands.



## Save Data to File dialog box help

Allows you to save Frequency Converter measurement data to an S2P file. The data is saved in S2P format much like standard PNA data. [Learn more about .S2P files.](#)

**Note:** Beginning with PNA rev. A.09.20, you can save \*.S2P data using the **File, Save Data As.** dialog.

**S2P Data Format** Select the data format. This selection is independent of the PNA display.

**Save As** Click to specify a file name and location for the saved data.

**Exit** Closes the dialog box without saving the data. To save the data, you must click on the Save As button before clicking the Exit button.

### Notes:

Each record contains 1 stimulus value and 4 parameters (total of 9 values) as follows:

Stim Real(p1) Imag(p1) Real(p2) Imag(p2) Real(p3) Imag(p3) Real(p4) Imag(p4)

where **pX** is the parameter depending on measurement type:

Measurement Type	p1	p2	p3	p4
Scalar	S11	SC21 (FWD)	SC12 (REV)	S22
Vector	S11	VC21	VC12	S22
Mixer Characterization	Directivity	Source Match	Reflection Tracking	M21

- If correction is OFF, data is only saved for the active parameter. Zeros are saved for all other parameters.
- If correction is ON, data is saved for all of the parameters.

All files contain the following Header Information: Brackets [ ] contain parameters.

```
!Agilent [Instrument Model Number]: [version]
!Mixer S2P File: [Mixer Measurement Type]
!Parameters: [Parameter List]
!Calibration State: [On/Off]
```

```
!# Begin Mixer Setup
![Mixer Setup parameters listed here]
![Mixer Parameter 1]
.
.
![Mixer Parameter n]
!# End Mixer Setup

# [S2P data here]
```

## Avoid Spurs

The Avoid Spurs feature of the Frequency Converter Application attempts to prevent unwanted mixing products from appearing on the PNA screen. The Avoid Spurs feature does not significantly impact measurement speed.

**Note:** The Avoid Spurs feature is OFF by default for FCA calibrations. For highest accuracy, make measurements with the Avoid Spurs feature at the same state (ON or OFF) as was used when calibrating.

To enable Avoid Spurs:

On the [Mixer Setup](#) dialog box, check **Avoid Spurs** or

Click **Response**, then **Measure**, then **Avoid Spurs**

### Description

A spur, or spurious signal, is a term used to describe the unwanted product of two signals mixing together. When you configure the mixer setup dialog box for a desired Output, the PNA computes the frequencies of potential unwanted signals. By manipulating internal PNA hardware, these signals are avoided and do not appear on the PNA display. This means you do not need to use external filters to prevent spurious signals from appearing on the PNA display.

The time required for the PNA to compute the frequencies of unwanted spurious signals MAY be noticeable depending on the number of data points in your measurement. However, once computed, the time required for the PNA to avoid the spurs is usually insignificant.

### Limitations

The Avoid Spurs utility cannot avoid every spur. However, when there is a choice of spurs to avoid, it will avoid the largest spur.

### The Computation of Avoided Spurs

The Avoid Spur computer avoids the following spurs:

- LO, and its interaction with internal PNA components, and 16 of its harmonics.
- Input frequencies and 16 of its harmonics.
- Undesired Image frequencies (Sum or Difference) and 16 of its harmonics.

---

Last modified:

25-Mar-2010    Add SMC with phase, Data Save As, GAV

5-Feb-2010    Add Speed SMC

3-Sep-2008    Removed legacy content

1-May-2008    Updated for selectable ports

5-Oct-2007    Added link to embedded LO

## Frequency Converter Application (Option 083) Calibrations

---

Frequency Converter Application (Option 083) offers two advanced calibration choices for mixer or converter measurements that provide exceptional amplitude and phase accuracy.

**Note:** [Option 082](#) allows you to make only SMC calibrations and measurements.

- [Comparison of Scalar and Vector Mixer Cals](#)
- [SMC Setup and Overview](#)
  - [SMC with Phase](#)
- [VMC Setup and Overview](#)
- [FCA Calibration Wizard](#)
- [How to Perform an FCA Calibration](#)
- [Apply an FCA Cal Set and Cal Type](#)

**Examples** (not in this topic)

[How to make a VMC Measurement](#)

[How to make an SMC Measurement](#)

Not sure if your PNA is equipped with Option 083? [Here's how to identify your analyzer.](#)

To learn more about the FCA capability and improving FCA measurement accuracy, see [FCA App notes](#).

Please submit FCA issues that you find, as well as enhancement requests, to [fca\\_support@agilent.com](mailto:fca_support@agilent.com) ([See Known FCA Issues.](#))

[Other Frequency Converter Application topics](#)

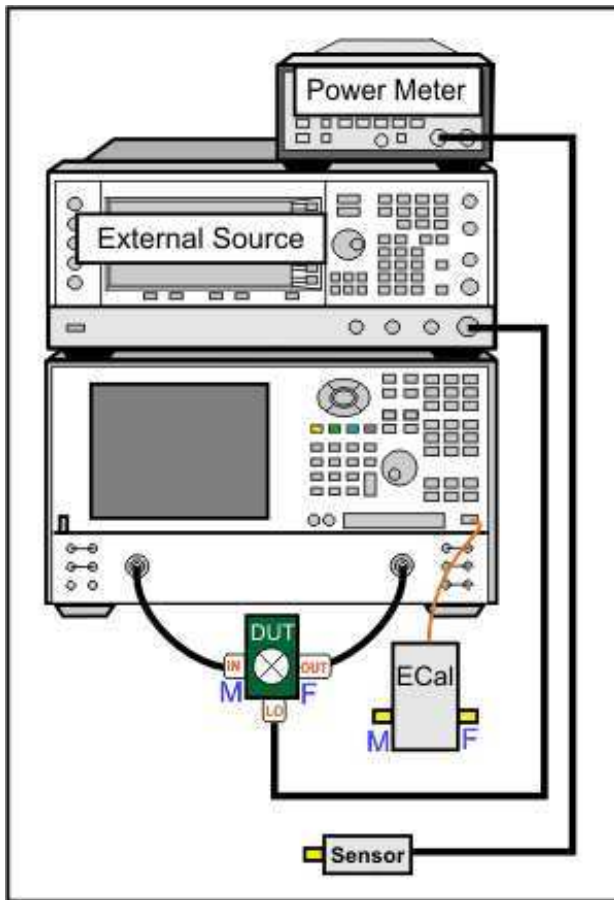
## Comparison of Scalar and Vector Mixer Cals

	Scalar Mixer Calibration	Vector Mixer Calibration
<b>Overview</b>	<p>Provides highest Scalar accuracy for measurements of conversion loss/gain.</p> <p><a href="#">Optionally measures phase.</a></p> <p>Combines SOLT and power-meter calibration.</p> <p>Simpler setup than Vector Mixer Calibration.</p>	<p>Provides unparalleled accuracy for measurements of relative phase and absolute group delay.</p> <p>Uses combination of SOLT standards and a reciprocal mixer/filter pair during calibration.</p> <p>More complicated setup and calibration procedure than Scalar Mixer Calibration.</p> <p>After calibration, both reciprocal and non-reciprocal mixers and converters can easily be measured.</p>
<b>Measurements Offered</b>	<p>Both forward and reverse directions.</p> <p>DUT can be connected to any PNA ports.</p>	<p>Amplitude response VC<sub>21</sub></p> <p>Phase response</p> <p>Group delay</p> <p>DUT input must be connected to PNA port 1.</p> <p>DUT output can be connected to any other PNA port.</p>
<b>Equipment Required</b>	Power meter and sensor	<p>Calibration mixer/filter combination (must be reciprocal <math>S_{21} = S_{12}</math>.)</p> <p>Reference mixer</p> <p>External source</p>
	<p><b>Common equipment for both SMC and VMC</b></p> <ul style="list-style-type: none"> <li>Mechanical cal kit or ECal module</li> </ul>	

[See Comparison of Mixer Characterization using New Vector Characterization Techniques.](#)

## SMC Calibration Setup and Overview





**Notes:**

- DUT can be connected to any PNA ports. [Learn more.](#)
- When using a PNA-L or PNA-X with Internal Second Source, the external source is NOT necessary. [Learn which PNA ports can be used for the LO.](#)
- Connect **External Source** and **Power Meter** to the PNA GPIB Controller port.
- See the rear panels of the [PNA-X](#) or [PNA 'C'](#) models.
- Learn how to [Configure an External LO Source](#)

**How to configure two power sensors** to cover the SMC measurement frequency range.

Using a dual channel power meter, with both sensors connected:

1. At the SMC [Select DUT Connectors](#) dialog, click **Source Cal Settings**
2. At [Source Calibration Settings](#) dialog, click **Power Meter Config**
3. At [Power Meter Settings](#) dialog, click **Sensors**
4. At [Power Sensor Settings](#) dialog, clear the **"Use this sensor only..."** checkbox for both sensors.
5. Then enter the **Min** and **Max Frequencies** for both sensors.

During the SMC Cal, you will be prompted to connect each sensor at the appropriate time.

### Overview of the Scalar Mixer Calibration.

The Calibration Wizard guides you through this process.

1. Connect a power meter / sensor to PNA Port 1. At each step of the input and output frequency, the PNA measures:
  - input match of the power sensor
  - source power of the PNA
2. Perform two [2-port SOLT calibrations](#): one over the INPUT frequencies and one over the OUTPUT frequencies of the DUT. (If your DUT is a linear device, the calibration uses only the INPUT frequency range.) Use either a mechanical calibration kit or an ECal module.

### SMC with Phase

Beginning with PNA Rev.A.09.20, you can optionally measure phase with SMC. This feature is available ONLY with Opt 083 on a PNA-X.

A Reference Mixer is NOT required as it is with VMC. Instead, during calibration the phase delay through a characterized Cal Mixer is measured and compared to the known delay. The difference between the measurement and the known delay is used to correct all subsequent SMC phase measurements.

### Comparison with VMC measurements

This feature is easier to make than VMC because a reference mixer is not required. Also the SMC calibration mixer is only required to have a known delay value, although an S2P characterization file provides more accurate results. When measuring converters with an embedded LO, SMC with phase can provide results that are as stable and free from sweep-to-sweep jitter as VMC.

### Characterizing the Calibration Mixer

Using an S2P file, which characterizes the delay across the entire frequency range, is more accurate than using a single known delay value. In either case, the group delay of the Cal Mixer must be known before starting the

Calibration Wizard.

The following are two methods to characterize the Calibration mixer:

1. Use the Mixer Characterization Wizard. This method requires that the Cal Mixer have the same requirements as the VMC Cal Mixer. [Learn more](#).
2. In a calibrated VMC channel, measure the group delay of the calibration mixer, then save to an S2P file. However, a characterized mixer is required to calibrate the VMC channel. You could use the same characterized mixer and S2P file for the SMC with phase measurement.
  - Click **Response**, then **Cal**, then **Mixer Characterization Wizard**

#### See Also

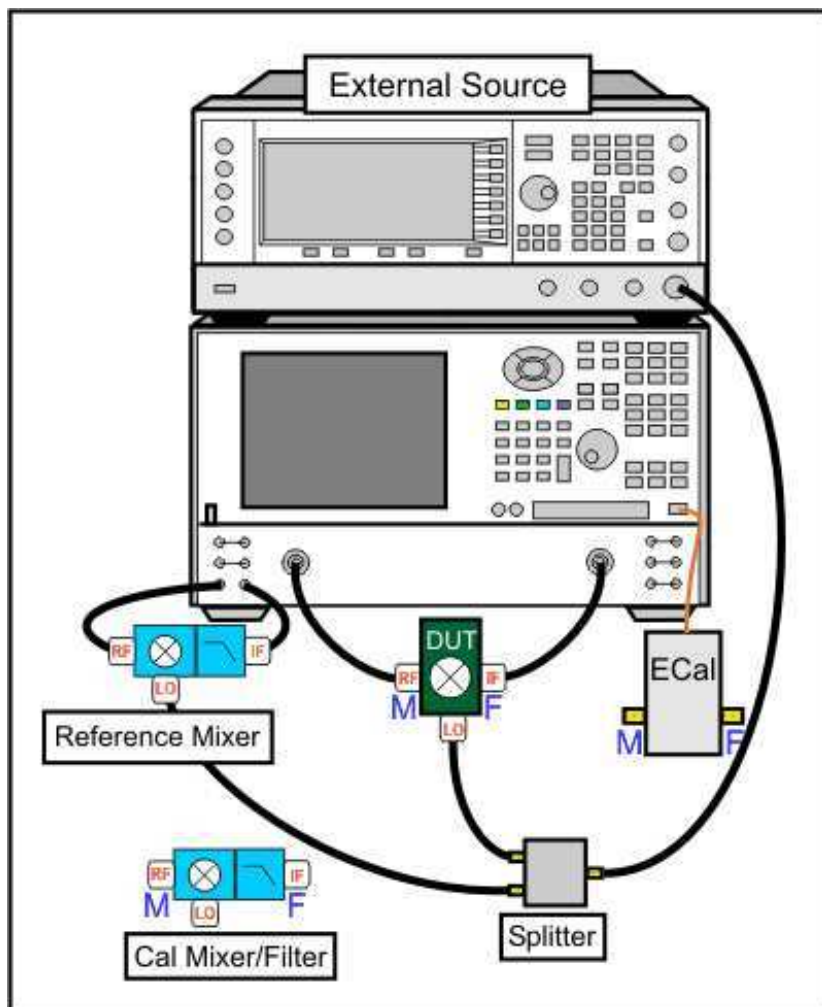
[Phase Settings dialog](#)

[Calibration Setup dialog](#)

[Measure Cal Mixer](#)

---

## VMC Calibration Setup and Overview



DUT Input (RF) must be connected to PNA port 1.

DUT Output (IF) can be connected to any other PNA port. [Learn more.](#)

**Notes:**

- When using a PNA-X with Internal Second Source, the external source is NOT necessary.
- [See note regarding LO power out both second source ports](#)
- [Learn which PNA ports can be used for the LO.](#)
- [Measure a DUT with an Embedded LO](#)

**Reference mixer** provides a phase reference for the measurements. The reference mixer is connected in the reference receiver path of the network analyzer, between the source out and receiver R1 in ports, as shown below.

The reference mixer is considered part of the test system setup like the test cables. It remains in place during the entire calibration and measurement process. The reference mixer is switched in and out of the measurement path by the PNA as needed. [See how to manually switch the reference mixer.](#)

The reference mixer does not need to be reciprocal and does not have to match the calibration mixer or the mixer-under-test in performance. The only requirement of the reference mixer is that it cover the same frequency range

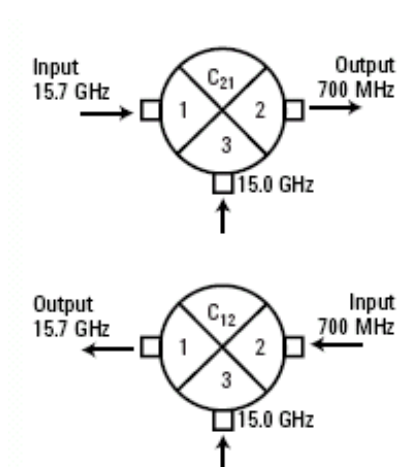
as the mixer under- test. In general, it is valuable to select a reference mixer that can be used with a variety of different setups. For example, a broadband mixer can be used in place of several narrow-band alternatives.

A low pass filter on the output of the reference mixer can be used to suppress the LO leakage signal that comes out of the reference mixer output. It is not strictly needed, but ensures that the PNA will not have any source unlock or unlevel errors due to the LO leakage.

- Connect the Reference Mixer INPUT to PNA **Ref 1 Source out**
- Connect the Reference Mixer OUTPUT to PNA **Rcvr R1 In**

**Calibration mixer/filter** is characterized either before or during a VMC calibration. It is used during the VMC calibration as the THRU standard. The calibration mixer/filter combination must meet the following requirements:

- The mixer must be reciprocal over the frequency range of the mixer under test. This means that it has the same magnitude and phase response in the up-converting and down-converting directions ( $C_{21} = C_{12}$ ) as shown in the following diagram.



- If the Input and Output frequency ranges are overlapping, the mixer must have Input to Output Isolation greater than 10 dB more than the conversion loss in the overlapping range.
- The filter must reject the undesired mixing product, and pass the desired mixing product, at the output of the cal mixer. This requirement can be made easier by characterizing the mixer/filter as a downconverter. [Learn more.](#)

#### **Important note: Orientation of Reference mixer and Calibration mixer/filter**

The reference mixer is ALWAYS connected in the same orientation as the DUT, since the output frequency of the reference mixer has to match that of the DUT. The same applies to the calibration mixer/filter if it is characterized as part of a full VMC cal.

If you characterize the calibration mixer/filter separately, you can characterize it as either an upconverter or downconverter. [Learn more.](#)

#### **LO Source(s)**

**Note:** When using a PNA-X with Internal Second Source, the external source is NOT necessary.

- [See note regarding LO power out both second source ports](#)
- [Learn which PNA ports can be used for the LO.](#)

Connect **External Sources** to the PNA GPIB Controller port.

See the rear panels of the [PNA-X](#) or [PNA 'C'](#) models.

Learn how to [Configure an External LO Source](#)

## Overview of the Vector Mixer Calibration

The Calibration Wizard guides you through this process. The first three steps characterize the calibration mixer that is used as the THRU standard during the calibration process.

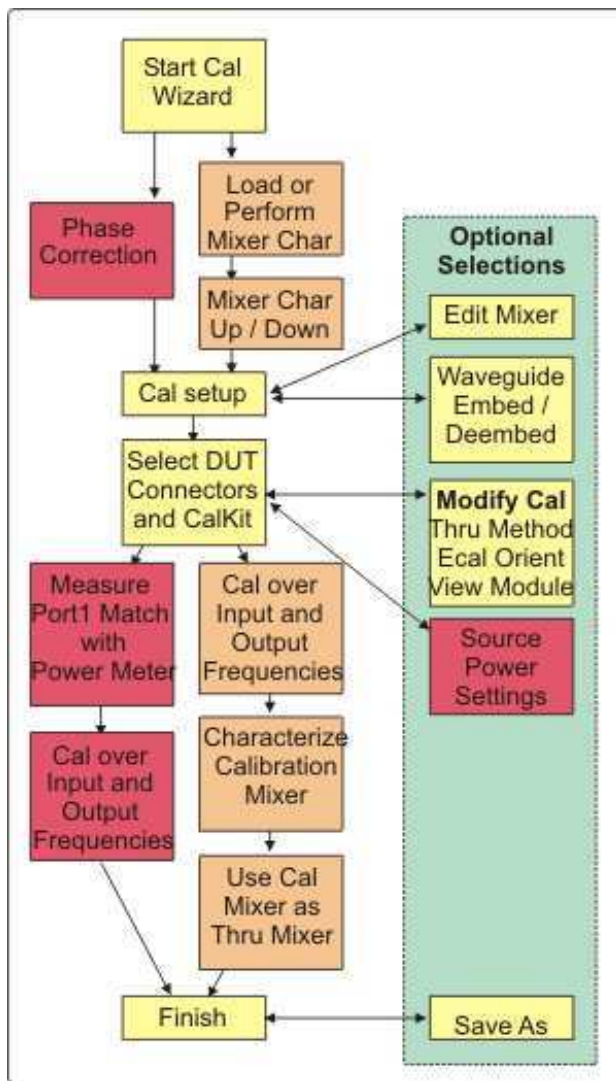
1. Perform a [2-port SOLT calibration](#) over the INPUT frequency range of the DUT, and another [2-port SOLT calibration](#) over the OUTPUT frequency range. Use either a mechanical calibration kit or an ECal module.
2. Characterize the input and output match of the [calibration mixer/filter combination](#) with the external LO connected and the output terminated with an open, short, and load. [Learn how to connect the calibration mixer/filter](#). Once characterized, an S2P file is saved and can be recalled for use in subsequent VMC calibrations using the same stimulus settings.
3. Connect the reference mixer between the Source Out and Rcvr R1 front-panel connectors. Connect the output port of the calibration mixer/filter combination to PNA Port 2 (or at the end of the cable attached to the port).
4. Measure the calibration mixer/filter combination as the THRU calibration standard.
5. The PNA calculates the error terms necessary to make corrected phase measurements of your mixer/converter under test.

To learn more about VMC capability and improving measurement accuracy, see [www.Agilent.com](http://www.Agilent.com) and search for App notes (AN 1408-1) and (AN1408-3).

## The FCA Calibration Wizard

The following dialog boxes are presented during SMC, VMC, and [Mixer Characterization](#) (used in VMC).

Click a box to learn about that step.



**Note:** In the above diagram and following procedure:

- **yellow** - steps that are common to both calibration methods.
- **tan** - VMC only steps.
- **red** - SMC only steps

## How to Perform an SMC, VMC, or Mixer Characterization Calibration

1. [Create an FCA measurement](#), then...

### Using front-panel HARDKEY [softkey] buttons

1. Press **RESPONSE**
2. then **[Cal Wizard]**

### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal Wizard**

## Programming Commands



### Calibration Setup dialog box help

Allows you to review and change the settings for your FCA calibration.

**Edit Mixer** Click to display the [Configure Mixer](#) dialog box.

**Waveguide/In-fixture/On-Wafer Setup** Invokes the following **Setup** dialog box.

**Separate power cal for input and output (do not use thru) SMC ONLY.** Check if a Thru standard is NOT available. During the [power cal](#), you will be prompted to connect the power sensor to the Input, then the Output port.





## Calibration (Phase) Setup dialog box help

Scroll up to [learn more about Including Phase with SMC](#).

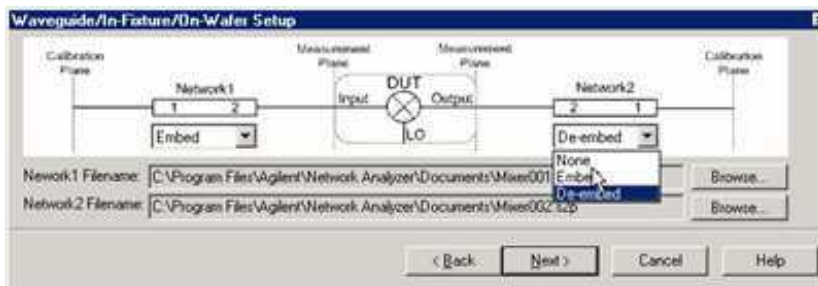
**Enable Phase Correction** Check to include the measurement of Phase with SMC. [See how to make phase settings](#).

### Phase Correction

Select the method by which the known delay of the calibration mixer is determined.

**Perform calibration using mixer with known delay** Enter the delay in seconds of the calibration mixer. This value can be obtained by measuring the group delay of the mixer from input to output.

**Perform Calibration using characterized mixer** More accurate method. The delay of the calibration mixer has been measured and saved to an S2P file. Click Browse then navigate to the S2P file. This can be the same calibration mixer used with VMC measurements.



## Waveguide/In-fixture/On-Wafer Setup dialog box help

This dialog box appears ONLY if you checked the **Waveguide/In-fixture/On-Wafer Setup** box in the previous [Cal Setup](#) dialog.

Allows you to embed or de-embed circuit networks on the input and output of your mixer under test.

For Network1 (Input) and Network2 (Output) select **Embed**, **De-embed**, or **None**.

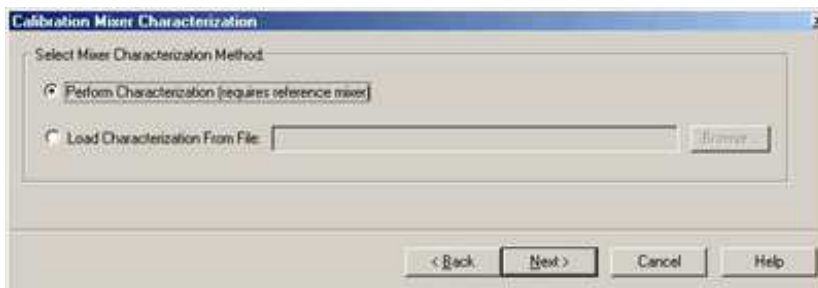
**Browse** Click to navigate to the .S2P file that models the network to embed or de-embed.

### To Embed or De-embed

- When you have a 2 port network that needs to be connected between the Cal reference plane and the DUT during the measurement, but it is NOT present during the calibration, then that network has to be **De-Embedded** from the port in question during the calibration. In other words, De-Embedding in FCA calibration extends the calibration reference plane to include the two port network.
- When you have a 2 port network that is included as part of the calibration reference plane but has to be disconnected during the measurement, then that 2-port network has to be **Embedded** for the port in question during the calibration. In other words, Embedding in FCA calibration retracts the calibration reference plane to exclude the two port network during the measurement.

## Notes

- [Characterize Adaptor Macro](#) can be used to create the S2P file.
- The S2P file for Network1 (on the input of the mixer), must cover the Input frequency range.  
The S2P file for Network2 (on the output of the mixer), must cover the Output frequency range.
- The frequency range of the S2P file must be the same, or larger than, the frequency range of the FCA measurement. If more frequencies are included in the file, and the data points do not exactly match those of the measurement, interpolation will be performed.
- As in the image on the dialog (above), in all cases:
  - Port 1 of each network is assumed to be connected to the PNA.
  - Port 2 of each network is assumed to be connected to the DUT.



## Calibration Mixer Characterization dialog box help

### VMC and Mixer Characterization ONLY

**What is Calibration Mixer Characterization?** For a brief explanation, see [Calibration Mixer](#).

### Select Mixer Characterization Method

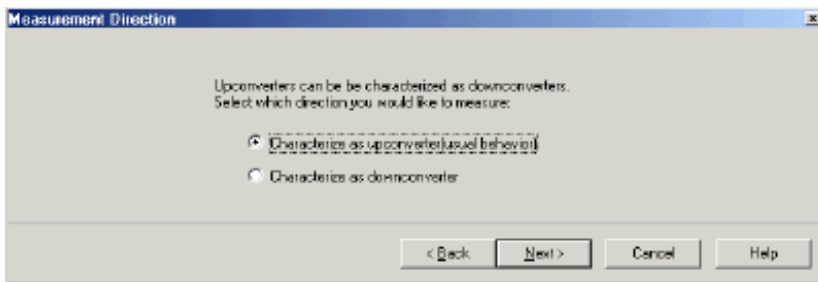
**Perform Characterization (requires a reference mixer)** Performs a Mixer characterization in addition to the VMC calibration. The mixer characterization file will be saved at the end for use in subsequent VMC calibrations. Choose this selection if you do NOT already have a mixer characterization file to load.

**Load characterization from file** Loads an S2P calibration mixer characterization file. Click **Browse** to locate the file.

- The frequency range of the S2P file MUST be the same, or larger than, the frequency range of the FCA measurement. If the S2P file frequency range is larger, or the data points do not exactly match those of the measurement, interpolation will be performed.
- The VMC calibration requires that the calibration mixer be connected in the same orientation as that in which it was characterized. The direction in which it was characterized is not part of the file that is recalled. You have to remember and connect it appropriately.

"Invalid Mixer Characterization File" is displayed if the frequency range of the S2P file is smaller than those of the measurement.

**Note:** A Mixer Characterization Cal can be performed separately. [Learn how.](#)



### Measurement Direction dialog box help

#### VMC and Mixer Characterization ONLY

This dialog box appears ONLY if your settings in the [Mixer Setup](#) dialog box indicate that your DUT is being tested as an upconverter (input < output). It allows you to characterize the [Calibration Mixer / Filter](#) as a downconverter (input > output) or an upconverter.

The following example shows why you would choose to characterize the calibration mixer as a downconverter. Consider a DUT being used as an upconverter. The input frequency is 70 MHz, the LO is 20 GHz, and the selected (+) output frequency is 20.07 GHz. If we chose (-) in the [mixer setup dialog](#), the output frequency would be 19.93 GHz.

- **Characterize as upconverter** A very sharp cutoff filter is required to reject the undesired output of 19.93 GHz and pass the desired 20.07 GHz.
- **Characterize as downconverter** The input frequency is 20.07 GHz; the LO is 20 GHz. The sum (+) output is 40.07 GHz and the diff (-) output is 70 MHz. These are very easy to separate with a low-pass filter. The original frequencies are always used in the downconversion process, so be sure to choose a filter that will pass 70 MHz and reject 40.07 GHz.

[See connection diagrams.](#)



## Select DUT Connectors and Cal Kits dialog box help

Allows you to specify the connector type and Cal Kit for each DUT port.

**Port n** For each listed PNA port, specify the DUT connector type and gender, and the Cal Kit to use.

**Mixer Out Port (VMC and Mixer Characterization ONLY)** Output port of the image filter that is connected to the [calibration mixer](#). Specify the Cal Kit / standards to use for the measurement of the calibration mixer / filter combination.

**Note:** When selecting a cal kit for the Mixer Out Port, be sure that the kit has standards with connectors that can mate to the mixer output port. If you choose an ECal, the ECal must have at least one port that can mate to the mixer output port.

**Source Cal Settings (SMC ONLY)** These settings allow you change ALL SMC Source Cal and Power Meter settings. Click to invoke the [Source Cal Settings](#) dialog.

**Note:** If your DUT connectors are:

- **Waveguide** Change the system impedance to 1 ohm before performing a calibration. See [Setting System Impedance](#).
- **Not listed** (male and female) Select **Type A** as the connector type. Type A requires a calibration kit file containing the electrical properties of the standards used for calibration (see [Calibration kits](#)).
- **Unspecified** (like a packaged device) Select **Type B** as the connector type. Type B requires a calibration kit file containing the electrical properties of the standards used for calibration (see [Calibration kits](#)).

**Modify Cal** Check to invoke the [Modify Cal](#) dialog. If performing a Mixer Characterization Cal at the same time as VMC Cal, two Modify Cal dialogs will be presented, one after the other.



## Modify Frequency Cal dialog box help

For SMC and VMC calibrations - NOT for Mixer Characterization.

### Thru Calibration Options

**Thru Cal Method** For each Thru connection, choose the Thru method. [Learn more about these choices.](#)

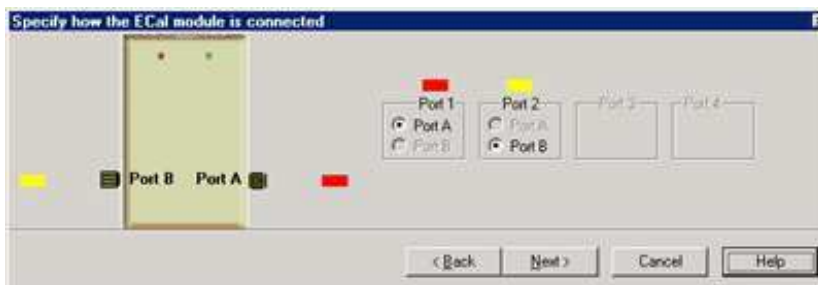
**Mod Stds** Click to invoke the [Modify Calibration Selections](#) dialog box.

The following selections are available **ONLY** if using an ECal module.

**Do orientation** When this box is checked (default) the PNA senses the ECal model and direction in which the ECal module port is connected to the PNA ports. If power to the ECal module is too low, it will appear as if there is no ECal module connected. If you use low power and are having this problem, clear this check box to provide the orientation manually.

Orientation occurs first at the middle of the frequency range that you are calibrating. If a signal is not detected, it tries again at the lowest frequency in the range. If you have an **E8361A** or **E836xB** PNA and do an ECal completely within 10 - 20 MHz OR 60 - 67 GHz, you may need to do orientation manually. There may not be sufficient power to orient the ECal module at those frequencies.

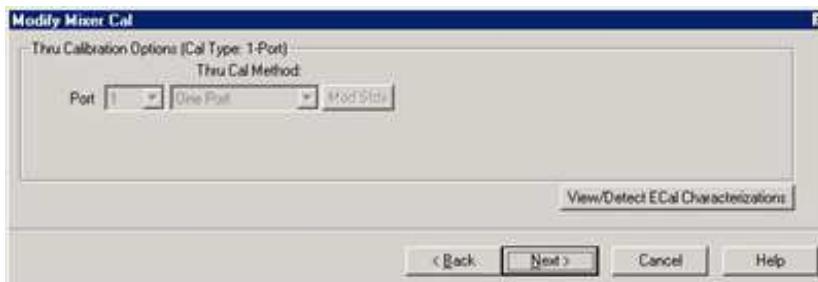
**View/Detect ECal Characterizations** Appears only if an ECal module is selected for use. Click to invoke the [View ECal Modules and Characterizations](#) dialog box. Displays a list of ECal modules that are connected to the PNA.



## Specify how the ECal module is connected dialog box help

This dialog box appears when the **Do orientation** checkbox in the previous **Modify Frequency** dialog box is cleared.

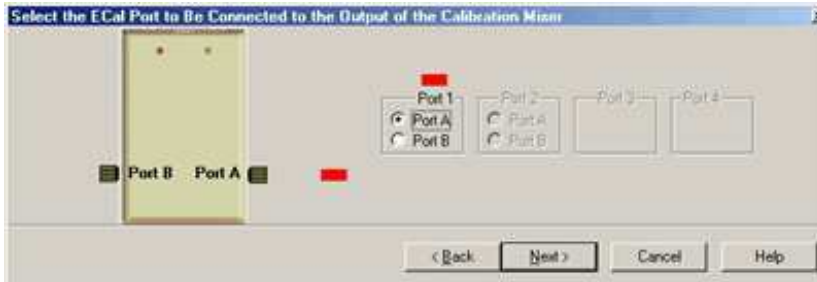
Click the ECal Port that is connected to each PNA port.



### Modify Mixer Cal dialog box help

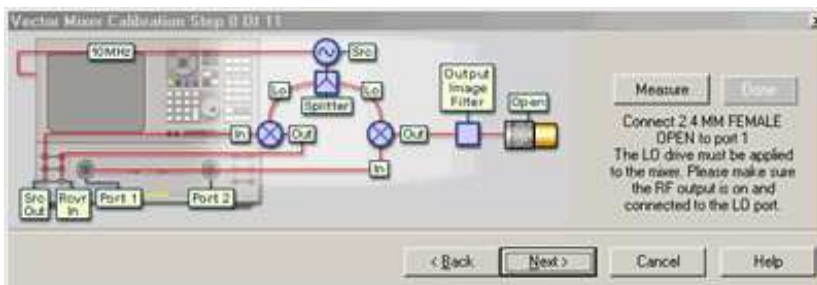
**Mixer Characterization ONLY.** The Thru standard is not measured. Therefore, the Thru Cal Method choices are not available.

**View / Detect ECal Characterizations** Available ONLY if using an ECal module. Invokes the [Select ECal Module and Characterization](#) dialog box.



### Select the ECal Port to be connected to the Output of the Calibration Mixer dialog box help

Select the ECal Port to be connected to the output of the image filter of the Calibration Mixer / Filter combination. [See connection diagram](#) of Calibration Mixer / Filter combination.



### Vector Mixer Cal dialog box help

#### VMC and Mixer Characterization

Connect the Open, Short, and Load standards to the image filter output, then click **Measure.**

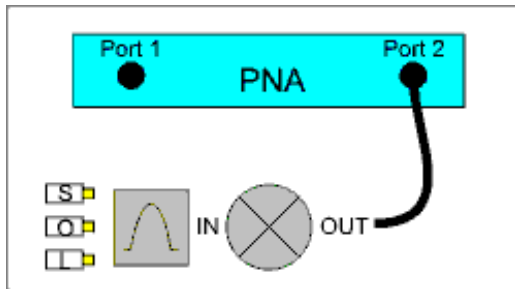
This portion of the calibration characterizes the calibration mixer.

The connection is different depending on if the calibration mixer is an upconverter being characterized as a down converter.

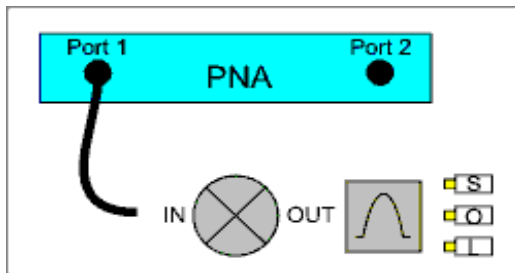
**Note:**

The following are **simplified** connection diagrams - the reference mixer and LO signals must also be connected. These images assume that the DUT output is connected to PNA port 2.

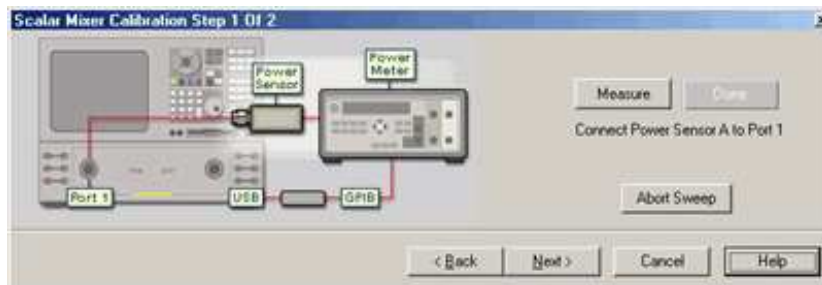
As a **Downconverter**. (The PNA automatically switches to make the S22 measurement on the device.)



As an **Upconverter**



**Done** Click to proceed to the [Calibration Complete](#) dialog. Available only after all measurements for the calibration are complete.



### Scalar Mixer Calibration - Power Cal dialog box help.

**SMC ONLY** Perform the power-meter portion of the calibration.

Connect your power sensor to the PNA port that is shown in the diagram. Then click **Measure**.

**Measure** Begins the power meter measurements and then continues to the next step.

**Done** Click to proceed to the [Calibration Complete](#) dialog. Available only after all measurements for the calibration are complete.

**Abort Sweep** Stops the power meter measurement.

**Back** Returns to the previous dialog box.

**Next** Continues to the next calibration step. Does NOT make a measurement.

#### Notes

- SMC calibration performs 10 averages at the beginning and at the end of the power cal step to ratio the difference between normal and offset R1 measurements in the calibration band of frequencies. The averaging is done to remove a reasonable amount of noise from the ratio measurement.
- From [Source Calibration dialog](#) you can use the [Power Loss Compensation Table](#) to compensate for an adapter used to connect the power meter sensor.
- See [how to configure two power sensors](#).
- When a Thru standard is NOT available, you can choose to perform a power cal at both the input and output reference planes. [Learn more](#).



### Measure THRU Mixer dialog box help

**SMC ONLY** [When Enable Phase](#) is selected. Connect the Calibration mixer and click **Measure**. Learn more about [SMC with Phase](#).





#### Measure Calibration Standards dialog box help

Prompts for standards to be measured. Connect the standard, then click **Measure**.

**Measure** Measures the mechanical standard and continue to the next calibration step.

**[ReMeasure]** Replaces Measure after standard has been measured. Allows you to remeasure a standard.

**Done** Click to proceed to the [Calibration Complete](#) dialog. Available only after all measurements for the calibration are complete.

**Back** Returns to the previous dialog box.

**Next** Does NOT make a measurement. Proceeds to the next required step.

**Cancel** Exits the Calibration Wizard.



#### Save Mixer Characterization dialog box help

##### VMC ONLY

Allows you to save the characterization data of your [calibration mixer](#). When performing another VMC calibration using the same calibration mixer, this [S2P file can then be recalled](#).

**Browse** Navigate to the location where you want to save the characterization data of your calibration mixer. Either use the default file name or enter a custom file name.

**Next** Saves the mixer characterization file and continues with the next step in the full system calibration routine.

**Finish** Replaces **Next** if you are only characterizing the calibration mixer instead of performing a full system calibration. Saves the mixer characterization file and exits the mixer characterization routine.



### Calibration Completed dialog box help

**Finish** Save to the channel's calibration register.

**Save As User Cal Set** Invokes the [Save as User Cal Set dialog box](#) AND save to the channel's calibration register.

**Cancel** Calibration is NOT applied or saved.

Learn about [Calibration Registers](#).

Learn about [User Cal Sets](#)

## Create and Apply an FCA Cal Set or SMC Cal Type

You can create an FCA measurement and apply an existing Cal Set as you can with any PNA measurement. Learn about [Cal Sets](#). In addition, from a Cal Set, you can apply a specific SMC Cal Type to an existing SMC measurement.

Although the Cal Type selection is available for VMC, there is only one VMC Cal Type.

### How to apply an SMC Cal Type

1. [Create an SMC measurement](#)
2. Calibrate or apply an existing SMC Cal Set, then...

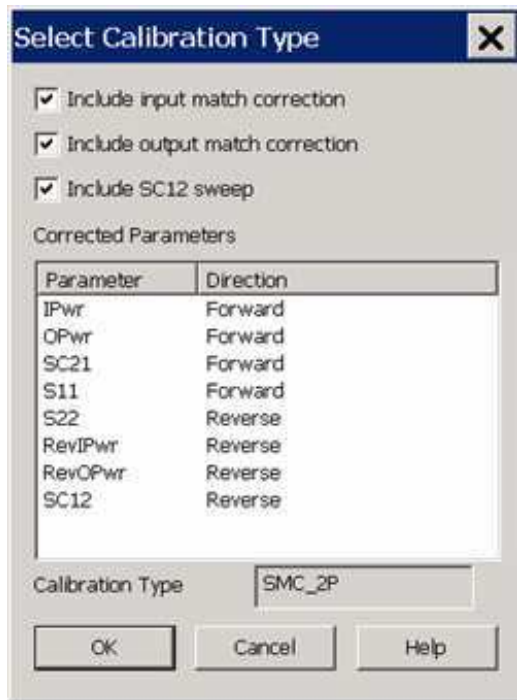
#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Manage Cals]**
3. then **[Cal Type]**

#### Using a mouse with PNA Menus

1. Click Response
2. then **Calibration**
3. then **Manage Cals**
4. then **Cal Type**

**Programming Commands**



#### Select Calibration Type dialog box help

By default, each SMC calibration requires FOUR sweeps. Clearing boxes will eliminate sweeps and speed up your SMC measurements. The difference in speed is most noticeable when making fixed input or fixed output measurements with an external LO source.

**Include input match correction** Check to perform a sweep to measure and correct for INPUT match. Clear this box if the input of your mixer is well-matched to the PNA, or if your setup does not permit a valid S11 measurement.

**Include output match correction** Check to perform a sweep to measure and correct for OUTPUT match. Clear this box if the output of your mixer is well-matched to the PNA, or if your setup does not permit a valid S22 measurement.

**Include SC12 Sweep** Check to perform a reverse sweep to measure SC12.

- When checked (default setting), a calibrated SMC measurement sweeps in both forward (SC21) and reverse (SC12) directions.
- Clear this checkbox to eliminate sweeps in the reverse direction. This means that the following measurements will NOT be corrected: SC12, RevOPwr, RevIPwr.

**Corrected Parameters** Lists the parameters that **can** be corrected given the boxes that are currently checked. These parameters may not be currently measured.

**Calibration Type** Shows the type of SMC Cal that will be applied given the boxes that are currently checked.

Learn about [Use Nominal Incident Power](#)

**How many sweeps can be eliminated?**

Setting	Parameters <a href="#">Learn about parameter abbreviations</a>	# of sweeps
ALL checked and clear <a href="#">Use Nominal Incident Power</a>	IPwr,OPwr,RevIPwr,RevOPwr,SC21,SC12,S11,S224 Total	
Perform this action...	to REMOVE these parameters...	and these sweeps
Clear "Include SC12"	Remove RevIPwr,RevOPwr,SC12	Removes 1
Clear "Include OUTPUT match"	Remove S22	Removes 1 when Nominal is checked*.
Clear "Include INPUT match"	Remove S11	Removes 1 when Nominal is checked*.
Check "Nominal Incident Power"	Remove IPwr, RevIPwr	May remove up to 2*
<b>ALL cleared and check Nominal Incident Power</b>	<b>OPwr,SC21</b>	<b>1 Total</b>
*S11 shares a sweep with IPwr and S22 shares a sweep with RevIPwr. Therefore, when <b>Include Input Match</b> or <b>Include Output Match</b> is checked, then checking Nominal incident power does nothing.		

Last modified:

- 5-Feb-2010 New Cal Type dialog
- 16-Sep-2009 Added separate power calcs
- 24-Mar-2009 Added orientation note
- 30-Oct-2008 Removed 1 GPIB port references
- 9-Sep-2008 Changed Embed or De-embed per DS
- 3-Sep-2008 Removed legacy content
- 28-Jul-2008 Added note for Mixer Output port
- 5-May-2008 Modified for selectable ports
- 13-Mar-2008 Added Embed or De-embed (per D.S.x2)
- 6-Mar-2008 Added procedure for two sensors
- 30-Aug-2007 Modified 82357A text
- 01-Jan-2007 MX Added PNA-X UI

## Configure a Mixer

---

- [How to Start the Mixer Setup dialog box](#)
- [Learning the Mixer Setup Dialog Box](#)
- [Rules for Configuring a Mixer](#)
- [Using Power Sweep for Testing Mixers](#)
- [Input > LO Example](#)
- [Configure Swept LO Measurements](#)
- [Fractional Multiplier Examples](#)

### See Also

- [How to make a VMC Measurement example](#)
- [How to make an SMC Measurement example](#)
- [Measure a DUT with an Embedded LO](#)

**Note:** Please submit FCA issues that you find, as well as enhancement requests, to [fca\\_support@agilent.com](mailto:fca_support@agilent.com) ([See Known FCA Issues.](#))

### [Other Frequency Converter Application topics](#)

#### How to start the Mixer Setup dialog box

1. [Create an FCA measurement.](#) Then...

##### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[Input, LO, or Output]**

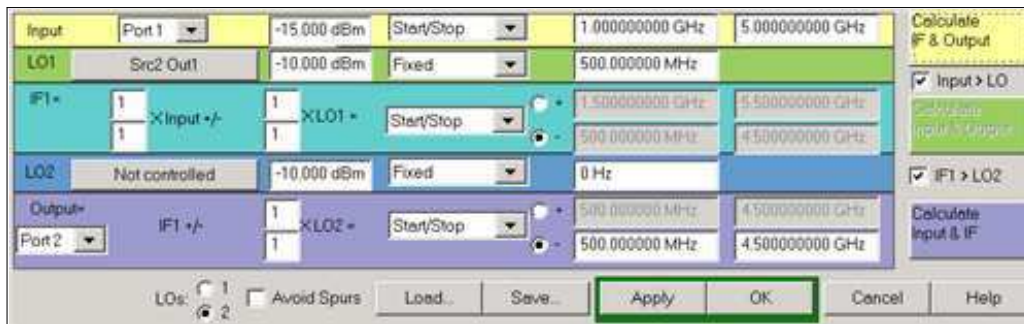
##### Using a mouse with PNA Menus

1. Click **Response**
2. then **Measure**
3. then **Configure Mixer**

[Programming Commands](#)

## Learning the Mixer Setup dialog box

Click on sections of the image to learn about a setting.



**Note:** This image shows two LO stages selected.

**Important Note:** Connecting your DUT to the PNA using FCA:

**RF** and **IF** terminology is NOT used in the FCA because the PNA does not know how the DUT is labeled or how it will be used. Instead, the general terms **INPUT** and **OUTPUT** are used to describe the following PNA behavior:

- **INPUT** - the stimulus frequencies, BEFORE conversion by your DUT.
- **OUTPUT** - the response frequencies, AFTER conversion (either UP or DOWN) by your DUT. Specify UP or DOWN conversion using the + or - symbol for each output.

See [Fractional Multiplier Examples](#) (below)

### Mixer Setup dialog box help

Throughout the dialog box, the Mixer / converter ports are color coded (Input, LO1, IF1, LO2, Output)

#### Rules for Configuring a Mixer

Red **Apply** and **OK** buttons indicate that one or more of the following settings are invalid.

1. The **INPUT** start frequency can NOT exceed the stop frequency. (The **OUTPUT** start frequency CAN exceed the stop frequency.)
2. **INPUT** or **OUTPUT** frequencies cannot be outside the range of the PNA.
3. Any combination of **INPUT** and **LO** which results in an **OUTPUT** that sweeps through zero Hz is NOT allowed.
4. The range for the numerator and denominator of a fractional multiplier is from +1 to +10. Negative values are NOT allowed.

#### Select Port

- **SMC** - select PNA ports to connect to the DUT **input** and **output**.

- VMC - select PNA port to connect to the DUT **output**. The DUT input must always be connected to PNA port 1 because of the need for a [reference mixer](#) on port 1.

**Power** Sets the power level of the input signal, and both LO signals. For added power level accuracy, use [Receiver Leveling](#).

**Frequency Format** Selects the format to specify the frequency information for each signal in your test setup. The Input, LO1, LO2, IF, and Output frequency information can be specified using start/stop or center/span formats. Only LO1, LO2, IF, and Output formats can be set to Fixed. When you select a swept LO, you can also select the information you want to [display on the X-axis](#).

**LO1 and LO2 Source Configuration Buttons** Performs the same function as the **configuration** buttons on the lower diagram. The current source, or **Not Controlled** is displayed on the button label. Click to launch the [Select Source dialog](#).

**Resulting Frequencies** Either sets or calculates the frequency values for each of the signals in your test setup. For example, if you enter the Input frequency range and press the Calculate button adjacent to the Input, the PNA will calculate and display the Output frequencies.

[Go to the Mixer Setup image](#)


### Input > LO

These check boxes remove ambiguity when using the Calculate button to determine the INPUT frequency.

<input checked="" type="checkbox"/> Input > LO	Check if the <b>INPUT</b> is <b>GREATER THAN</b> the <b>LO</b> Clear if the <b>INPUT</b> is <b>LESS THAN</b> the <b>LO</b>
<input checked="" type="checkbox"/> IF1 > LO2	Check if the <b>IF1</b> is <b>GREATER THAN</b> the <b>LO2</b> Clear if the <b>IF1</b> is <b>LESS THAN</b> the <b>LO2</b>

These boxes are only used when all 3 of the following conditions are TRUE:  
(If ALL 3 are NOT true, the PNA does not read these boxes).



1. Difference (Low) sideband  is selected for the corresponding Calculate button AND
2. Output frequency is less than the LO frequency AND
3. The Green or Blue [Calculate](#) button is used to calculate the Input frequency.

[To learn more, see this example.](#)

### Fractional Multiplier

The combination of (numerator / denominator) forms a fractional value that is multiplied by the input and LO frequency ranges (also the IF and LO2 frequency ranges for a test setup with two LOs). These values are used to Calculate the response frequency of the PNA receiver. Use the fractional multipliers to:

- replicate the action of harmonic mixers
- replicate the action of multipliers and dividers that may exist in your test setup

- tune the PNA receiver frequency to a harmonic of the mixer/converter

See [Fractional Multiplier examples](#).

[Go to the Mixer Setup image](#)

**Mixer-Product Selector** Determines whether the receivers will tune to the Sum (+) or the Difference ( - ) of the Input and LO frequencies. Click the adjacent Calculate button after your selection.

**Calculate buttons** Calculates frequency information based on your other mixer settings. The mixer port settings next to the Calculate button you press remain fixed. For example, in a 1-LO scenario, specify the Input and LO frequencies, specify + (sum), then click the **Calculate** button next to the Input. The input remains fixed and the output frequency range is calculated for you.

Calculate buttons for a Fixed frequency range do NOT appear; only for swept frequency ranges.

**Hide / Show Diagrams** Hides and displays the test setup diagram. Your measurement trace is displayed when the diagram is hidden.

**LOs** Click **1** or **2** to select the number of LO sources in your test setup. When you select 2 LOs, the IF1 frequencies are set for you. You can also measure devices with an [Embedded LO](#).

**Avoid Spurs** Check to invoke the [Avoid Spurs](#) feature.

**Load** Loads a previously-configured mixer attributes file (.mxr).

**Note:** A .mxr file includes an LO source name. However, It does NOT include the LO Source configuration. Therefore, when using a .mxr file that was created on a different PNA, the PNA will display an error if does not find the LO Source configuration using EXACTLY the same LO source name.

**Save** Saves the settings for your mixer/converter test setup to a mixer attributes file (.mxr).

#### About Mixer (\*.mxr) Files

The mixer setup files that are used with [IMDx](#) ARE compatible with the mixer setup files used with FCA. However, IMDx files contain more information than FCA files.

- When FCA files are recalled into IMDx, the PNA attempts a 'best guess' at missing information.
- When IMDx files are recalled into FCA, the extra information is ignored.

**Apply** Applies the settings for your mixer/converter test setup to the measurement. The mixer setup dialog box remains OPEN. If shaded red, [see rules](#).

**OK** Applies the settings for your mixer/converter test setup to the measurement. The mixer setup dialog box CLOSES. If shaded red, [see rules](#).

**Cancel** Closes the mixer setup dialog box and does NOT apply the settings.

**Frequency Diagram:** Provides a display of the frequency information for the signals in the test setup.

[Go to the Mixer Setup image](#)





### Select Source dialog box help

This dialog is launched when clicking the Mixer Setup **LO1** or **LO2** button.

Click one of the following to select a source for LO control:

- An existing External Source setup. Calibrate the source using a standard [Source Power Cal.](#)
- A port number to use an [internal second source](#)
- **Not controlled** the PNA will not attempt communication to control a source for the LO.
- **Manage External Sources** button to launch the [External Source Configuration](#) dialog.

#### Note: VMC measurements using a PNA-X with [Internal Second Source](#)

Source 2 is automatically configured to supply power to BOTH available ports simultaneously. This setting can NOT be changed.

In addition, [power can be uncoupled](#) to provide different power levels at each port. This feature allows power to be delivered to both the DUT LO and Reference Mixer LO without use of a splitter. [See VMC setup.](#)

## Using Power Sweep for Testing Mixers

To measure the gain compression of a mixer, you need to sweep the input power to the mixer. The input and output frequencies are fixed but offset from one another. To set Power Sweep and the input and output frequencies of the mixer under test:

1. On the mixer dialog box, set the LO frequency, identical input start and stop frequencies, and identical output start and stop frequencies. These selections create fixed input and output frequencies.
2. On the PNA menu, click **Sweep**, then **Sweep Type**. Select **Power Sweep**. Do NOT change the CW frequency on the Power Sweep dialog box. The mixer dialog box settings will not be automatically updated.

For more information, see [Conversion Compression](#).

## Input > LO Example

For the following single stage mixer:

- Output = 2 GHz
- LO = 3 GHz
- Diff (-) selected

Clicking **Calculate Input** could yield two Input frequencies:

Formula for **Diff**:

$$\text{Input} - \text{LO} = \text{Output}$$

Substitute our example values in the formula:

$$\text{Input} - 3\text{GHz} = 2\text{GHz}$$

Solving the formula can yield either:

$$\text{Input} = 5\text{GHz}$$

OR

$$\text{Input} = 1\text{GHz}$$

(Although  $1 - 3 = -2$  GHz, the analyzer displays the absolute value of the frequency.)

5.000000000 GHz	5.000000000 GHz	Calculate Output
3.000000000 GHz	<input checked="" type="checkbox"/> Input > LO	
<input type="radio"/> +	8.000000000 GHz	Calculate Input
<input checked="" type="radio"/> -	2.000000000 GHz	

**Check** - use the Input frequency (5 GHz) that is greater than LO (3 GHz)

1.000000000 GHz	1.000000000 GHz	Calculate Output
3.000000000 GHz	<input type="checkbox"/> Input > LO	
<input type="radio"/> +	4.000000000 GHz	Calculate Input
<input checked="" type="radio"/> -	2.000000000 GHz	

**Clear** - use the Input frequency (1 GHz) that is less than LO (3 GHz)

## Configure Swept LO Measurements

**Note:** With a corrected VC21Swept LO measurement, the phase data is displayed relative to the phase of the [calibration mixer](#) that was used during the VMC calibration. In addition, Group delay display format is NOT valid.

See Examples of Fixed Output Measurements

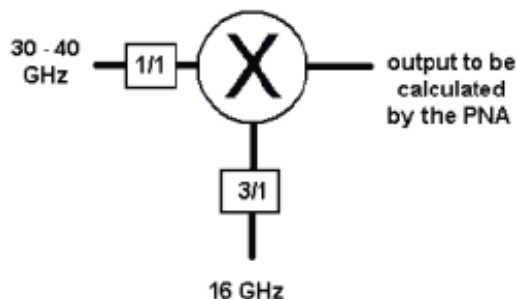
- [SMC](#)

- [VMC](#)

## Fractional Multiplier Examples

### Example 1

Use the LO fractional multiplier to replicate the action of the third-harmonic mixer so the PNA can accurately calculate the receiver frequency. The input and LO frequencies are known.



Enter these settings in the **Mixer Setup** dialog box:

- **Input Start Freq:** 30 GHz
- **Input Stop Freq:** 40 GHz
- **LO Fixed Freq:** 16 GHz
- **Mixer-Product Selector:** - (difference)
- **LOs:** 1
- **LO fractional multiplier:** 3/1
- **INPUT fractional multiplier:** 1/1

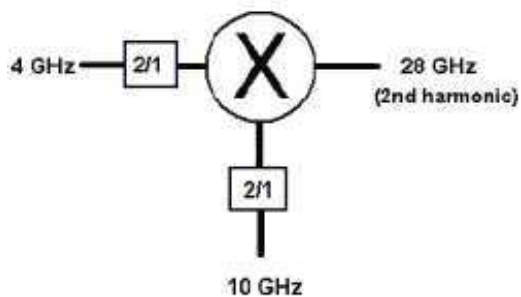
Click **Calculate Output**

Results:

- **Output Start Freq:** 18 GHz
- **Output Stop Freq:** 8 GHz

### Example 2

Use the fractional multipliers to tune the PNA receiver frequency to the second harmonic of the mixer's 14 GHz fundamental output. The input, LO, and output frequencies are known.



Enter these settings in the **Mixer Setup** dialog box:

- **Input Start Freq:** 4 GHz
- **Input Stop Freq:** 4 GHz
- **LO Fixed Freq:** 10 GHz
- **Mixer-Product Selector:** + (Sum) of the input and LO signals
- **LOs:** 1
- **INPUT fractional multiplier** = 2/1
- **LO fractional multiplier** = 2/1

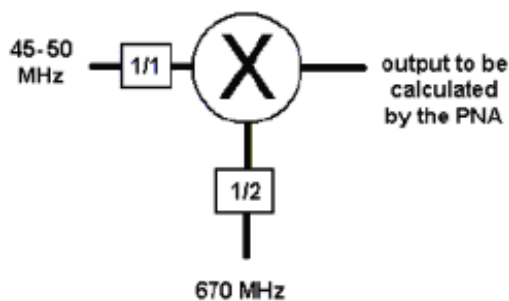
Click **Calculate Output**

Results:

- **Output Start Freq:** 28 GHz
- **Output Stop Freq:** 28 GHz

### Example 3

Use the LO fractional multiplier to replicate the action of the divide-by-two mechanism inside the mixer package. Having done this, the PNA can accurately calculate the receiver frequency. The input and LO frequencies are known.



Enter these settings in the **Mixer Setup** dialog box:

- **Input Start Freq: 45 MHz**
- **Input Stop Freq: 50 MHz**
- **LO Fixed Freq: 670 MHz**
- Mixer-Product Selector: + (Sum) of the input and LO signals
- LOs: 1
- INPUT fractional multiplier = 1/1
- LO fractional multiplier = 1/2

Click **Calculate Output**

Results:

- **Output Start Freq: 380 MHz**
- **Output Stop Freq: 385 MHz**

---

Last Modified:

- |             |  |
|-------------|--|
| 2-Sep-2009  | Added Calc buttons for fixed freq ranges |
| 9-Apr-2009  | Changed image without Show/Hide          |
| 12-Mar-2009 | Added note about mxr files               |
| 2-Oct-2008  | Changed Mixer diag image                 |
| 3-Sep-2008  | Removed legacy content                   |
| 5-May-2008  | Modified for selectable ports            |
| 20-Apr-2007 | MX Updated for internal second source    |

## Configure an External Device

---


An external device is a source or a power meter which can be configured and controlled by the PNA. In this topic:

- [How to Configure an External Device](#)
    - [External Source Configuration](#)
    - [Generic Source Commands](#)
    - [Power Meter As Receiver \(PMAR\) Configuration](#) (separate topic)
- 

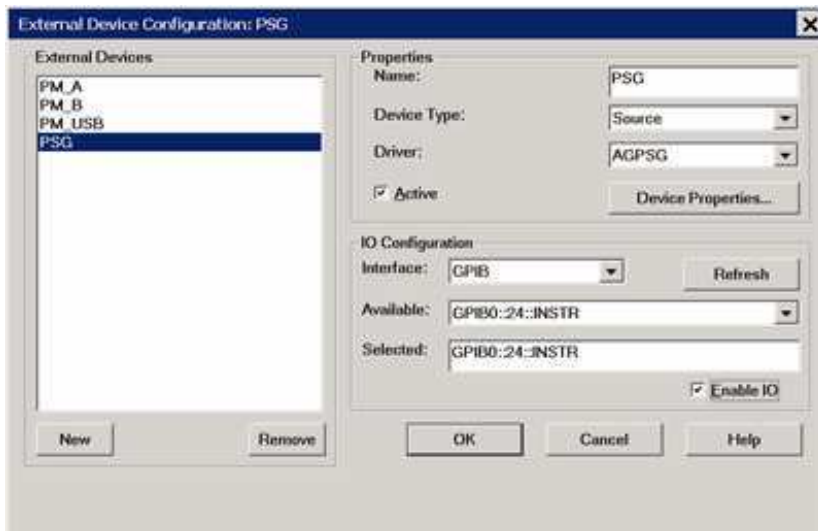
### How to Configure an External Device

[PNA Applications](#) have additional methods of launching this dialog.

Using front-panel HARDKEY [softkey] buttons	Using a mouse with PNA Menus
<ol style="list-style-type: none"><li>1. Press <b>SYSTEM</b></li><li>2. then <b>[Configure]</b></li><li>3. then <b>[External Device Configuration]</b></li></ol>	<ol style="list-style-type: none"><li>1. Click <b>Utility</b></li><li>2. then <b>System</b></li><li>3. then <b>Configure</b></li><li>4. then <b>External Device Configuration</b></li></ol>



See [Remotely Specifying a Source Port](#)



### External Device Configuration dialog box help

**Note:** To configure a single power meter for a Source Power Cal, use the [Power Meter Settings](#) dialog.

To use this dialog, your PNA must have [FOM Option 080](#). Without this option:

- You must control an external source manually. See [Synchronize an External Source](#) for help with manual source control.
- You can always use a power meter to perform a [Source Power Calibration](#).

This dialog allows you to configure an external source or power meter to be controlled by the PNA. Once you create and activate an external source or power meter from this dialog box, it becomes available from the following PNA dialog boxes as well as the softkeys and entry toolbar, as if it were an internal PNA source or receiver.

- External sources - [Power and Attenuators dialog](#); [FOM dialog](#), [New Trace / Receivers tab dialog](#)
- External power meter - New Trace/[Receivers dialog](#), [FOM dialog](#)

## Important Notes

- By default, an external device is **de-activated** when the PNA is Preset or when a Instrument State is recalled. This behavior can be changed with a [Preference setting](#) so that it remains active through a Preset or Instrument State recall.
- Device configuration is NOT saved in an Instrument State file. Therefore, recalling a state file that refers to a device that has been removed, or recalling a state file on a different PNA will result in a "Device configuration not found" error.
- Multiple configurations for the same physical device can be Active. Multiple configurations for the same PMAR device can also be Enabled. However, only one configuration for the same external source can have the [I/O Enabled](#).
- PMAR device names must begin with **PM\_n** where n is any number or character string of any length. For example, PM\_123.

## External Devices

The devices that are currently configured appear in this list. The number of devices that can be configured is limited by the specified Interface.

**New** Click to create a new source or power meter configuration. The default name is Device<n>, where <n> is the next number for 'Device'.

**Remove** Click to remove the selected device from the list.

## Properties

**Name** Enter a device name as it will appear when referring to this device in all PNA dialog boxes. Edit the name at any time. Duplicate names are not allowed.

**Note:** PMAR device names must begin with **PM\_n** where n is any number or character string of any length. For example, PM\_123.

**Device Type** Select either Source or Power Meter.

**Driver** Select the appropriate model of the device to be configured. AG is short for Agilent.

- For Power Meters, **AGPM** represents all Agilent Power Meters.
- For External Sources, choose from:
  - **AGESG (ESG)**
  - **AGPSG (PSG)**
  - **AG836XX (8360 and 8340)**
  - **AGMXG - (MXG)** Requires at least firmware A.01.44 for FOM power sweep to work correctly.
  - **AGGeneric** - For sources NOT listed. Click **Device Properties**, then **Edit Commands** to send commands to these sources. [Learn how.](#)



**Active** Check to make the device available for use in the FOM, New Trace, and Receiver Leveling dialogs. An instrument state that is saved with an Active device (checked) will include the device in the state file. Otherwise, if the Active box is cleared, the device will NOT appear in the state file.

**Enable I/O** Clear this box to disable communication with the selected device. You would do this to configure a device that is not yet connected to the PNA.

- Communication with devices is attempted when **Enable I/O** is checked, **Active** is checked, and **OK** is pressed.
- If communication with a device is lost the affected channels are put into Hold.
- When communication is attempted, devices with **Enable I/O** checked are queried for limits for frequency, power, and number of points. If there are limit problems, the PNA sends an error and the affected channels are put into Hold. These limits are enforced by the dialog box in which they are set. Resolve the reported limit problem and then restore the triggering.
- For Power Meters and Sensors, communication is also attempted when clicking the **Settings** button on the [Configure Power Sensor](#) dialog. You can not change any of the sensor settings unless **Enable I/O** and **Active** are checked and communication is possible with the sensor.

**Device Properties** Click to launch the properties (settings) dialog for the external device:

- [External Sources](#)
- [Power Meters](#)

## IO Configuration

**Interface** Select the interface that is used to connect the device to the PNA. These devices will then appear in the 'Available' field. Choose from:

- **GPIB**  - Devices connected to the System Controller GPIB port.
- **USB**  - Devices connected to the PNA USB ports
- **Aliases** - Devices that are connected to ANY interface for which you created an alias. [See Configure Alias and LAN devices.](#)
- **LAN** - Devices connected to a network using a LAN connection. The PNA must also be connected to the network.

**Note:** Devices connected to LAN must first be configured in Agilent IO libraries before they will appear on the Available list. [See Configure Alias and LAN devices.](#)

**Available** Shows a list of devices that are connected to the specified IO Interface.

**Refresh** Click to rescan the specified interface for devices.

**Selected** Enter the IO configuration or select from the available list of IO Interfaces found.


## Configure Alias and LAN Devices

Use this procedure to configure a device using a LAN interface. Also use for ANY device for which you want to

set an alias (easily-recognized) name. The alias name appear in the Available field when Aliases is selected as the Interface.

1. On the PNA, minimize the PNA application.
2. In the system tray (lower-right corner) right-click the IO icon, then click **Agilent Connection Expert**

To Add a LAN Device:

1. In ACE, click  Add Instrument
2. Select Add LAN Instrument (TCPIP0) or USB0, then click OK.
3. Click, then enter the IP address of the external source.
4. Click Test Connection to verify communication.
5. Click **OK**.

To create an Alias for a connected device:

1. In the list of connected instruments, right click the external source, then **Add VISA Alias**.
2. Enter the same PNA source name that was, or will be, used in the [External Device Configuration](#) dialog.

## External Source Configuration

### Important Notes about External Sources

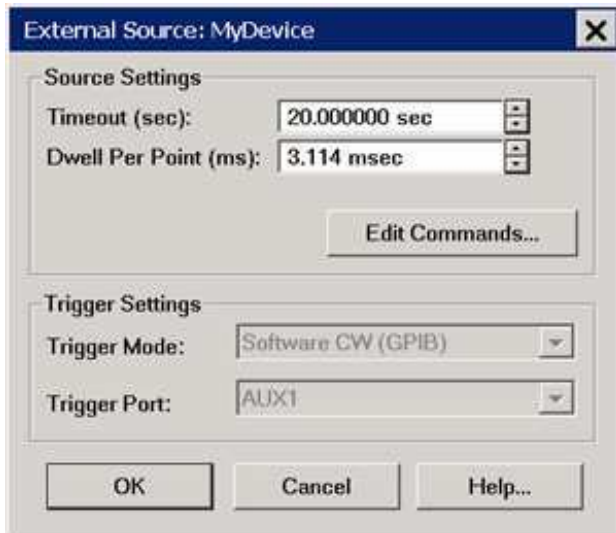
- By default, an external source is **de-activated** when the PNA is Preset or when a Instrument State is recalled. This behavior can be changed with a [Preference setting](#).
- External Agilent sources are usually limited to 1601 points with List-sweep mode. To 'work around' this limitation, divide the measurement among multiple channels. For example, to attain a sweep of 3200 points, create two channels of 1600 points. You can also use manual source control which supports Step-sweep mode. In this mode an external source can have up to 65,535 points. See [Synchronize an External Source](#) for help with manual source control.
- External sources should always share the same [10 MHz Reference signal](#) as the PNA. Connect a BNC cable from the PNA 10 MHz Ref Output to the External Source Input.
- All newly-activated sources are preset, with source power OFF. Source power must be turned ON in the [Power dialog](#). Frequency Offset must be enabled in the [FOM dialog](#).
- When daisy-chaining multiple sources in [Hardware List triggering](#), the source to receive the Trigger signal from the PNA must be the first source listed in the **Selected** column of the External Device Configuration dialog. Devices are listed in the order in which they are created. You may have to delete, then re-create a source to move it down on the list.

- The same source can NOT be used more than once in the same channel.
- All External Sources, including those used for an FCA LO, are calibrated by doing a standard [Source Power Calibration](#).
- In all applications that use this dialog to control an external source, the PNA automatically controls triggering of the external source. The PNA can be triggered internally or externally using the Internal, Manual, or External [trigger settings](#). When set to External, the trigger signal must come through a PNA rear panel connector that is not being used to trigger the external source.
- Use Meas Trig to trigger the PNA. For 'C' models, you have to manage the external trigger yourself.
- For more information, see [PNA Trigger model](#)

## ◀ Programming Commands ▶

### See Also

[Remotely Specifying a Source Port](#)



### External Source Configuration dialog box help

This dialog box is used to make external source settings.

See [Important Notes about External Sources](#)

#### Source Settings

**Timeout (sec)** Sets a time limit for the source to make contact with the PNA. If this time limit is exceeded, the PNA stops the measurement procedure and displays the following error message.

**EXECUTION ERROR;OPC QUERY TIMEOUT ERROR: FREQUENCY NOT SETTLED**

If this occurs, check the connections between your PNA and external source.

**Dwell per point (ms)** Applies a dwell in Hardware List triggering ONLY. Set the time (in milliseconds) the external source will wait before data acquisition.

**Edit Commands** Provides a method to send SCPI commands to AGGeneric (not listed) sources.

### Trigger Settings

ONLY used when the external source is stepped, as with FCA swept LO measurements.

### Trigger Mode

**Software CW (GPIB)** Slowest method.

- The external source receives each CW frequency from the PNA over GPIB.

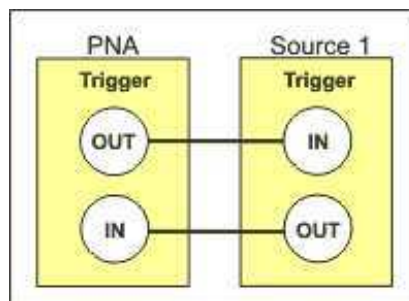
**Hardware List (BNC)** Fastest method.

- The external source receives a list of CW frequencies from the PNA, then receives BNC trigger signals as required from the PNA.
- If the number of data points used in the measurement exceeds the capability of the external source, the PNA automatically switches to Software CW (GPIB) trigger mode. This will slow the measurement significantly.
- NOT available for AGGeneric (not listed) sources.

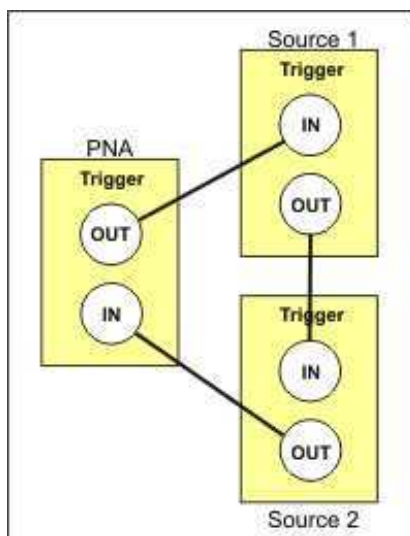
**Trigger Port** Select the PNA rear panel connector to be used for triggering. The sources must be connected as follows:

- **PNA-X models:** Connect multiple sources using the following daisy-chain, or directly using Aux1 or Aux2. [See rear panel Aux connectors](#)
- **'C' models** - Use rear-panel BNC [Trigger](#) connectors as follows:

#### 1 External Source



#### Daisy-chain 2 External Sources



**Note:** Source 1, which receives the trigger out of the PNA, must be the first source listed on the [External Devices Configuration](#) dialog box. Devices are listed in the order in which they are created. You may have to delete, then re-create a source to move it down on the list.

**Generic Source Commands** [X]

Operation complete (\*OPC):

Preset:

Set CW Frequency:

Set CW Sweep Mode:

Set Power:

Set Power State:

## Generic Source Commands dialog box help

Enter the SCPI commands that control the following functions on your AGGeneric (not listed) source. A field without a SCPI command entered will be ignored and that function will not be set.

To launch this dialog, click **Edit Commands** on the [External Source Properties](#) dialog.

### Operation Complete (\*OPC) .

**Preset** Presets the source

**Set CW Frequency** Sets CW Frequency

**Set CW Sweep Mode** Sets source sweep mode

**Set Power** Sets source power

**Set Power State** Turns power ON or OFF

### Last Modified:

30-Jul-2009	Major changes (9.0)
7-Apr-2009	Added Dwell
23-Mar-2009	Added note about MXG and E836x
3-Sep-2008	Removed legacy content
22-Aug-2008	Added list of supported..
22-Aug-2008	Added Generic source control
11-Feb-2008	Added limitation note at top
23-Jan-2008	Added Selected ordering notes
5-Nov-2007	Added links for remote selection
18-Jul-2007	Edited for FCA LO Cal changes
30-Apr-2007	MX Modified for ALL external source config.

## How to make a VMC Fixed Output Measurement

---

The following is a step-by-step example illustrating how to measure a mixer in swept LO mode using FCA Vector Mixer Calibration.

There are fewer components required for [SMC](#) as compared to VMC, and fewer measurement steps. Therefore, if you do NOT need to make relative phase measurements, SMC is an easier measurement. Also, ONLY SMC (not VMC) can measure the reverse conversion loss of the mixer.

This procedure can also be used for making **fixed** LO measurements, which is quite similar. Although the LO source is still required, the physical triggering cables that connect the PNA and External Source are not required.

### Required Equipment

- PNA-X, E8362B/C, E8363B/C, E8364B/C or E8361A/C
  - with option 083 (FCA)
  - with PNA Rev 6.03 or greater
- GPIB External Source (Agilent ESG or PSG works best). Not necessary when using PNA-X with Internal Second source.
- Reference Mixer ([see requirements](#))
- Calibration Mixer/Filter ([see requirements](#))
- Power splitter - Not necessary when using PNA-X with Internal Second source.
- ECal module with connectors that match the Input and Output connectors of the DUT. You can use adapters to make the ECal module match the DUT connectors, but first perform an [ECal user-characterization](#) with the adapters attached. ECal makes the FCA calibration much easier.
- Cables and adapters
- **Optional** GPIB Power meter and sensor (for LO power calibration)

### The example mixer

The example device is a mixer with the following characteristics:

- LO and Input Frequency Range: 2 GHz to 4.2 GHz
- Output Frequency Range: DC to 1.3 GHz

We will measure:

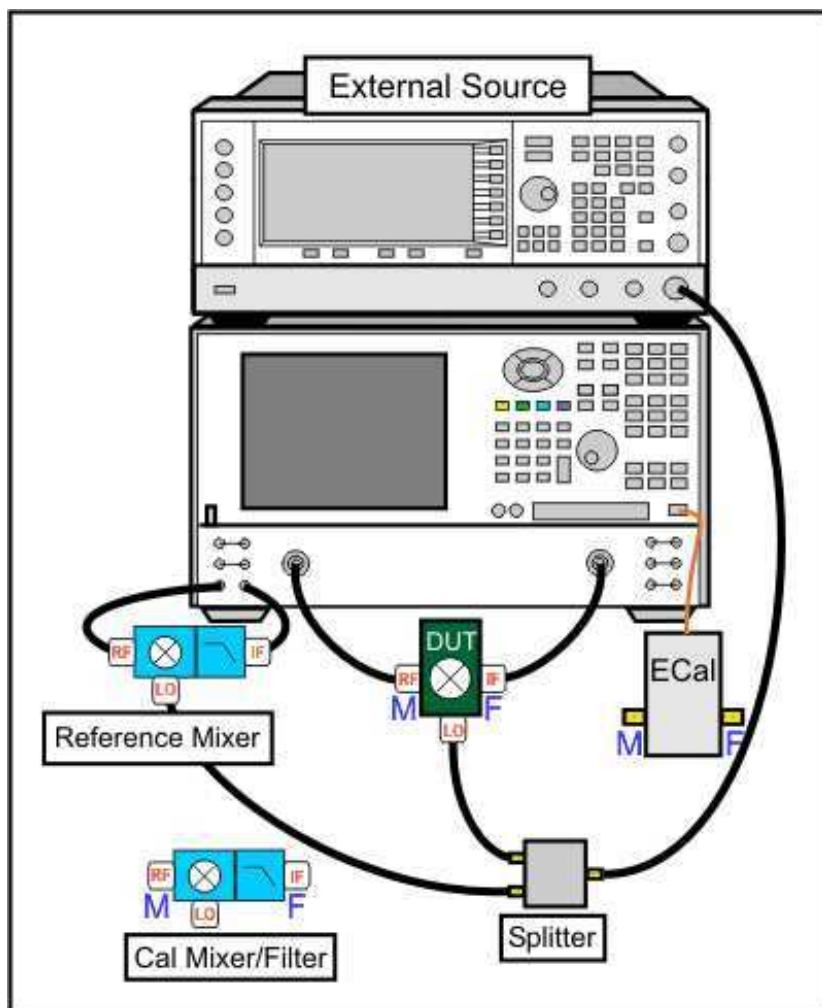
- Fwd Conversion Loss (VC21)
- Input match (S11)
- Output match (S22)

- Rev Conversion Loss is NOT possible because of the reference mixer.

## VMC Setup

Connect the devices as shown in the following diagram:

**Note:** This setup can also be used for SMC measurements, allowing you to make VMC and SMC measurements simultaneously on separate channels. The Reference Mixer is automatically switched during SMC measurements. The Cal Mixer/Filter is not used.



### Notes:

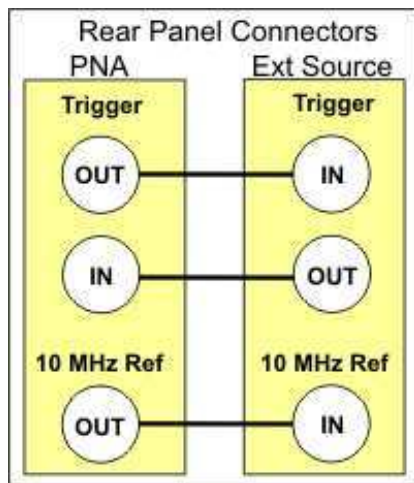
- When using a PNA-X with Internal Second Source, the external source is NOT necessary.
  - [See note regarding LO power out both second source ports](#)
  - [Learn which PNA ports can be used for the LO.](#)
- The low-pass filter on the output of the Reference Mixer is recommended, but NOT required. [Learn more.](#)
- When using 4-port models, the mixer input must be connected to PNA port 1. The mixer output can be



connected to any other PNA port.

### Make Connections on the Instrument rear panels:

1. Connect the PNA and Source using a GPIB cable. A USB to GPIB adapter can also be used if you need to control the PNA from a remote PC.
2. Using two BNC cables, connect the Source and PNA Trigger connectors as shown in the following image. This is not necessary when making fixed LO measurements, or using a PNA-X with Internal Second Source.
3. Using a BNC cable, connect the PNA **10 MHz Reference Output** to the Ext. Source **10 MHz Reference Input**.



### Create the Measurement

#### For this document:

- Front-panel hardkeys are formatted as "Press **TRACE**"
- Front-panel softkeys are formatted as "Press [**S11**]"
- Menus are formatted as "Click **System**"

1. Connect the DUT.
2. Press **PRESET** to make sure you are starting with a known state.
3. Press **MEAS** then [**Measurement Class**], then **Vector Mixer/Converter** then **OK**. At the **Confirm...** dialog, click **OK**. An S11 trace is created.
4. Press [**VC21**] to replace the S11 trace.

### Configure the Mixer settings

1. Press **FREQ** then **Input** to start the Mixer Setup dialog.
2. Enter the Mixer setup values as shown in the image below.

#### Notes:

- To provide quicker access to the Mixer Setup dialog, use the Setup softkey. [Learn how.](#)
- Rather than enter ALL of the frequency settings, you can enter the Input and the Output frequencies, then click **Calculate LO**.
- If **Input>LO** is NOT checked, the PNA assumes you want the Input < LO frequencies, and higher LO frequencies are calculated as a result.
- The LO power level setting specifies the power out of the external source (not at the DUT) unless an LO power cal is performed.
- The Avoid Spurs feature is useful for eliminating spurs in test setups with excessive LO leakage.
- When the settings are valid, the background color around the **Apply** and **OK** buttons changes from Red to Green.



#### Configure the External LO Source

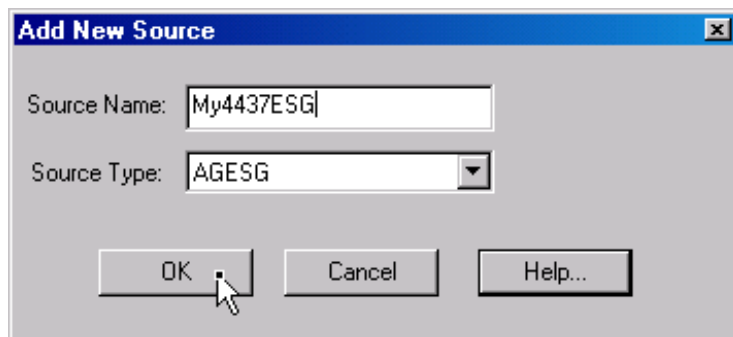
When using a PNA-X with Internal Second Source, the external source is NOT necessary.

- [See note regarding LO power out both second source ports](#)
- [Learn which PNA ports can be used for the LO.](#)

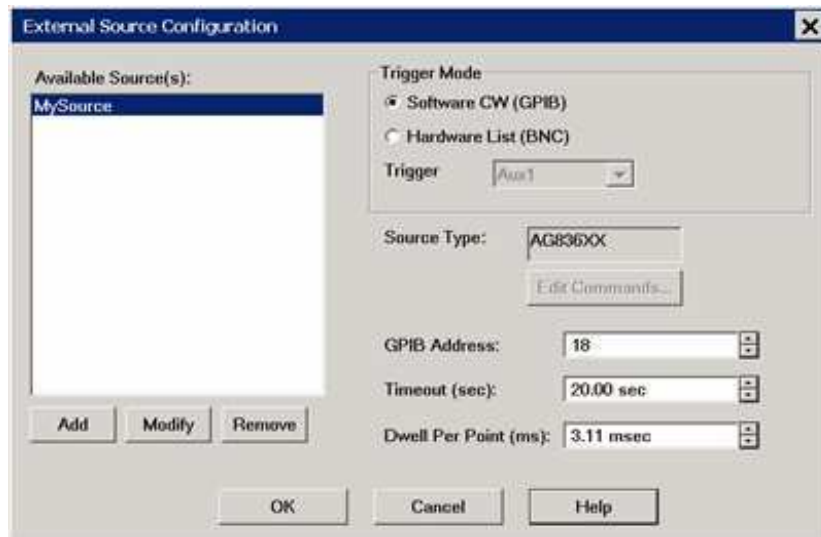
1. Click **Not Controlled** to set up the External LO source. The following dialog appears:



2. Click **Select External Sources** to perform a one-time configuration of the external source.
3. On the Select Sources dialog, click **Configure**
4. If the source that you want to control appears, select it, then skip to Step 4. Otherwise, click **Add**, to add a source.

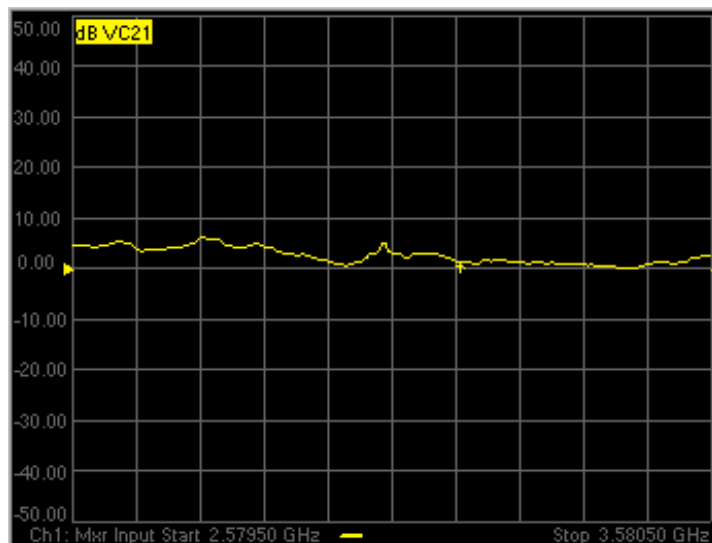


5. In the Add New Source dialog, type an identifying Source Name, such as the model of your source. In Source Type, select the model of source you are using. Then click **OK**.
6. Click **Hardware List (BNC)**, which is the fastest measurement method. This method requires the BNC Trigger cables that connect the PNA and source. If not available, **Software CW** can be used, but measurements are much slower.
7. If necessary, change the GPIB Address to match that of the source. This is NOT automatically detected.



Depending on the model of your source, this is what it looks like AFTER entering the settings.

8. On the open dialogs, make sure that the source is selected, then click **OK**. If there is a problem communicating with the source, the PNA will display an error. [See Problems?](#)
9. Click **OK** to return to the Mixer Setup dialog. The Not Controlled should now be replaced with the name of the source to use as the LO.
10. Save the mixer settings in a file so you can recall them easily. Click **Save...**, then type a descriptive filename.
11. The trace should begin to sweep as the external source steps in frequency. It should look something like this:



Because of the reference mixer, the uncorrected VMC measurement can look like it has gain.

## Problems?

### Not sweeping:

- On the PNA, press **TRIGGER**, then **Continuous** to start the PNA sweeping. Watch for error messages on the PNA and source.

### Problems communicating with the source:

- Press **SYSTEM** then **Configure** then External Source. On the Select Sources dialog, click Configure. Select your LO source. Click **Software CW trigger**, then close the dialog. Again, press **TRIGGER**, then **Continuous** to start the PNA sweeping. If this works, then something is wrong with **Hardware (BNC)**. Check the trigger cables on the rear panel.
- As a last resort, try rebooting the PNA. First, [save the entire setup to a .csa file](#). When the PNA preset measurement appears, recall the .csa file to resume at this step.

### If the source is sweeping, and the PNA Input is sweeping, but there is still no output.

- Check power levels at the LO and Input.
- Check the DUT by making a fixed LO measurement - much easier.

## Perform a VMC Calibration

**Note:** Optionally perform a [Source Power Cal](#) before the VMC Cal to specify the LO Power at the DUT. This requires a power meter be connected to the GPIB.

1. Disconnect the DUT.
2. Connect the ECal module to a PNA USB port.
3. Press **CAL**, then **[Start Cal]**, then **[Cal Wizard]**. Because the VC21 measurement is active, the Cal Wizard automatically begins a VMC Calibration.
4. At the **Calibration Setup** dialog, click **Next**.
5. At the **Calibration Mixer Characterization** dialog, click **Next**. We will perform characterization of the Calibration mixer as part of the VMC cal. Later we will save the Calibration mixer characterization so that, in future VMC calibrations that use this same frequency range, we can recall the Calibration mixer characterization by clicking **Load Characterization from file**.
6. At the **Select DUT Connectors and Cal Kits** dialog, for **DUT Port 1** select the connector type and gender of your DUT INPUT. For **DUT Port 2** select the connector type and gender of your DUT OUTPUT. Then select ECal as the Cal Kit to use for each connector. Click **Next**.
7. At the **Select the ECal Port to be Connected** dialog, ensure that **Port A** is selected for **Port 1**, then click **Next**.

8. At the **Vector Mixer Calibration Step 1 of 3** dialog, connect the ECal module Port A to the Port 1 cable, and Port B to the Port 2 cable. Then click **Measure**. This portion of the calibration gathers the linear (non-frequency-translating) error terms of the test setup at the input and output frequencies.
9. At the **Vector Mixer Calibration Step 2 of 3** dialog, connect the following, then click **Measure**. This portion of the calibration will connect reflection standards to characterize the S-parameters of the calibration mixer/filter.
  - Port 1 cable to the Input of the calibration mixer.
  - LO cable to the LO port of the calibration mixer.
  - ECal module to the Output of the calibration mixer/filter.
10. At the **Vector Mixer Calibration Step 3 of 3** dialog, disconnect the ECal module and connect the Port 2 cable to the output of the calibration mixer/filter, then click **Measure**. This step completes the calibration using the characterized mixer/filter as a Thru standard.
11. At the **Save Mixer Characterization** dialog, click **Browse**, then type a unique filename and click **OK**. Then click **Next**. This saves the Calibration Mixer characterization to an S2P file. This file can be recalled for subsequent VMC calibrations.
12. At the **Calibration completed** dialog, you can choose to save the VMC calibration as a User Cal Set. Otherwise, click **Finish** to complete the VMC calibration. Correction is turned ON and applied to the VMC trace that we set up earlier.

### What is happening?

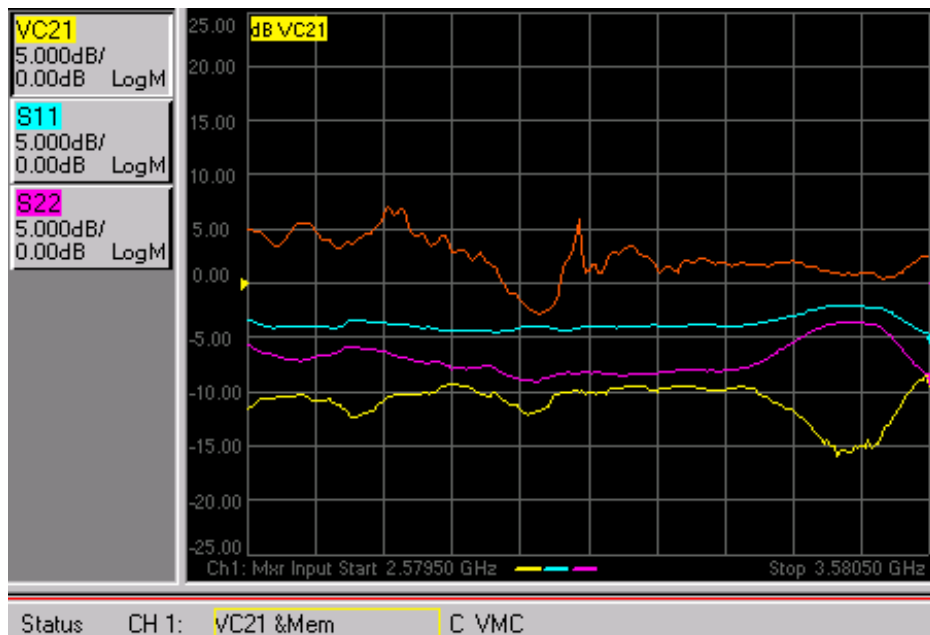
When an external source is sweeping, the measurements are much slower. When correction is ON, you will see that there are times when nothing is happening on the screen. This is because there are background measurements being made but not displayed.

This is exactly the same as when full 2-port correction is applied to an S-parameter. All four parameters are measured, then correction is applied, then all four measurements are updated. This occurs much faster when there is no external source. With a VMC measurement, there is no VC12 (reverse transmission measurement), so there are only three background measurements. With correction OFF, the traces are updated as the data is measured. You can see this taking place by creating the following measurements.

### Create S11 Input and S22 Output Match

1. Press **MEAS** then [**S11**] and [**S22**] to add these measurements to the same channel.
2. While the source is sweeping, watch the source port indicator on the front of the PNA. First, the port 1 indicator will light for two sweeps, then the port 2 indicator will light for 1 sweep while all 3 traces update.
3. Press **CAL**, then [**Correction OFF**]. Notice that the relevant traces will update as the sweep is occurring.

The following image shows the corrected Conversion Loss (VC21), Input Match (S11), Output Match (S22) and the uncorrected Conversion Loss (VC21), which is a memory trace.



Last Modified:

- 16-Apr-2009 Updated for PNA-X
- 6-Apr-2009 Replaced N5242a with PNA-X
- 5-May-2008 Added selectable output note.
- 17-Apr-2008 Added 10 MHz Ref image

## How to make an SMC Fixed Output Measurement

---

The following is a step-by-step example illustrating how to measure a mixer in swept LO mode using FCA Scalar Mixer Calibration.

There are fewer components required for SMC as compared to [VMC](#), and fewer measurement steps. Therefore, if you don't need to make relative phase measurements, SMC is an easier measurement. Also, ONLY SMC (not VMC) can measure the reverse conversion loss of the mixer.

This procedure can also be used for making **fixed** LO measurements, which is quite similar. Although the external source is still required, the physical triggering cables that connect the PNA and External Source are not required.

### Required Equipment

- PNA-X, E8362B/C, E8363B/C, E8364B/C, E8361A/C, or N5230A/C
  - with option 083 (FCA) or option 082 (SMC)
  - with PNA Rev. 6.03 or greater
- GPIB External Source (Agilent ESG or PSG works best). Not necessary when using PNA-X with Internal Second source.
- ECal module with connectors that match the Input and Output connectors of the DUT. You can use adapters to make the ECal module match the DUT connectors, but first perform an [ECal user-characterization](#) with the adapters attached. ECal makes the FCA calibration much easier.
- GPIB or USB power meter / sensor
- Cables and adapters

**Note:** This procedure refers to an External Source to control the LO. If using a PNA-X with an [Internal second source](#), an external source is not necessary. Connect the LO directly to the second source output.

### The example mixer

The example device is a down-converter mixer with the following characteristics:

- LO and Input Frequency Range: 2 GHz to 4.2 GHz
- Output Frequency Range: DC to 1.3 GHz

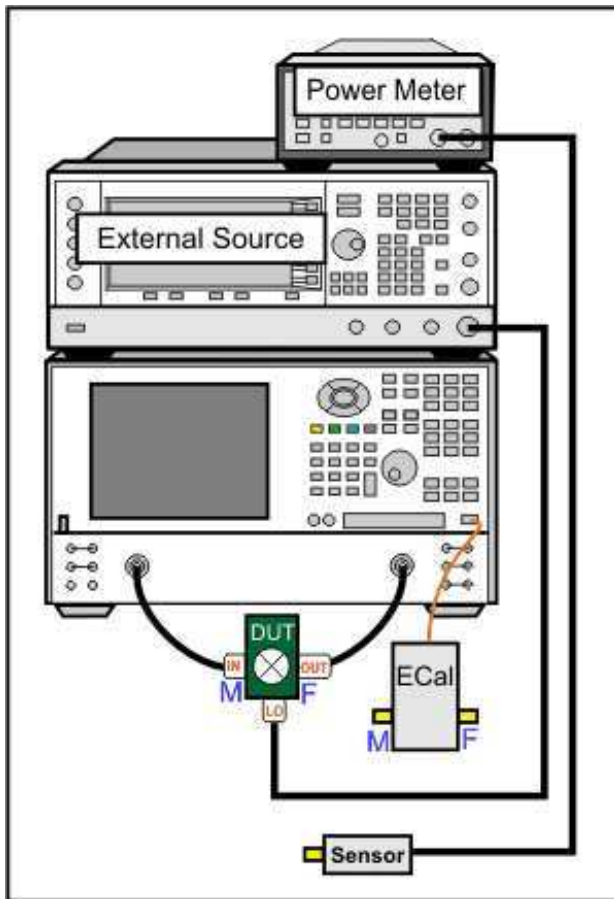
We will measure:

- Fwd Conversion Loss (SC21)
- Input Match (S11)
- Output Match (S22)
- Reverse Conversion Loss (SC12)



## SMC Setup

Connect the devices as shown in the following diagram:



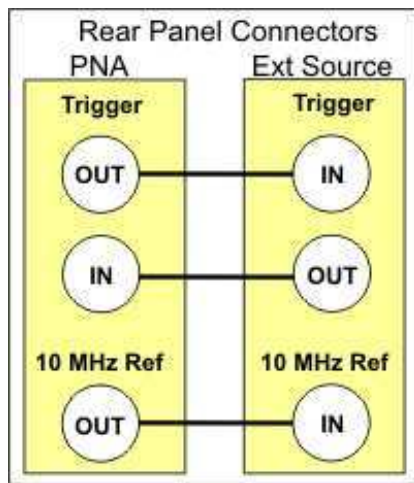
The DUT can be connected to any PNA ports. [Learn more.](#)

This procedure uses DUT input to PNA port 1 and output to PNA port 2.

This procedure refers to an External Source to control the LO. If using a PNA-X with an [Internal second source](#), an external source is not necessary. Connect the LO directly to the second source output.

### Make Connections on the Instrument rear panels:

1. If using a GPIB power meter, connect the power meter to the external source GPIB connector. If using a USB power meter, connect it to any unused USB port.
2. If using a PNA-X with an [Internal second source](#), the following three steps are not necessary. Connect the LO directly to the second source output.
3. Using a GPIB cable, connect the PNA GPIB controller port to the external source GPIB connector.
4. Using two BNC cables, connect the Source and PNA Trigger connectors as shown in the following image. This is not necessary when making fixed LO measurements.
5. Using a BNC cable, connect the PNA **10 MHz Reference Output** to the Ext. Source **10 MHz Reference Input**.



## Create the Measurement

For this document:

- Front-panel hardkeys are formatted as "Press **TRACE**"
- Front-panel softkeys are formatted as "Press **[S11]**"
- Menus are formatted as "Click **System**"

1. Connect the DUT.
2. On the PNA, press **SYSTEM**, then **[Configure]**, then **[Power Meter Settings]**.
3. Under Interface, select **GPIOB**, then enter the power meter address. Or select **USB**, then select the USB power meter that is connected to the PNA.
4. Press **PRESET** to make sure you are starting with a known state.
5. Press **MEAS** then **[Measurement Class]**, then **Scalar Mixer/Converter**, then **OK**. At the **Confirm...** dialog, click **OK**. An S11 trace is created.
6. Press **[SC21]** to replace the S11 trace.

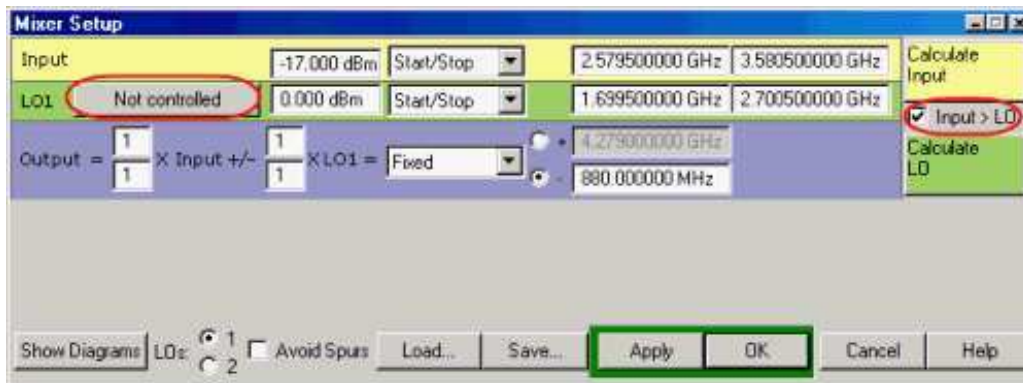
## Configure the Mixer settings

1. Press **FREQ** then **Input** to start the Mixer Setup dialog.
2. Enter the Mixer setup values as shown in the image below.

Notes:

- To provide quicker access to the Mixer Setup dialog, use the Setup softkey. [Learn how](#).

- Rather than enter ALL of the frequency settings, you can enter the Input and the Output frequencies, then click **Calculate LO**.
- If **Input>LO** is NOT checked, the PNA assumes you want the Input < LO frequencies, and higher LO frequencies are calculated as a result.
- The LO power level setting specifies the power out of the external source; not at the DUT) unless an LO power cal is performed.
- The Avoid Spurs feature is useful for eliminating spurs in test setups with excessive LO leakage.
- When the settings are valid, the background color around the **Apply** and **OK** buttons changes from Red to Green.



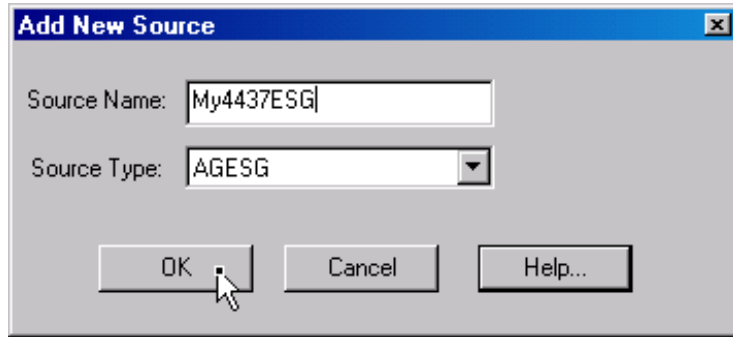
### Configure the External LO Source

1. Click **Not Controlled** to set up the External LO source. The following dialog appears:

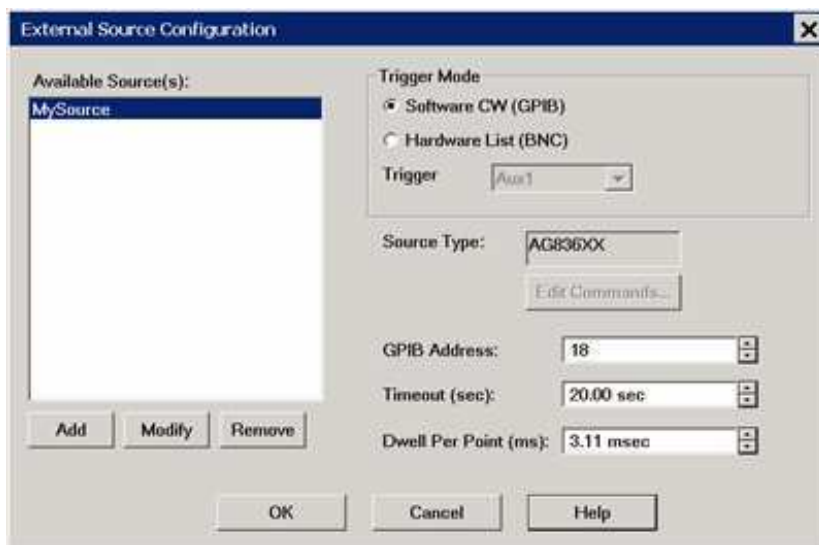


2. Click **Select External Sources** to perform a one-time configuration of the external source.
3. On the Select Sources dialog, click **Configure**
4. If the source that you want to control appears, select it, then skip to Step 4. Otherwise, click **Add**, to add a

source.



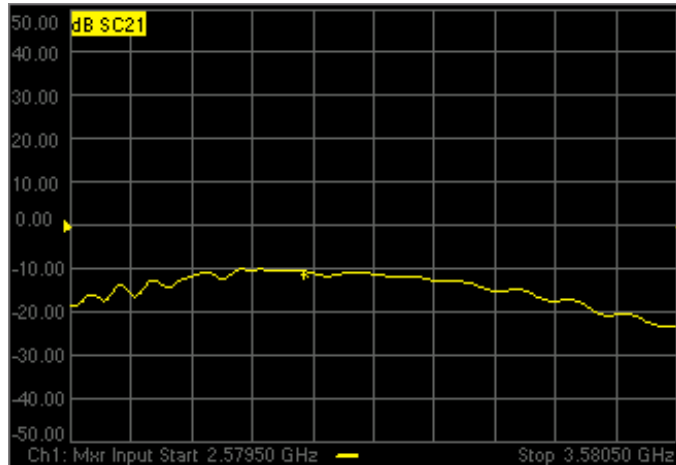
5. In the Add New Source dialog, type an identifying Source Name, such as the model of your source. In Source Type, select the model of source you are using. Then click **OK**.
6. Click **Hardware List (BNC)**, which is the fastest measurement method. This method requires the BNC Trigger cables that connect the PNA and source. If not available, **Software CW** can be used, but measurements are much slower.
7. If necessary, change the GPIB Address to match that of the source. This is NOT automatically detected.



Depending on the model of your source, this is what the External Source configuration dialog looks like AFTER entering settings.

8. On the open dialogs, make sure that the source is selected, then click **OK**. If there is a problem communicating with the source, the PNA will display an error. [See Problems?](#)
9. Click **OK** to return to the Mixer Setup dialog. The Not Controlled should now be replaced with the name of the source to use as the LO.
10. Save the mixer settings in a file so you can recall them easily. Click **Save...**, then type a descriptive filename, such as "FixedOutputMixer".

11. The trace should begin to sweep as the external source steps in frequency. It should look something like this:



### Problems?

#### Not sweeping:

- On the PNA, press **TRIGGER**, then **Continuous** to start the PNA sweeping. Watch for error messages on the PNA and source.

#### Problems communicating with the source:

- Press **SYSTEM** then **Configure** then External Source. On the Select Sources dialog, click Configure. Select your LO source. Click **Software CW trigger**, then close the dialog. Again, press **TRIGGER**, then **Continuous** to start the PNA sweeping. If this works, then something is wrong with **Hardware (BNC)**. Check the trigger cables on the rear panel.
- Can the PNA communicate with the power meter? If not, there is something wrong with the GPIB or USB communication.
- As a last resort, try rebooting the PNA. First, [save the entire setup to a .csa file](#). When the PNA preset measurement appears, recall this .csa file and continue at this step.

#### If the source is sweeping, and the PNA Input is sweeping, but there is still no output.

- Check power levels at the LO and Input.
- Check the DUT by making a fixed LO measurement - much easier.

**Tip:** You can optionally calibrate the LO Power level at the DUT using a standard [Source Power Calibration](#). Select the source port in the Source Power Cal dialog.

### Perform an SMC calibration

1. Disconnect the DUT.
2. Connect the ECal module to a PNA USB port.
3. Press **CAL**, then **[Start Cal]**, then **[Cal Wizard]**. Because the SC21 measurement is active, the Cal Wizard automatically begins an SMC calibration.
4. At the **Calibration Setup** dialog, click **Next**.
5. At the **Select DUT Connectors and Cal Kits** dialog, for **DUT Port 1** select the connector type and gender of your DUT INPUT. For **DUT Port 2** select the connector type and gender of your DUT **OUTPUT**. Then select ECal as the Cal Kit to use for each connector. Click **Next**.
6. At the **Scalar Mixer Calibration Step 1 of 2** dialog, connect the power sensor to the Port 1 test cable, then click **Measure**. The data will be used to correct for input mismatch errors.
7. At the **Scalar Mixer Calibration Step 2 of 2** dialog, connect the ECal module Port A to the Port 1 cable, and Port B to the Port 2 cable. Then click **Measure**. This portion of the calibration gathers the linear (non-frequency-translating) error terms of the test setup at the input and output frequencies.
8. At the **Calibration completed** dialog, you can choose to save the SMC calibration as a User Cal Set. Otherwise, click **Finish** to complete the SMC calibration. Correction is turned ON and applied to the SMC trace.

### What is happening?

When an external source is sweeping, the measurements are much slower. When correction is ON, you will see that there are times when nothing is happening on the screen. This is because there are background measurements being made but not displayed.

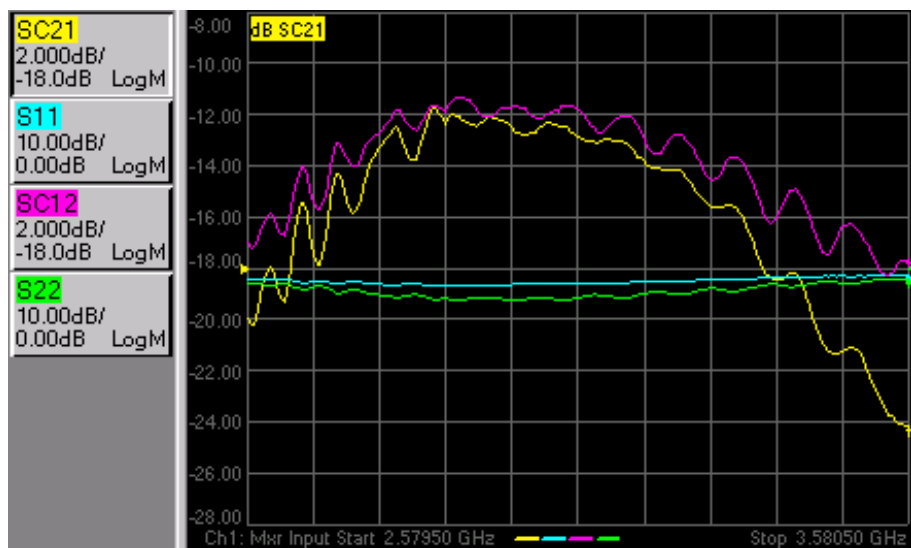
This is exactly the same as when full 2-port correction is applied to an S-parameter. All four parameters are measured, then correction is applied, then all four measurements are updated. This occurs much faster when there is no external source. With correction OFF, the traces are updated as the data is measured. You can see this taking place by creating the following measurements.

### Create S12 Upconverter, S11 Input and S22 Output Match

1. Press **MEAS** then **[S11]**, **[SC12]**, and **[S22]** to add these measurements to the same channel.
2. While the source is sweeping, watch the source port indicator on the front of the PNA. First, the port 1 indicator will light for two sweeps, then the port 2 indicator will light for 2 sweeps. During the last sweep, all 4 traces update.
3. Press **CAL**, then **[Correction OFF]**. Notice that the relevant traces update as the sweep is occurring.

With the SC12 measurement you can see the reciprocity of the mixer.

Note: With the recent improvements to FCA, this step is MUCH easier than before. SMC forward and reverse measurements can now reside in the same channel and are calibrated automatically at the same time.



Last Modified:

- 14-Apr-2010 Added N5230A/C and Opt 082
- 6-Apr-2009 Updated for PNA-X
- 5-May-2008 Modified for selectable ports
- 17-Apr-2008 Added 10 MHz ref image

## Embedded LO Measurements

---

The Embedded LO feature allows you to make [VMC](#), [SMC](#), [IMDx](#), [IMx Spectrum](#), and [NFX](#) measurements of mixers that have a FIXED LO inside the DUT.

**Note:** This feature is available as Opt 084, and must be [enabled](#).

Measurements of these devices are challenging for a couple of reasons:

1. The VMC measurement process requires the use of a [reference mixer](#) that has the same LO frequency as the DUT. A separate internal or external source must be used for the reference mixer LO. [This LO must be controlled by the PNA](#). A PNA with an internal second source is much faster.
2. All Embedded LO measurements require the PNA receivers to be tuned to the correct frequency to measure the mixer output, which is highly dependent on the exact LO frequency.

### How the PNA measures the embedded LO

The nominal frequency of the embedded LO is input into the [Mixer Setup dialog](#). This is used as a starting point for the measurement.

Before each DUT measurement sweep, background sweeps are made to determine the frequency of the embedded LO to a configurable degree of accuracy.

Background sweeps...

- **Broadband Sweep** - rough measurement of the embedded LO frequency, made around a selectable data point over a selectable frequency span. The input signal to the DUT is tuned to a selectable CW frequency. The B receiver is swept across a selectable span around the anticipated output frequency. The difference between the frequency of the found signal and the desired output frequency is then applied as an adjustment.
- **Precise Sweep** The B receiver is measured at the selectable data point. Measurements of phase versus time are made, from which the exact offset frequency is computed, until either the tolerance value or maximum iterations are met.
- For VMC measurements, the reference mixer frequency is updated as the embedded LO frequency is determined.

### How to measure a DUT with an Embedded LO

1. Create a [VMC](#), [SMC](#), [IMDx](#), or [NFX](#) measurement.
2. In the mixer setup dialog, enter the nominal frequency of the embedded LO as the LO frequency.
3. Perform a calibration as usual.
4. [Launch and complete the Embedded LO Mode dialog box](#) (below)

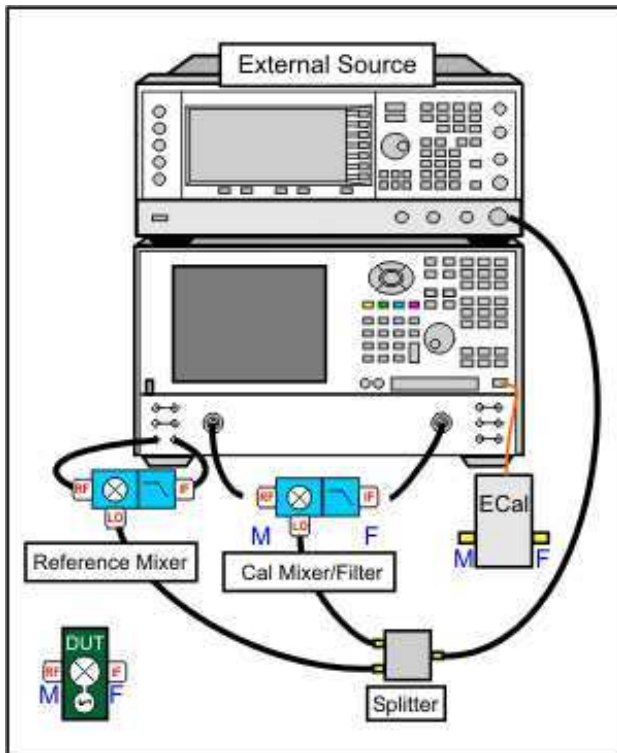
**For VMC (FCA App) ONLY:**



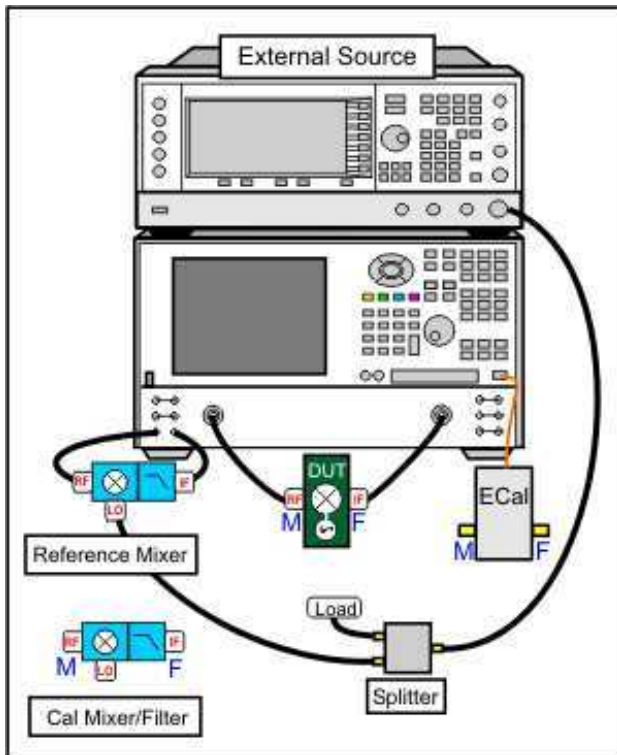
The LO source for the Reference Mixer can be either:

- An **Internal** source when using a [PNA-X that has two sources](#).
- An **External** source:
  - Must be controlled by the PNA. [Learn how.](#)
  - Must be locked to the PNA using the [10 MHz reference](#).

**During Calibration** - The LO source is shared between the Reference Mixer and the Calibration Mixer/Filter. This requires a splitter when using an external source, as shown in the following image.



**During the Measurement** - Only the Reference Mixer uses the LO source. Terminate the LO source port that is no longer used by the Calibration Mixer/Filter to ensure that the match seen by the Reference Mixer LO port does not change after the calibration, as shown in the following image. This precaution is not necessary when using the internal second source (ports 3 and 4) of the PNA-X.



**For SMC, IMDx, IMxSpectrum, and NFX measurements:**

No unique setup is required for embedded LO measurements.

#### How to Launch the Embedded LO Mode dialog box

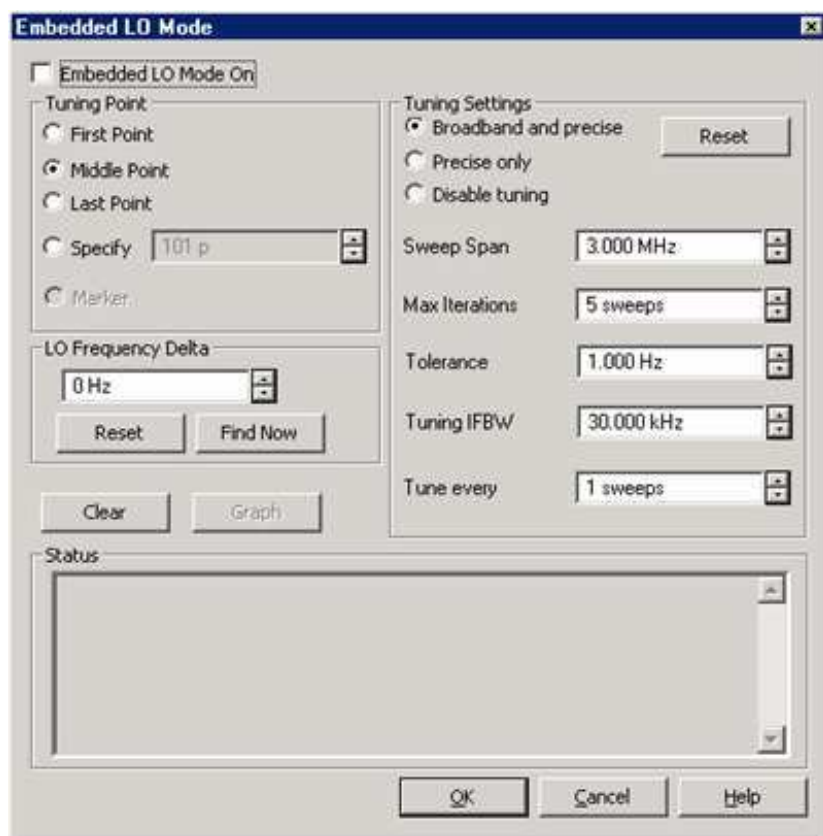
**Using front-panel  
HARDKEY [softkey] buttons**

1. Not Available

**Using a mouse with PNA Menus**

1. Click **Trace/Chan**
2. then **Measure**
3. then **Embedded LO**

**Programming Commands**



### Embedded LO dialog box help

The Tuning Settings balance LO measurement speed versus accuracy. You can see that accuracy is becoming compromised when noise starts to appear on the measurement trace.

[Scroll up](#) to learn more about the Embedded LO measurement process.

**Embedded LO Mode On** Check to enable measurement of the Embedded LO.

**Tuning Point** Select, or specify, the data point in the mixer sweep that will be used to find the embedded LO frequency. If a marker is enabled, that data point can be used. For broadband and Precise sweeps, choose a point in the mixer sweep where noise is least likely to be found. This is generally the center of a sweep or the center of a filter if used.

**LO Frequency Delta** The absolute difference between the measured embedded LO frequency and the LO setting that is entered in the [Mixer Setup dialog](#). This value is updated each time the embedded LO frequency is measured. Entering a value is a way to change the LO frequency on the mixer setup without invalidating the calibration.

**Reset** Set the LO Frequency Delta back to 0 Hz.

**Find Now** The PNA finds and measures the actual LO frequency using the current dialog settings. This data is displayed in the **Status** box.

**Tuning Settings** These settings determine the amount of time spent versus the degree of accuracy to which the LO Frequency is measured. You can see that accuracy is becoming compromised when noise starts to appear on the measurement trace.

**Reset** Set all Tuning Settings back to the defaults.

**Broadband and Precise** Do the entire tuning process for each background sweep.

**Precise only** Does NOT perform broadband tuning on each sweep. Use this setting when the embedded LO is stable.

**Disable tuning** Only the previously measured LO Frequency Delta is applied to the reference mixer LO and PNA receivers.

**Sweep Span** Narrowing the sweep span limits the number of data points that are measured in the broadband sweep and makes the measurement faster.

**Max Iterations** The maximum number of Precise sweeps to make. When this number is reached, the final measurement is used.

**Tolerance** When two consecutive Precise measurements are made within this value, the final measurement is used. If this is not achieved within the Max Iterations value, then the last measurement is used. This is the best of the 'Tunings settings' to change to improve accuracy.

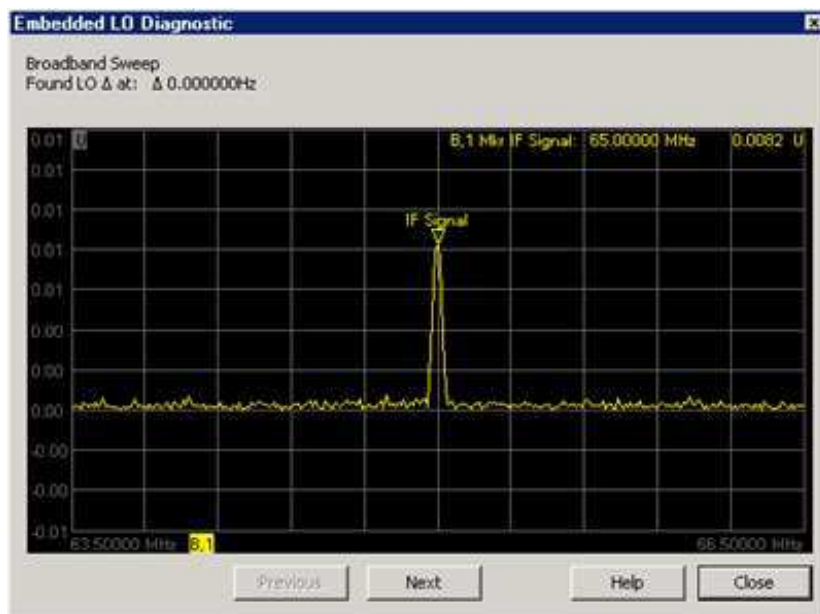
**Tuning IFBW** IF Bandwidth used for Broadband and Precise tuning sweeps. The larger the IFBW, the faster the sweep, but the signal may not be found.

**Tune every** Set the interval at which tuning is performed before a measurement sweep. 'Tune every 3 sweeps' means that every third measurement sweep is preceded by tuning sweeps. If the embedded LO drifts, or if regularly changing DUTs, use 'Tune every 1 sweep'.

**Status** Allows textual and graphical representation of the Embedded LO measurement sweeps.

**Clear** Removes the text information currently being displayed.

**Graph** Launches the following graphical (spectrum analyzer type) display sweeps of the latest embedded LO measurement.



## Embedded LO Diagnostic dialog box help

Presents a graphical (spectrum analyzer type) display of the latest embedded LO measurement.

Click **Previous** and **Next** to view available Broadband and Precise sweeps. The LO Frequency is displayed in the Marker annotation.

---

### Last Modified:

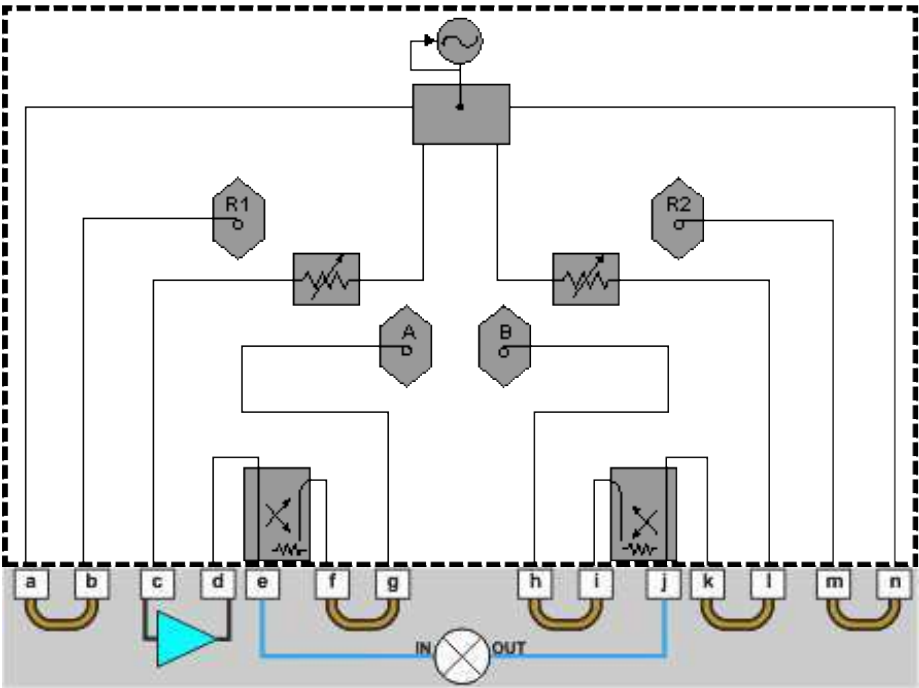
12-Aug-2009	Added SMC and IMD measurements
3-Sep-2008	Removed legacy content
5-Oct-2007	Added config image and text
5-Jul-2007	Update access point
6-Apr-2007	MX New topic

## SMC with a Booster Amp

If your mixer measurement requires more source power on the input than the PNA can provide, a booster amplifier can be used to provide the additional power. This topic describes how to configure and make a calibrated SMC measurement using a booster amplifier.

### Connect

Connect the booster amplifier between the Source-Out and Coupler-Thru connectors on the front-panel as shown in the following diagram. Your PNA block diagram may not look like this.



Item	Description	Item	Description
a	SOURCE OUT	h	RCVR B IN
b	RCVR R1 IN	i	CPLR ARM
c	SOURCE OUT	j	PORT 2
d	CPLR THRU	k	CPLR THRU
e	PORT 1	l	SOURCE OUT
f	CPLR ARM	m	RCVR R2 IN
g	RCVR A IN	n	SOURCE OUT

### Measurement and Calibration Setup

In the following procedure:

- **Test Port power** is the power level out of the source.
- **Corrected power** is the power level you require at the mixer input and output.

This procedure assumes you will applying stimulus power to the mixer **input** to make SC11 and SC21 measurements, and to the **output** of the mixer to make SC22 and SC12 measurements.

1. Determine the gain of the booster amplifier. If the gain has significant slope across the **input and output range** of the mixer, see [Booster Amp with a Gain Slope](#).
2. Determine the corrected power for both the input (port 1) and output (port 2) of the mixer.
3. Calculate the Test Port power for both ports by subtracting the gain of the amplifier from both the input and output corrected power levels.

For example, the following values assume a 25 dB booster amp on port 1 as in the diagram above.

	Corrected Power	-	Amp Gain	=	Test Port Power
Port 1 (input)	0 dBm	-	25 dB	=	-25 dBm
Port 2 (output)	-20 dBm	-	25 dB	=	-45 dBm

4. On the PNA [Power dialog](#), clear the **Port Power Coupled** checkbox, which allows different power levels for each port.
5. Enter the calculated **Test Port Power** values for each port.
6. During the SMC Cal Wizard **Select DUT Connectors and Cal Kits** dialog, click **View/Modify Source Cal Settings** to invoke the [Source Calibration Settings dialog](#).
7. In **Power Offset**, enter the booster amplifier gain.

### Booster Amp with a Gain Slope

SMC calibration takes place over the entire input and output range of the mixer. Therefore, the booster amplifier will also be subjected to the entire input and output frequency range of the mixer.

To compensate for a gain slope, you might have to experiment with the source attenuator setting, power-offset value, and initial power value to get a combination that will not cause the PNA source to go unlevelled during or after the cal.

For example, assume the booster amp gain is 30 dB at the low end, and 20 dB at the high end. If you enter 30 dB for the power offset value, the PNA might run out of ALC range when the actual gain drops to 20 dB. The PNA will try to increase its source power to account for the 10 dB gain drop. Therefore, pick a power offset value that is in the middle of the amplifier gain band (25dB).

If possible, select a PNA attenuator setting that puts the ALC approximately in the middle of its range at the desired corrected power with the mid-band gain. This condition means the ALC can set the power higher and lower to account for the gain slope, without unleveling.

If the gain slope is too large, then there may not be a setting that prevents a source unlevel. In this case, a flatter booster amp must be used.

Last Modified:

19-May-2008    Edited for generic image



## Frequency Offset Mode

Frequency Offset Mode (FOM) provides the capability to have the PNA Sources tune to frequencies that are different (offset) from the PNA Receivers.

PNA Option 080 provides you with the hardware and basic software capability to make Frequency Offset Measurements. This topic discusses the PNA settings that are relevant to making these types of measurements. See [Frequency Converting Device Measurements](#) for more information on making specific device measurements.

- [Frequency Offset Dialog Box](#)
- [Setup Examples](#)
- [Test Set \(Reference Switch\) Dialog Box](#)

### Other Frequency Offset topics

#### How to make Frequency Offset settings

##### Using front-panel HARDKEY [softkey] buttons

1. Press **STIMULUS**
2. then **[Frequency Offset]**

##### PNA Menu using a mouse

1. Click Stimulus
2. then **Frequency**
3. then **Frequency Offset**

#### Programming Commands

	Mode	Sweep Type	Settings
Primary		Linear Frequency	300.000000 kHz - 20.000000 GHz, 201
Source	Coupled	Linear Frequency	300.000000 kHz - 20.000000 GHz
Receivers	Un-Coupled	Segment Sweep	300.000000 kHz - 1.000000 MHz
Source2	Coupled	Linear Frequency	300.000000 kHz - 20.000000 GHz

X-Axis Display  
Annotation: Receivers ☐ X-Axis Point Spacing

OK Cancel Help

#### Frequency Offset dialog box help

The following are major changes to FOM:

- Stimulus and Response are now called Sources and Receivers.
- Sources and Receivers settings can be made in two ways:
  1. By **Coupling** to the Primary (Channel) settings. This is the only method used in previous releases.
  2. By **Uncoupling** and setting Sources and Receivers values independently. This is the new, simplified method.
- External sources can be controlled from this dialog. [Learn more.](#)

**Note:** Source2 supplies power for ports 3 and 4. **Turn Source2 power ON** using the [Power and Attenuators dialog](#). This (Frequency Offset) is the only dialog for controlling the frequency of Source 2. Learn more about [Source2](#).

**Frequency Offset (ON/OFF)** Enables Frequency Offset Mode on ALL measurements that are present in the active channel.

When FOM is NOT enabled, all frequencies are the same as the active channel.

**Tip:** First make other settings on this dialog box, then click **Frequency Offset ON**.

**Primary** The current Active Channel settings. When a Source or Receiver is coupled to the Primary settings, its Sweep Type is the same as that of the Primary. The frequency settings of the coupled range are mathematically derived from the Primary settings using the [Multiplier, Divisor, and Offset values](#). With this approach, only the Primary settings need to be changed in order to affect change in the coupled Sources and Receivers. Changes to the Primary channel settings occur when Frequency Offset is checked ON. [See example using Primary and Coupled setting.](#)

**Tip:** Primary settings are ONLY used when Sources and Receivers are Coupled. It is often easier to Uncouple, then set Sources and Receivers independently.

Source and Source2 if available. [Learn more about Internal Second Source.](#)

**Receivers** All receivers that are used in the channel, including Reference receivers, are tuned to the specified frequency settings.

## Mode

**Coupled** Source and Receiver settings are mathematically derived from the Primary settings using Multiplier, Divisor, and Offset values. [Learn more.](#)

**Uncoupled** Source and Receiver settings are entered independently, without reference to Primary settings. When Uncoupled, Source and Receiver Ranges can use separate sweep types.

**Sweep Type** Click to change the type of sweep for each range. Only available for Primary and Uncoupled Sources and Receivers.

### Unsupported Sweep Type combinations

- Power Sweep and Segment Sweep can NOT be used together.
- Uncoupled Log Sweep yields **invalid data** whenever the sources are offset from the receivers.
- Coupled Log Sweep is allowed only for the following two conditions:
  1. The offset = 0, the multiplier = 1, and the divisor = 1.
  2. The multiplier = 0

**Settings** To change settings, click **IN** the appropriate Settings cell, then click **Edit**.

- If coupled, invokes the [Coupled dialog](#).
- If uncoupled or Primary invokes the [Uncoupled settings dialog](#).

**X-Axis** Select the settings to be displayed on the X-Axis.

**X-Axis Point Spacing** Only available when a Segment Sweep Type is selected as the X-Axis display. [Learn more](#).

**Note:** When Frequency Offset is enabled, ALL receivers on the channel, including the reference receivers, tune to the new offset frequencies, Therefore the source and reference receiver will be at different frequencies. Therefore, FOM measurements that include a reference receiver, which includes all S-parameters, display invalid data.

To measure and display measurements at both the source and receiver frequencies, you must use two channels. Use [Equation Editor](#) to calculate the conversion loss. [See a calibrated FOM conversion loss example](#).

[Learn how to calibrate frequency offset measurements](#).

**Source (Coupled)**

Frequency:  
Offset: 1.000000 GHz  
Multiplier: 15  
Divisor: 1

Start Frequency 21.000000 GHz  
Stop Frequency 27.500000 GHz

OK Cancel Help

## Coupled settings dialog box help

### Coupled Formulas:

Range Start = [Primary Start x (Multiplier / Divisor)] + Offset

Range Stop = [Primary Stop x (Multiplier / Divisor)] + Offset

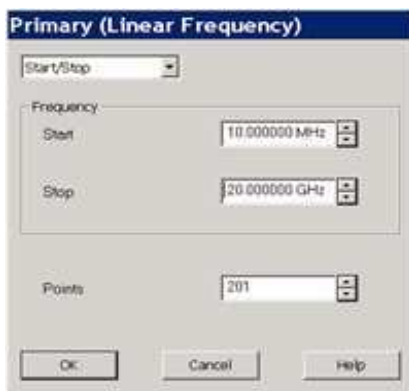
### Where:

**Offset** Specifies an absolute offset frequency in Hz. For mixer measurements, this would be the LO frequency. Range is +/- 1000 GHz. Offsets can be positive or negative.

**Multiplier** Specifies (along with the divisor) the value to multiply by the stimulus. Range is +/- 1000.

- Negative multipliers cause the stimulus to sweep in decreasing direction. For downconverter mixer measurements, this would be for setups requiring the Input frequency to be less than LO frequency. [See an example.](#)
- 0 (zero) as the multiplier nulls the Primary setting. Then the Offset value adds to zero.

**Divisor** Specifies (along with the multiplier) the value to multiply the stimulus. Range is 1 to 1000.



## Primary and Uncoupled settings dialog box help

This dialog will vary depending on the sweep type:

### Linear and Log frequency

Uncoupled Log sweep yields **invalid data** whenever the sources are offset from the receivers.

Select Start/Stop or Center/Span

**Frequency** Enter values

**Points** (Primary only) Enter number of data points for the sweep.

### Power

**CW Freq** Enter frequency in Hz.

**Points** (Primary only) Enter number of data points for the power sweep.

## CW Time

**CW Freq** Enter frequency in Hz.

**Sweep Time** Enter time to complete one sweep. Enter 0 for the fastest sweep.

**Segment Sweep** Edits are made exactly like the [standard segment table](#).

**For Advanced Users:** Uncoupled Segment Sweep offers great flexibility in configuring measurements. In segment sweep mode:

- The **OK** button is NOT available until the total number of data points for all segments matches the number of Primary data points.
- [Independent IF Bandwidth](#) and [Independent Sweep Time](#) are available ONLY on the Primary (channel) and the Uncoupled **Receivers** - NOT Sources.
- [Independent Power](#) is available ONLY on the Primary (channel) and the Uncoupled **Sources** - NOT Receivers.

## Setup Examples

Although the Frequency Offset settings can be used with many types of devices, these examples include mixer terminology.

[See a Mixer Compression and Phase \(AM-PM\) Measurement using FOM.](#)

[See a calibrated FOM conversion loss example.](#)

### 1. Fixed LO - Upconverter

- **Swept Stimulus (Mixer Input):** 1000 MHz - 1200 MHz
- **Fixed LO:** 1500
- **Swept Response (Mixer Output):** 2500 MHz to 2700 MHz

Make the following settings on the FOM dialog

**Source:** Uncoupled

Sweep Type: Linear

Click Settings, then Edit. In the Source dialog:

Start Frequency = 1000 MHz

Stop Frequency = 1200 MHz

**Receiver:** Uncoupled

Sweep Type: Linear

Click Settings, then Edit. In the Receiver dialog:

Start Frequency = 2500 MHz

Stop Frequency = 2700 MHz

## LO Settings

Set external source to CW - 1500 MHz.

**Source2:** Uncoupled (Only with [Second PNA Internal Source](#))

Sweep Type: CW Time

Click Settings, then Edit. In the Source2 dialog:

CW Frequency = 1500 MHz

---

## 2. Fixed LO - Downconverter (Input < LO)

- **Swept DECREASING Stimulus (Mixer Input):** 1100 MHz to 1000 MHz
- **Fixed LO:** 2500 MHz
- **Swept INCREASING Response (Mixer Output)** 1400 MHz to 1500 MHz

Make the following settings on the FOM dialog

**Primary:** Not used

**Source** (Input): Uncoupled

Sweep Type: Linear

Click Settings, then Edit. In the Source dialog:

Start Frequency = 1100 MHz

Stop Frequency = 1000 MHz

**Receiver** (Output): Uncoupled

Sweep Type: Linear

Click Settings, then Edit. In the Receiver dialog:

Start Frequency = 1400 MHz

Stop Frequency = 1500 MHz

## LO Settings

Set external source to CW - 2500 MHz.

**Source2:** Uncoupled (Only with [Second PNA Internal Source](#))

Sweep Type: CW Time

Click Settings, then Edit. In the Source2 dialog:

CW Frequency = 2500 MHz

---

## 3. Swept LO - Fixed Output - Upconverter

Swept External LO measurements in Frequency Offset Mode can be very difficult. The external LO source must be synchronized with the swept output or input (as in this case). See [Synchronizing and External Source Control](#) to see how this is done. The [Frequency Converter Application Opt 083](#) performs makes these measurements easily.

- **Swept Stimulus (Mixer Input):** 1000 MHz to 1100 MHz
- **Swept LO:** 1500 MHz to 1400 MHz
- **Fixed Response (Mixer Output):** 2500 MHz

Make the following settings on the FOM dialog

**Source:** Uncoupled

Sweep Type: Linear

Click Settings, then Edit. In the Source dialog:

Start Frequency = 1000 MHz

Stop Frequency = 1100 MHz

**Receiver:** Uncoupled

Sweep Type: CW Time

Click Settings, then Edit. In the Receiver dialog:

CW Frequency = 2500 MHz

#### **LO Settings**

- If using external source, set to sweep from 1500 - 1400 MHz.
- If using **Source2** ([Second Internal Source](#)): set to Uncoupled, then:

Sweep Type: Linear

Click Settings, then Edit. In the Source2 dialog:

Start Frequency = 1500 MHz

Stop Frequency = 1400 MHz

#### **4. Power Sweep for Mixers**

To measure the gain compression of a mixer, the input power to the mixer is swept. The input and output frequencies are fixed but offset from one another.

This is a good use of Coupled settings because the same compression test can be performed at several different frequencies. With coupled Source and Receiver ranges, the Primary (channel) frequency can be easily changed from the front panel. The coupled source and receiver frequencies will update accordingly.

- **Swept Input Power:** -10 dBm to 0 dBm
- **Fixed Input Frequency:** 1500 MHz
- **Fixed LO:** 500 MHz
- **Fixed Output:** 2000 MHz

Make the following settings on the FOM dialog

**Primary:**

Sweep Type: Power Sweep

Click Settings, then Edit. In the Primary dialog:

CW Frequency = 1500 MHz

**Source:** Coupled

Default settings make CW Frequency: 1500 MHz (same as Primary)

**Receiver:** Coupled

Default settings make Sweep Type: CW Time

Click Settings, then Edit. In the Receiver dialog:

Offset = 500 MHz

**LO Settings**

- If using external source, set to CW: 500 MHz.
- If using **Source2** ([Internal Second Source](#)), set to Coupled, then:

Sweep Type: Power Sweep

Click Settings, then Edit. In the Source2 dialog:

CW Frequency = 500 MHz

---

**Test Set Reference Switch**

PNA models with [option 081](#) and all PNA-X models have a switch in the test set that allows you to bypass the port 1 reference receiver through the front panel Reference 1 connectors. This switch lets you easily switch between standard S-Parameter measurements and measurements using a reference mixer. You could use this feature to make standard S11 measurements and converter transmission measurements relative to a reference ("golden") mixer.

**Note:** The Frequency Converter Application [Option 083](#) simplifies the task of making extremely accurate phase measurements on MOST frequency converting devices.



## How to access the Test Set dialog box

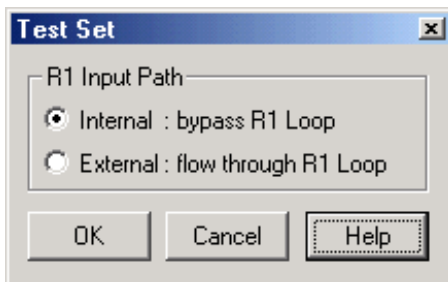
### Using front-panel HARDKEY [softkey] buttons

1. Press TRACE/CHAN
2. then [Channel]
3. then [More]
4. then [Path Config]
5. select

### PNA Menu using a mouse

1. Click **Trace/Chan**
2. then **Channel**
3. then **More**
4. then **Path Config**
5. **Select**

## Programming Commands



### Test Set dialog box help

**Note:** This feature is available on PNA models with [Option 081](#) (external reference switch) and all PNA-X models.

#### R1 Input Path

**Internal: bypass R1 Loop** Connects the port 1 source directly to the R1 receiver.

**External: flow through R1 Loop** Allows direct access to the R1 receiver through the Reference 1 front-panel connectors.

See [specifications](#) which include a block diagram of reference switch.

Last modified:

30-Apr-2009	Added PNA-X models for test set switch
3-Sep-2008	Removed legacy content
25-Feb-2008	Added link to AM-PM procedure
16-Oct-2007	Minor edits
11/21/06	MQQ Modified for new dialog

## Frequency Converting Device Measurements

---

Many frequency offset measurements can be made using the PNA with option 080. The following is a list of some of those measurements and how they are made.

- [Conversion Loss](#)
- [Conversion Compression](#)
- [Return Loss and VSWR](#)
- [Isolation](#)
- [Harmonic Distortion](#)

See Also: [Frequency Offset Measurement Accuracy](#)

## Frequency Offset Measurement Accuracy

---

This topic discuss methods that can be used to make accurate frequency offset measurements.

- [Calibrations](#)
- [Mismatch Errors](#)
- [Accurate and Stable LO](#)

[See other Mixer Measurement topics](#)

### Calibrations

With Frequency Offset measurements, the stimulus and response frequencies are different. Standard calibration error terms are calculated using reference measurements. Therefore, traditional calibration methods such as full 2-port SOLT cannot be used with frequency offset.

[Source and Receiver Power calibrations](#) can be used to calibrate your Frequency Offset measurements.

[Frequency Converter Application](#) (option 083) offers fully calibrated scalar and vector frequency offset measurements.

#### Source Power calibration:

- Sets accurate power level at stimulus frequencies regardless of the receiver that will be used in the measurement.
- Can be copied to other channels with copy channels feature.
- Can be interpolated.

#### Receiver Power Cal:

- Requires a source cal to have already been performed and applied.
- Cannot be copied to other channels.

#### Therefore:

- Start by performing a [source power cal](#) over the combined stimulus and response frequencies.
- [Copy the channel](#) to other needed channels and the source power cal is copied.
- Change the frequency range of the copied channel to response frequencies.
- Perform a [receiver cal](#) at the response frequencies on individual channels.

- Change the frequency range to stimulus frequency and switch [frequency offset ON](#).
- On [Status Bar](#), ensure that source and receiver cals are ON (source cal will be interpolated).

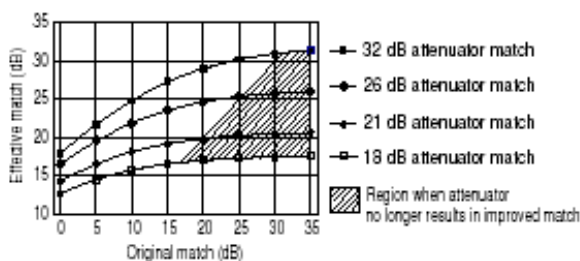
See [Frequency Offset Conversion Loss Measurements](#) to see a step-by-step example.

## Mismatch Errors

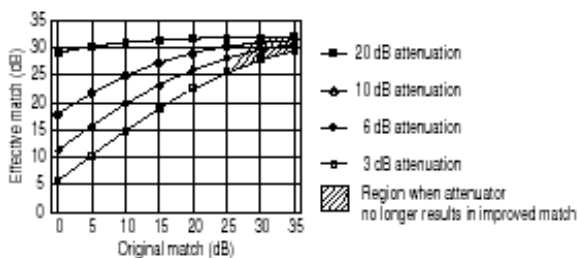
Mismatch errors result when there is a connection between two ports that have different impedances. With S-parameter measurements, these mismatches are measured and mathematically removed during a full 2-port calibration. This is much more difficult with frequency offset measurements. A much easier solution is to use high-quality attenuators on the input and output of the mixer.

By adding a high-quality attenuator to a port, the effective port match can be improved by up to twice the value of the attenuation. For example, a 10-dB attenuator, with a port match of 32 dB, can transform an original port match of 10 dB into an effective match of 25 dB. However, as the match of the attenuator approaches the match of the original source, the improvement diminishes.

**Note:** The Frequency Converter Application (option 083) uses calibration techniques that correct for mismatch errors.



The larger the attenuation, the more nearly the resulting match approaches that of the attenuator, as shown in the following graphic. However, excessive attenuation is not desired because that will decrease the dynamic range of the measurement system.



## Accurate and Stable LO

When using frequency offset mode, if the LO signal is not accurate and stable, the output signal will not be at the expected response frequency. As a result, the output signal can fall on the skirts of the PNA receiver IF filter, or fall completely outside of the receiver filter passband.

Also, the LO power level is critical in mixer measurements. Be sure to monitor these power levels closely.

## Conversion Loss (or Gain)

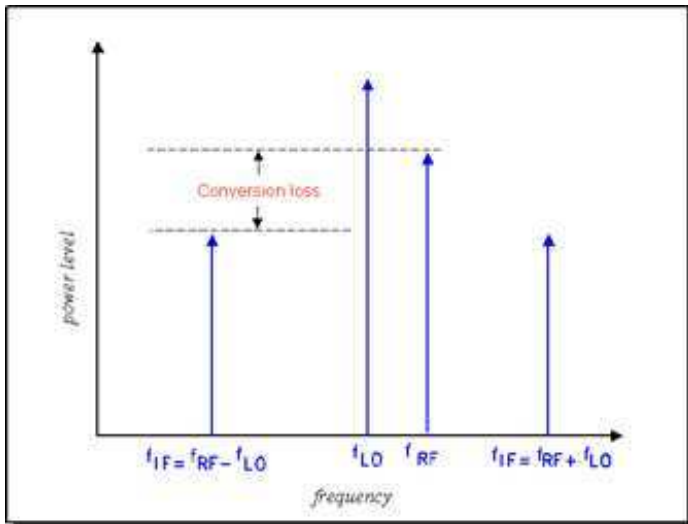
---

- [What is Conversion Loss?](#)
- [Why Measure Conversion Loss?](#)
- [How to Measure Conversion Loss](#)

[See other Frequency Converting Device Measurements](#)

### What is Conversion Loss?

Conversion loss is defined as the ratio of the power at the output frequency to the power at the input frequency with a given LO (local oscillator) power. This is illustrated in the graphic below. A specified LO power is necessary because conversion loss varies with the level of the LO, as the impedance of the mixer diode changes.



### Why Measure Conversion Loss?

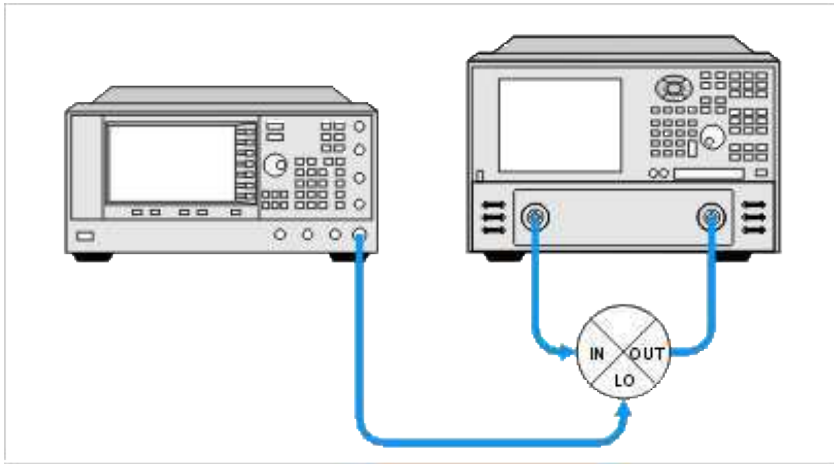
Conversion loss (or gain in the case of many converters and tuners) is a measure of how efficiently a mixer converts energy from the input frequency to the output frequency. If the conversion loss response of a mixer or converter is not flat over the frequency span of intended operation, valuable information may be lost from the resulting output signal.

### How to Measure Conversion Loss

Conversion loss is a transmission measurement. It is measured by applying an input signal (stimulus) and an LO signal at specific known power levels, and measuring the resulting output signal level. Because the output frequency is different from the input frequency, [frequency offset](#) mode (option 080) must be used for this measurement.

**Note:** This measurement is made much easier if your PNA has the [Frequency Converter Application](#)

### Equipment Setup



### Example: A calibrated Conversion Loss (Down-converter) measurement

#### Swept Input with Fixed LO = Swept Output

- RF Input: 3.1 - 3.3 GHz
- LO: 2.2 GHz
- IF Output: 900 - 1100 MHz

#### PNA setup and calibrate on channel 1

1. On channel 1 create an [unratioed](#) R measurement over the ENTIRE input and output frequency span (.9 - 3.3 GHz). This will be the base source power cal that will be copied to the R and B channel measurements.
2. Perform a [source calibration](#) using a power meter. This makes the power level at the input of the mixer very accurate.

#### Setup Reference measurement on channel 2

1. [Copy channel](#) 1 to channel 2 which will display the reference input to the mixer. The channel 1 source power cal is copied with the other channel settings.
2. Change measurement to R1 unratioed.
3. Change RF Input frequency to 3.1 - 3.3 GHz. The source power cal becomes interpolated.
4. Perform [receiver power cal](#). Do not need to make physical connections. The PNA source is internally connected to the R1 receiver. Makes the R receiver read the source power level.

## Setup B measurement on channel 3

1. Copy channel 1 to channel 3. This channel will display the output of the mixer. The channel 1 source power cal is copied with the other channel settings.
2. Change measurement to B unratioed.
3. Change IF Output frequency to .9 - 1.1 GHz. This causes the source power cal becomes interpolated.
4. Connect thru line from port 1 to port 2.
5. Perform receiver power cal. This makes the B receiver read the source power at the IF Output frequencies.
6. [Turn OFF receiver power cal.](#) This prevents an error when changing to input frequencies (next step).
7. Change RF Input frequency to 3.1 - 3.3 GHz. This changes the channel back to the mixer RF Input frequencies.
8. [Enable Frequency Offset.](#)
9. Change Offset to (-2.2 GHz). This tunes the B receiver to the IF Output frequencies .9 to 1.1 GHz. **Note:** The minus sign indicates a down-converter measurement.
10. Turn ON receiver power cal.

## Measure the Mixer

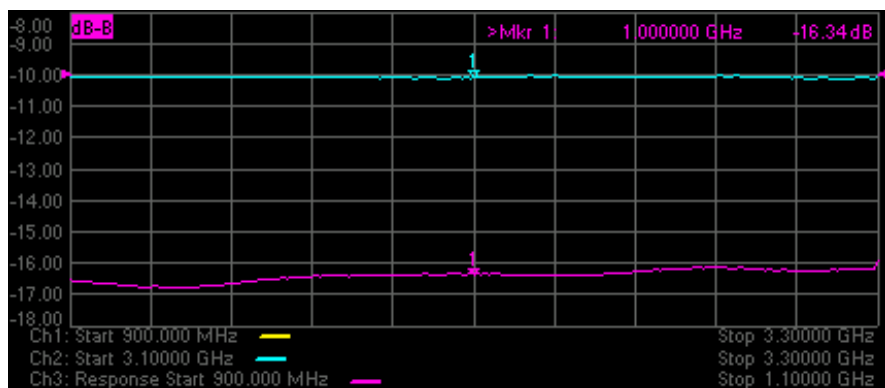
1. Connect the mixer.
2. Adjust [scale](#) to suit your needs.
3. Enable [markers](#) to read power levels for each trace.

The display below shows:

- Ch3 B receiver (bottom trace) absolute output power.
- Ch2 R1 receiver measurement (top trace) absolute input power to the mixer.

With this method, the conversion loss math (B/R1) can be performed with [Equation Editor](#) (not shown). The B/R1 ratio measurement is not supported with receiver power Cal turned on. However, conversion loss (C21) measurements can be made directly and are much easier using the Frequency Converter Application, FCA (Opt 083).





## Conversion Compression

---

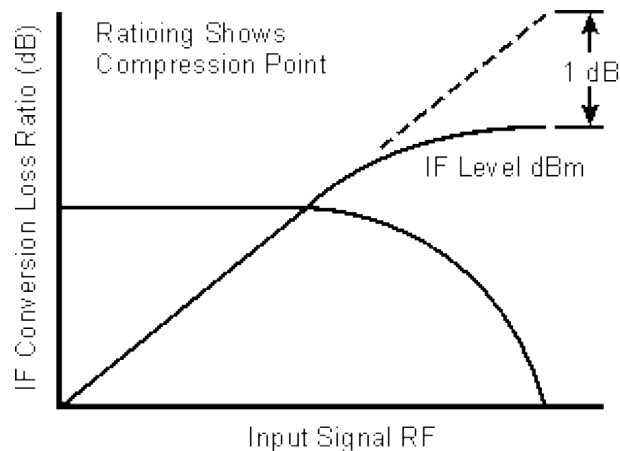
- [What is Conversion Compression?](#)
- [Why Measure Conversion Compression?](#)
- [How to Measure Conversion Compression](#)
- [Measurement Accuracy Considerations](#)

[See other Frequency Converting Device Measurements](#)

### What is Conversion Compression?

Conversion compression is a measure of the maximum input signal level for which a mixer will produce linear operation. It is very similar to the [gain compression](#) experienced in amplifiers.

To understand conversion compression, you must first understand [conversion loss](#). This is the ratio of the mixer output level to the mixer input level. This value remains constant over a specified input power range. When the input power level exceeds a certain maximum level, the constant ratio between input and output power levels begins to change. The point at which the ratio has decreased 1 dB is called the 1-dB compression point. This is illustrated in the graphic below.



### Why Measure Conversion Compression?

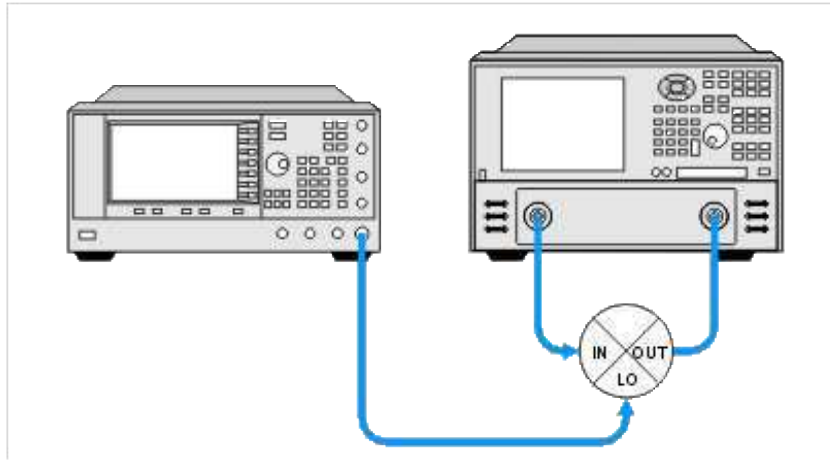
Conversion compression is an indicator of the dynamic range of a device. Dynamic range is generally defined as the difference between the noise floor and the 1-dB compression point.

### How to Measure Conversion Compression

The equipment and setup used to measure conversion compression are essentially the same as for measuring

conversion loss and is illustrated in the following graphic.

The PNA performs a power sweep using [frequency-offset mode](#) and the resulting display shows the mixer's output power as a function of its input power. The 1-dB compression point (or others such as 3-dB) can be determined using markers.



## Measurement Accuracy Considerations

### Equipment Setup Considerations

- The couplers in the PNA have very good directivity. If the return loss of the DUT is bad, the reflected signal gets sampled by the PNA and can result in errors. This relates to error in DUT gain. To increase the accuracy, an attenuator can be added between the PNA's source port and the DUT's input port. Normally a 6- to 10-dB attenuator is sufficient. Addition of this attenuator, however, decreases the available drive to the DUT.
- With high drive levels the PNA can be driven into compression resulting in measurement error. With excessive drive levels, the PNA can be damaged. Add an attenuator between the output of the DUT and the receiver input of the PNA to avoid these problems.

### Calibration Considerations

- [Source power calibration](#) can be used to provide a high level of accuracy for this measurement.
- If your PNA has the [Frequency Converter Application](#) (option 083), you can perform a [Scalar Mixer Calibration](#) to obtain a more accurate measurement.

## Isolation Measurements of Frequency Converting Devices

---

- [What is Isolation?](#)
- [Why Measure Isolation?](#)
- [How to Measure Isolation](#)

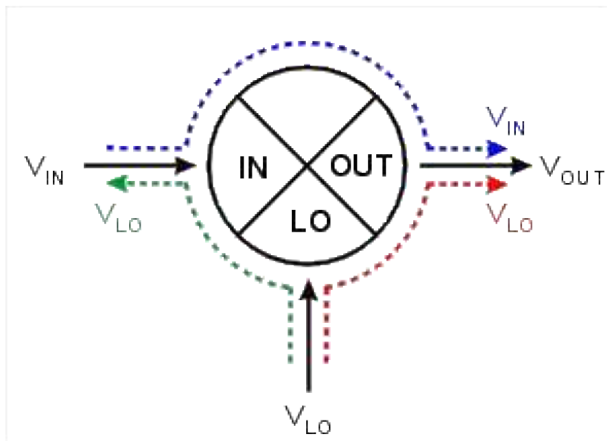
[See other Frequency Converting Device Measurements](#)

### What is Isolation?

Isolation is a measure of the leakage, or feedthrough, from one port to another. The more isolation a mixer provides, the lower the amount of feedthrough. Isolation is measured at the same frequency as the stimulus, not the converted or shifted frequency. Therefore, Frequency Offset capability is not necessary for these measurements.

Three main isolation terms are of interest for mixer measurements:

- LO-to-OUT isolation ( $V_{LO}$ )
- LO-to-IN isolation ( $V_{LO}$ )
- IN-to-OUT feedthrough ( $V_{IN}$ )



### Why Measure Isolation?

Any unwanted signal "leaking" through the device will mix with the desired output signal creating intermodulation products, adding to intermodulation distortion. These unwanted signals may be difficult to filter out.

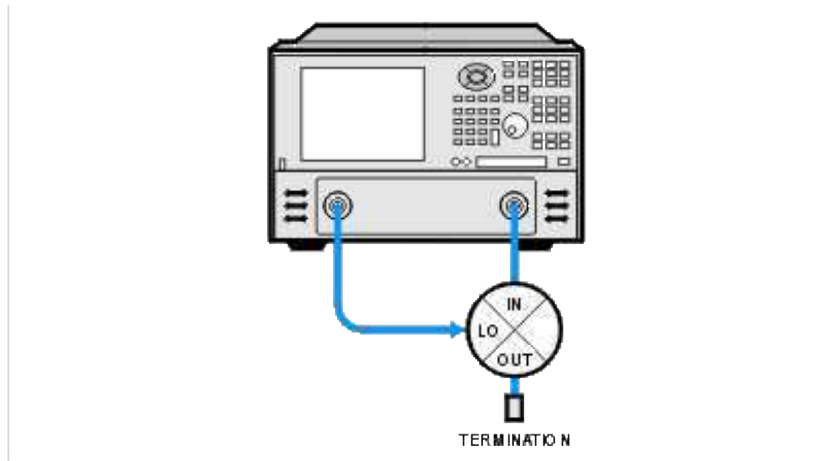
### How to Measure Isolation

Use the following setups to measure the isolation of a mixer:

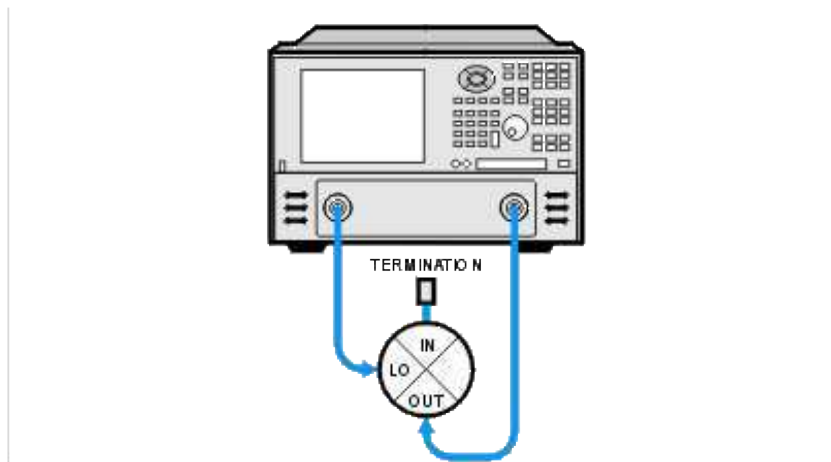
Note the following:

- The Input to Output isolation is very dependent on the LO power level. Isolation should be measured with the LO power at its normal operating level.
- Each of the ports not being tested should be terminated with an impedance typical of actual operation. This may not always be the characteristic impedance,  $Z_0$  (usually 50 or 75 ohms). For example, if the OUT port of a mixer is intended to be directly connected to a filter, then this filter should be used when measuring the LO-to-IN feedthrough.

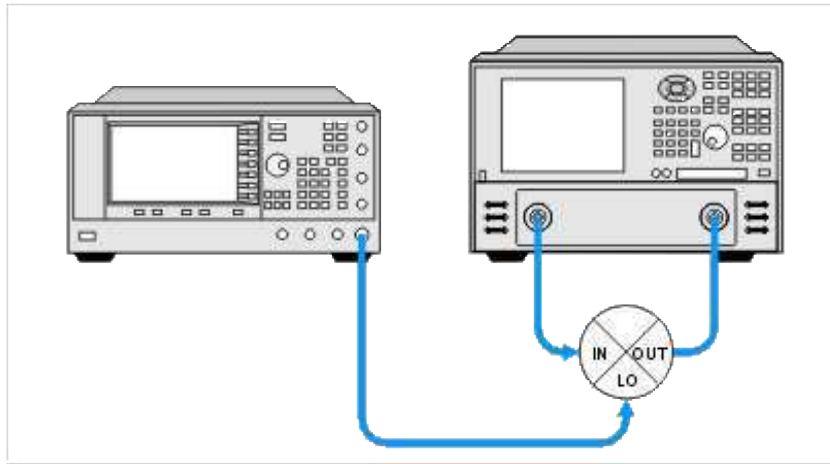
#### LO-TO-IN ISOLATION



#### LO-TO-OUT ISOLATION



#### IN-TO-OUT ISOLATION



### Measuring Converters vs. Mixers

Measuring IN-to-OUT feedthrough of a converter is identical to that of a mixer. The IN-to-OUT feedthrough is generally very small for a converter due to the inclusion of an IF filter in the device. Because of this, the measurement may require the PNA to have increased [dynamic range](#).

Measuring LO leakage (LO-to-OUT and LO-to-IN) of a converter requires a different technique because the LO port is typically not accessible:

- The PNA can be tuned to the frequency of the LO signal and either the OUT or IN port connected to the PNA receiver port. The PNA source port is not connected.
- A spectrum analyzer can be connected to either the OUT or IN port and tuned to the frequency of the LO signal.

## Harmonic Distortion

---

[What is Harmonic Distortion?](#)

[Why Measure Harmonic Distortion?](#)

[How to Measure Harmonic Distortion](#)

[Measurement and Accuracy Considerations](#)

[See other Frequency Converting Device Measurements](#)

### What is Harmonic Distortion?

Harmonics are multiples of any signal appearing at the mixer input and also multiples of the LO input. The distortion of the mixer's output characteristics caused by these harmonics is referred to as harmonic distortion. Harmonic distortion is caused by non-linearities in the device.

Harmonics are NOT signals created by two or more signals interacting (mixing); these signals are known as intermodulation products, which result in intermodulation distortion.

### Why Measure Harmonic Distortion?

- It can degrade the performance of devices connected to the output of the mixer.
- The harmonics can also mix with other signals present in the mixer, adding to the intermodulation distortion of the mixer.

### How to measure Harmonic Distortion

The harmonics can be measured using the PNA with [Frequency Offset](#) (option 80). The frequency of the LO to the mixer is set to zero and multiplier of the RF input is used to set the IF frequency (the harmonic). The equipment setup is shown below.

Since harmonics are specified in dBc, the fundamental RF and both the second and third harmonics are measured and the differences calculated. Multiple channels can be used to do this.

1. Connect the equipment.
2. Setup the measurement for calibration. See also [Measurement and Accuracy Considerations](#).  
Use three channels and [frequency offset mode](#):  
Channel 1 = F1 to F2  
Channel 2 = F1 to 2F2 (frequency offset mode, multiplier = 1)  
Channel 3 = F1 to 3F2 (frequency offset mode, multiplier = 1)

- Perform a source power calibration and receiver power calibration over the entire frequency range. See [Measurement and Accuracy Considerations](#).
- Reduce the frequency span and increase the frequency offset multiplier on Channels 2 and 3:  
Channel 2 = F1 to F2 (frequency offset mode, multiplier = 2)  
Channel 3 = F1 to F2 (frequency offset mode, multiplier = 3)  
**Note:** Because the frequency span has been changed from that used for calibration, the source and receiver calibrations will be interpolated.
- Connect the DUT, make the measurement, and calculate the harmonic response:  
Set up markers on Channels 1, 2 and 3, and determine the difference between the marker values to get the dBc value of each harmonic.  
Channel 1 - Channel 2 = 2nd harmonic (dBc)  
Channel 1 - Channel 3 = 3rd harmonic (dBc)  
**Note:** Be sure to set the markers to the appropriate stimulus. Channel 2 markers should be set to twice the frequency of Channel 1 markers. Channel 3 markers should be set to three times the frequency of Channel 1 markers.

## Measurement and Accuracy Considerations

### Equipment Setup Considerations

- A filter must be used at the input of the mixer to remove the PNA source harmonics.

### Calibrations

- If your PNA has the [Frequency Converter Application \(FCA\)](#), you can perform a [Scalar Mixer Calibration](#) to obtain a more accurate measurement.



## Return Loss and VSWR

---

- [What are Return Loss and VSWR?](#)
- [Why Measure Return Loss and VSWR?](#)
- [How to Measure Return Loss and VSWR](#)

[See other Frequency Converting Device Measurements](#)

### What is Return Loss and VSWR?

Return loss and VSWR are both linear reflection measurements, even when testing frequency conversion devices, because the reflected frequency is not converted. These measurements are essentially the same as for filters and amplifiers. Learn more about [Reflection Measurements](#).

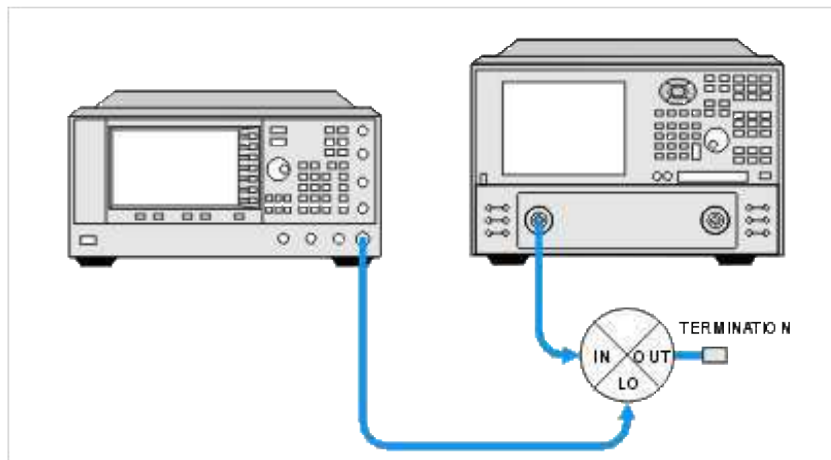
### Why Measure Return Loss and VSWR?

Devices which have poor return loss and VSWR result in loss of signal power or degradation of signal information.

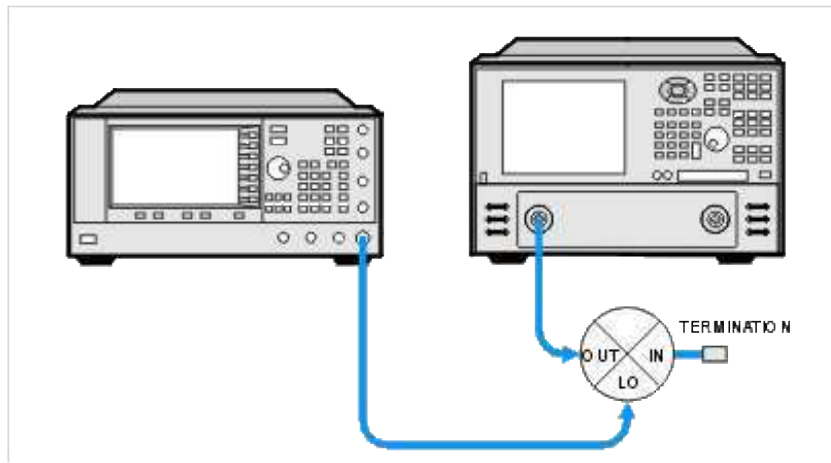
### How to Measure Return Loss and VSWR

Setup the PNA measure return loss and VSWR as you would any two-port device. Connect your frequency converting device as shown in the following diagrams:

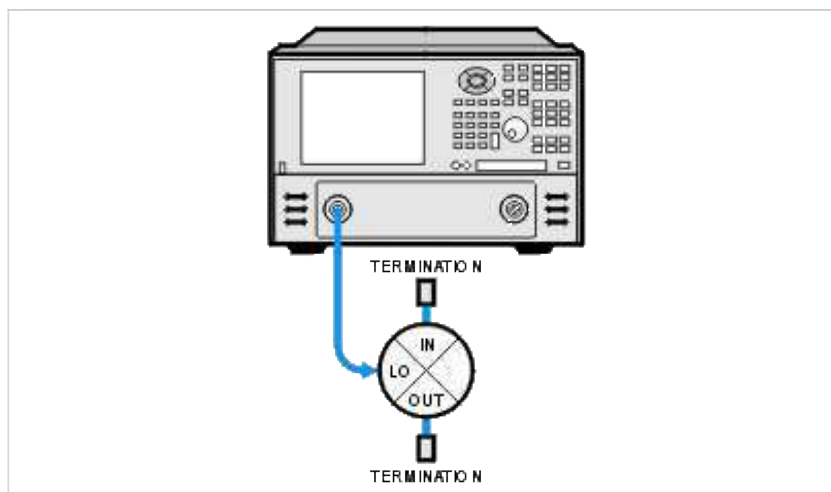
**RETURN LOSS AND VSWR OF MIXER INPUT PORT**



**RETURN LOSS AND VSWR OF MIXER OUTPUT PORT**



### RETURN LOSS AND VSWR OF MIXER LO PORT



## FIFO (Opt 118) and other Antenna Features

---

The following features, used with the PNA-X models and the N5264A, were designed specifically for Antenna applications.

- [Fast Sweep Features \(Opt 118\)](#)
    - [FIFO Buffer](#) and [Fast CW](#)
    - [FIFO Buffer and Fast Groups](#)
    - [FIFO Buffer and Fast Segments](#)
  - [Other Useful Antenna Features](#)
    - [Point Averaging](#)
    - [Point Sweep](#)
    - [Trace Triggering](#)
  - **See Also**
    - [N5264A](#)
    - [Pulsed Measurements](#)
    - [Frequency \(Security\) Blanking](#)
    - [External Triggering](#)
- 

### Fast Sweep Features (Opt 118)

The following features allow you to **very quickly** measure and download data to a remote computer.

- [Fast CW](#), [Fast Groups](#), and [Fast CW Segments](#) all work ONLY with the FIFO Data Buffer.
- These features can be used ONLY with [SCPI or COM commands](#). COM is faster than SCPI when using [DataInCompactForm](#). Otherwise, SCPI is faster than COM.
- These features are all demonstrated in the following example program:

Download and install the **FIFO Tester** - a C# program that uses either the COM or SCPI interface to download data to, and read data from, the FIFO buffer. Source code is also provided. See <http://na.tm.agilent.com/pna/apps/applications.htm>

## The FIFO Data Buffer

The FIFO (First-IN, First-OUT) data buffer is a circular buffer that allows very fast Read-Write access.

- When enabled, all the data gathered is placed into a 4 GB FIFO buffer.
- You can write to, and simultaneously read from, the FIFO buffer.
- A maximum of 1 million data points can be read for each query.
- A preset or instrument state recall will turn off the FIFO buffer collection.
- When more than one measurement is present, data from each measurement is stored in the FIFO buffer in the following order. These measurements are separated into lines for easier reading.
  - R, A, B, C, D,
  - R/R, A/R, B/R, C/R, D/R,
  - R/A, A/A, B/A, C/A, D/A,
  - R/B, A/B, B/B, C/B, D/B,
  - R/C, A/C, B/C, C/C, D/C,
  - R/D, A/D, B/D, C/D, D/D

## Fast CW

In Fast CW mode the PNA display is not updated. There is no background computation or other 'interference' from the PNA computer. Therefore, data is acquired real-time.

The following **requirements** must be met **before** sending the Fast CW command.

- FIFO is ON
- A single channel is being measured. Other channels can be in Hold.
- All measurements are acquired in a single sweep.

### Notes:

- See example programs in [SCPI](#) and [COM](#).
- Fast CW sets the number of data points, overwriting the standard channel setting.
- When exiting Fast CW, the FIFO data buffer is cleared.
- External trigger signals are allowed only through the rear-panel [Trig In connector](#) - NOT the Aux1 and Aux2 In connectors.
- An error message appears if triggering is sent to the PNA faster than it can respond.

### Fast Groups with FIFO Data Buffer

With this speed optimization feature, interaction with Windows or other PNA 'overhead' calls are suspended, allowing very fast and predictable measurement timing.

Fast Groups is automatically enabled when the following **requirements** are met:

- FIFO is ON
- A single channel is being measured. Other channels can be in Hold.
- All measurements are acquired in a single sweep.
- [Group trigger is enabled](#) with count > 1.

#### Notes:

- Fast CW can **NOT** be used with Fast Groups.
- Fast Groups and Fast CW Segments were designed to be used together, but not required.
- The [FIFO Tester](#) example program demonstrates this feature.

### Fast CW Segments with FIFO Buffer

In this optimization feature, each CW segment (where the start and stop frequency is identical) within a channel is measured at speeds as fast as the Fast CW mode sweep.

Fast CW Segments is automatically enabled when the following **requirements** are met:

- FIFO is ON
- Start and stop frequency of a segment is identical.
- External trigger signals are allowed only through the rear-panel [Trig In connector](#) - NOT the Aux1 and Aux2 In connectors.

#### Notes:

- Fast CW can **NOT** be used with Fast CW Segments.
- The sweep can include non-CW segments, but these are not acquired in Fast mode.
- Fast Groups and Fast CW Segments were designed to be used together, but not required.
- The [FIFO Tester](#) example program demonstrates this feature.
- In Fast CW Segments, when data is not being acquired in real-time, the following message appears:  
**Caution: Sweep time jitter. Try reducing the number of segments.** To avoid this error, reduce the number of segments in the channel.

## Other Antenna Features

### Point Averaging

This feature is selected on the [Average](#) dialog.

When selected, each data point is measured the specified number of averages before stepping to the next data point. When [point trigger](#) is selected, only one trigger is required for each data point regardless of the number of averages.

### Point Sweep

This feature is selected on the [Sweep Setup](#) dialog.

In Point Sweep mode, the PNA measures both the forward and reverse parameters at each frequency point before stepping to the next frequency. The display trace is updated as each data point is measured. Point sweep is the same as stepped sweep mode of the 8510 and 8530.

### Trace Triggering

This feature is selected under [Trigger Mode](#) on the Trigger dialog.

Available ONLY when [Point Sweep](#) is selected. Each trigger signal causes two identical measurements to be triggered separately - one trigger signal is required for each measurement. Other trigger mode settings cause two identical parameters to be measured simultaneously.

Trace triggering is NOT permitted when a channel is using a 2 port (or more) S-Parameter calibration.

## Other Useful Antenna Features

- [Pulsed Measurements](#)
- [Frequency \(Security\) Blanking](#)
- [External Triggering](#)

---

Last Modified:

6-Mar-2009    MX New topic

## Gain Compression Application GCA (Opt 086)

---

- [What's New](#)
- [Features, Requirements, and Limitations](#)
- [Gain Compression Concepts](#)
- [Understanding the GCA Displayed Traces](#)
- [Gain Compression Parameters](#)
- [Compression Methods](#)
- [Acquisition Modes](#)
- [Using Gain Compression App](#)
  - [Frequency tab](#)
  - [Power tab](#)
  - [Compression tab](#)
  - [Safe Sweep Mode dialog](#)
- [Compression Analysis](#)
- [Saving GCA Data](#)
- [GCA Measurement Tips](#)
- [Macros](#)

### See Also

- [GCA Calibration](#)
- [Programming commands](#)
- **App Note** [Amplifier Linear and Gain Compression Measurements](#)

### Other PNA Applications

## Features, Requirements, and Limitations

### Features

- Fast, easy, and complete Gain Compression measurements for amplifiers.
- Many [compression parameters](#) to choose from, including gain, input power at compression, output power at compression, input match, and compression level.
- Several [compression methods](#) to choose from, including deviation from linear gain, deviation from max gain, back-off, and X/Y, and compression from saturation.
- Three [acquisition methods](#) to choose from: Power per Freq, Freq per Power, and SMART Sweep
- [SMARTCal Calibration Wizard](#) to guide you through Full 2-Port or Enhanced Response calibration, plus Source Power calibration.
- [Compression Analysis](#) allows traditional power sweep view at a selected frequency.
- [Receiver Leveling](#) provides continuous source power accuracy.
- Supports Wideband and Narrowband Pulse measurements using the new integrated [Pulse setup dialogs](#).

## Requirements

- PNA-X with Opt 086 (software option only) [must be enabled](#).
- When performing an optional calibration:
  - ECal module or Calibration Kit
  - Power meter and power sensor

## Limitations with GCA

- Number of points limited to 20,001 for two-dimensional acquisitions, 10,000 points for SMART Sweep.
- Standard CW power sweep is NOT supported in a Gain Compression channel.
- No Independent IFBW, Independent power levels in segment table.
- Stepped sweep mode only.
- Linear, Log, and Segment frequency sweep modes only.
- 2-port DUTs ONLY.

The following PNA Features are **NOT** Available in a Gain Compression channel:

- [Unratioed receiver measurements](#) (A, B, R)
- [ECal User Characterization](#)
- Some [Fixturing Features](#)



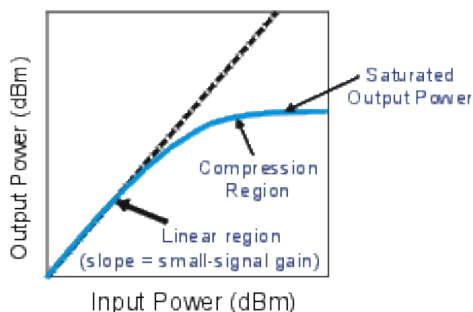
- [FOM](#) or [FCA](#)
- [External Test Set Control](#) (Option 551)
- [Interface Control](#)
- [Copy Channel](#)
- [IF Path Configuration](#)
- [Time Domain](#)
- [Balanced measurements](#)
- [Point and Sweep trigger](#)
- Save [Auto Formatted Citifile](#) data.

## Gain Compression Application Concepts

### What is Gain Compression

An amplifier has a region of linear gain, where the gain is independent of the input power level. This gain is commonly referred to as small signal gain. As the input power is increased to a level that causes the amplifier to approach saturation, the gain will decrease. The 1 dB gain compression is defined as the input power level that causes amplifier gain to drop 1 dB relative to the linear gain.

You can quickly measure the gain compression using a [compression marker](#) on a power sweep trace.



### Terms used in GCA

**Linear Power Level** The specified input power that yields linear gain (also known as 'small-signal gain') in the amplifier.

**Reference gain** The measured gain that is used as a reference for determining compression level. The [Compression Method](#) that is used could cause this value to be different.

**Compression level** The specified amount of gain reduction from the reference gain.

**Target gain** The gain at the specified compression level. Although this term does not appear in GCA, it is important to understand when discussing the various compression parameters.

For example, when using [Compression from Linear Gain](#) method with the following settings:

- Linear gain (measured at Linear Input power) = 10.2 dB

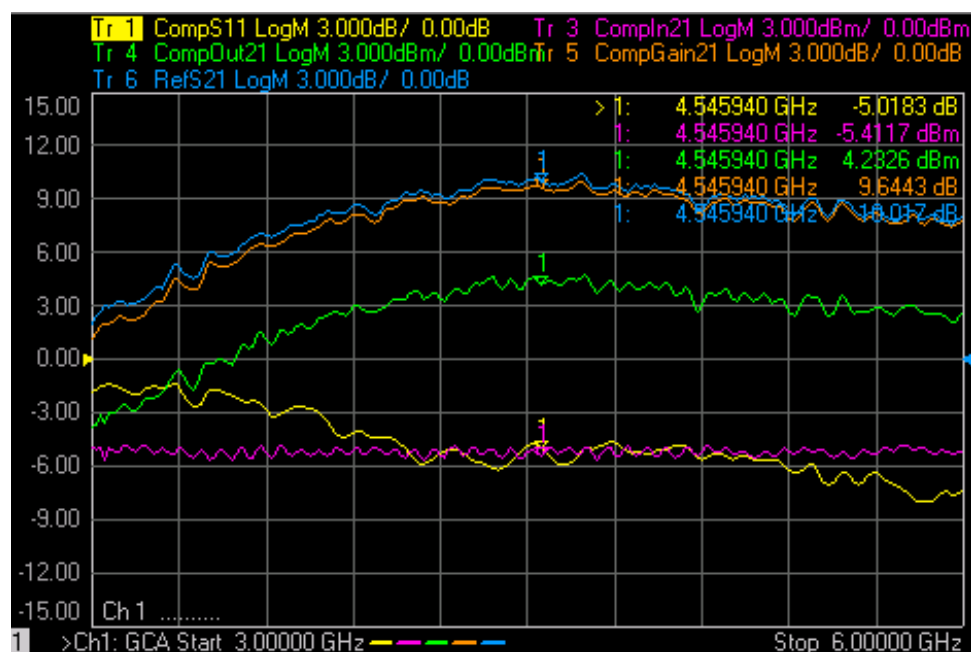
- Compression level (specified) = 1 dB
- Target gain = 9.2 dB

This is called 'Target' gain because GCA will search for the closest measured gain to 9.2000 dB. It may not measure this gain exactly.

**Compression point** The operating point at which the measured gain is closest to the Target Gain. All [compression parameters](#) report data for this operating point.

## Understanding the GCA Displayed Traces

One of the most important concepts to remember with GCA is that, each frequency data point represents many measurements using different input power levels.



Some things to notice about how GCA displays **compression** data:

1. The X-axis values are ALWAYS frequency. Imagine behind each frequency data point, a traditional power sweep curve with corresponding measurements and calculations to find the specified compression point.
2. The Y-axis values are always reported at the [compression point](#). The value that is displayed depends on the **compression** parameter that you choose. The [S-parameters](#) that are displayed in a GCA channel are always measured at the [linear and reverse](#) power level.

**Example:** Five of the six GCA [compression parameters](#) are displayed in the above image. The missing trace, [DeltaGain21](#) is discussed below.

- Markers are placed at 4.549 GHz for all of the parameters.
- **Tr 3 Compln21** (Input power at the compression point) shows the marker value to be **-5.4117 dBm**. This is the power into the DUT that was required to achieve the compression point. Notice that this is about the

same input power required to achieve the specified compression at ALL frequencies.

- **Tr 5 CompGain21** (Gain at the compression point) shows the marker value **9..6443 dB** . This is the measured gain at the compression point.
- To see the gain at a different input power at this frequency, use the [Compression Analysis](#) feature.

## Gain Compression Parameters

There are several Gain Compression parameters, as well as standard S-parameters and ADC parameters, that can be measured in a GCA channel.

### How to add or change GCA Parameters

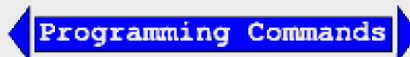
First, create a GCA channel. [Learn how](#). The default parameter is S21.

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then select a parameter

#### Using a mouse with PNA Menus

1. click **Response**
2. then **Measure**
3. then select a parameter



## Linear S-Parameters

For convenience, the standard S-parameters are offered in a GCA channel. S11 and S21 are measured at the specified Linear Input level. S22 and S12 are measured at the specified Reverse power level.

Parameter	Description	When Measured
S11	Input Match	Always
S21	Gain	Always
S22	Output Match	See <a href="#">Reverse</a>
S12	Reverse Isolation	See <a href="#">Reverse</a>

## ADC Parameters

Four ADC analog-to-digital converter measurements are offered in a GCA channel:

- **AI1** and **AI2** are measured at the specified Linear Input level.
- **CompAI1** and **CompAI2** are measured at the specified compression point.

These DC measurements, along with [Equation Editor](#), allow you to make PAE measurements at the Linear Input level and compression point.

These measurements are made at pins 7 and 8 of the PNA-X [Power IO connector](#).

[Learn more about ADC measurements.](#)

## Compression Parameters

**Note:** The following table assumes: DUT **Input** = PNA **port 1** and DUT **Output** = PNA **port 2**.

When the Port mapping is different, the parameters in GCA are updated accordingly. For example, with Input = port 2 and Output = port 1, then "Compln12" would be displayed.

The raw data for these parameters are always measured.

Parameter	Description
Compln21	Input power at the compression point.
CompOut21	Output power at the compression point.
CompGain21	Gain at the compression point.
CompS11	Input Match at the compression point.
RefS21	Linear Gain value used to calculate the compression level. This is calculated differently depending on the compression method.
DeltaGain21	CompGain21 MINUS Linear Gain (in Log Mag format). This trace can be used to learn a lot about the DUT compression point. <a href="#">Learn more.</a>

## Compression Methods

GCA offers the following methods to find the compression point of an amplifier using GCA:

### Compression from Linear Gain

The Reference Gain is measured using the specified Linear (Input) Power Level. The Target Gain is calculated as the Linear Gain minus the specified Compression Level. For example 8.3 dB - 1 dB = 7.3 dB.

### Compression from Max Gain

Available in [SMART Sweep](#) and [2D Acquisition modes](#).

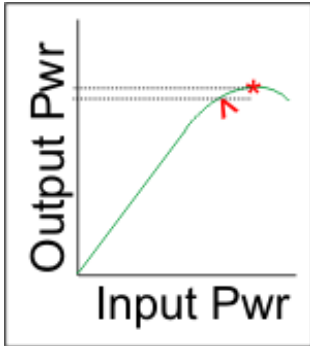
The linear region of an amplifier gain may not be perfectly linear. The highest gain value that is found at each frequency is used as the Reference (S21) Gain. The [Target Gain](#) is found in the same way as Compression from Linear Gain.

### Compression from Saturation

Available in [SMART Sweep](#) and [2D Acquisition modes](#).

This method is used to better find the compression point when measuring amplifiers with non-linear gain as

shown in the following image:



The Max power out value \* is found at each frequency. Then input power is lowered until the output power decreases by the specified 'From Max Pout' value. This is the compression point. ^

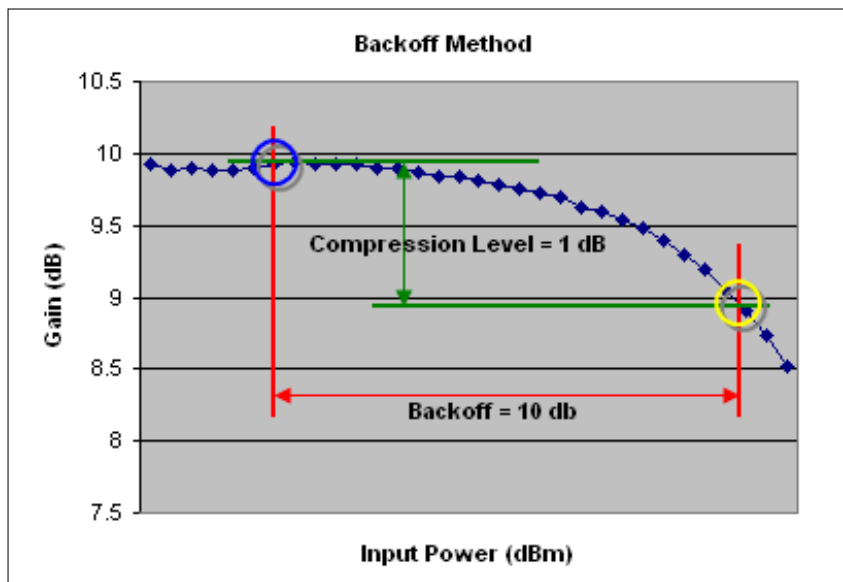
### Backoff and X/Y method

These two compression methods are very similar.

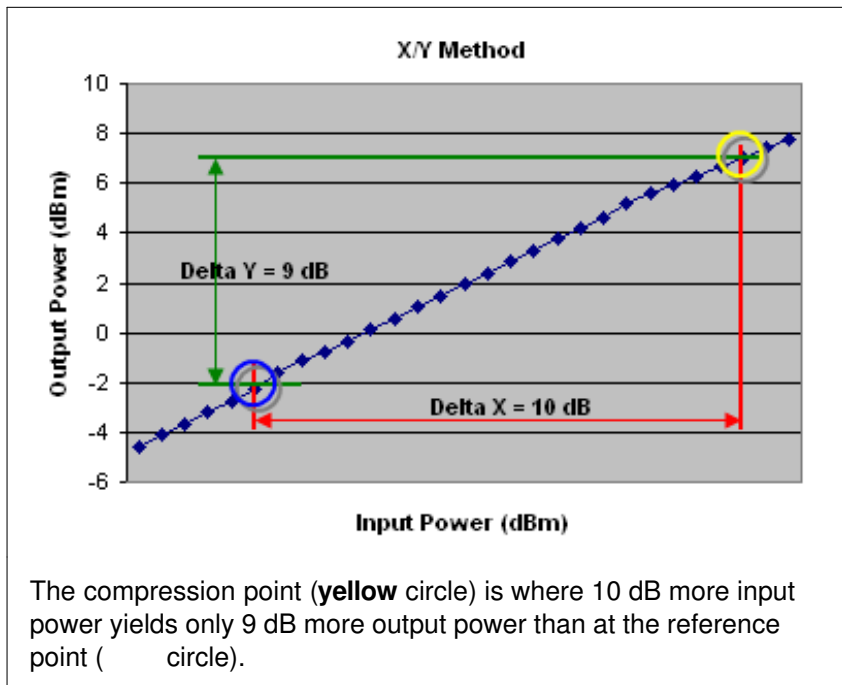
- Both methods specify a difference in input power (X axis) between the linear region and compression point.
- For the Y-axis difference:
  - **Backoff method** specifies Compression Level which is a difference in **Gain**.
  - **X/Y method** specifies Delta Y which is a difference in **Output Power**.

GCA searches for these points differently for [2D sweeps](#) and [SMART sweep](#).

The following images show how Backoff and X/Y method is calculated at ONE frequency.



The compression point (**yellow** circle) is where 10 dB more input power yields 1 dB less gain than at the reference point (circle).



## Acquisition Modes

The GCA offers three modes for data acquisition: Two 2D sweep modes, and [SMART sweep](#).

To see a traditional power sweep at a single frequency, use the Compression Analysis feature. [Learn more.](#)

### 2D (two-dimensional) Sweeps

This is the easiest method to understand, and the least efficient for finding the compression point. Both 2D sweep modes work as follows:

1. All GCA measurements begin by measuring S-parameters at the specified Linear Power level. Reverse parameters are measured ONLY if Full 2-port calibration is applied or if a reverse parameter is displayed. [Learn more about Cal choices.](#)
2. Gain measurements are then made at ALL of the specified frequency and power values. Although these are conceptually 2-Dimensional sweeps, a single sweep is constructed in firmware. [See Data Points Limit.](#)
3. After data has been measured, a search is performed to find the compression point. You can choose to interpolate between the two measured points closest to the target gain. [Learn more.](#)

As each sweep is performed, dots are plotted next to the **Ch** indicator in the lower left corner of the display to indicate progress for the current sweep.

**Note:** For [Backoff and X/Y compression method](#), GCA does not verify that the specified Start - Stop power range is

at least the size of the specified Backoff or X value. The closest compression point is always reported.

## 2D Sweep Modes

- **2D Sweep Power per Frequency** - Input power is stepped from [Start to Stop](#) at each specified frequency. From the following example you can see that the device is exposed to the highest power level (p3) at the first frequency (f1). This could heat the device early in the measurement and affect compression results.

The following examples show (frequency, power) values for three frequency points and three power points, resulting in a total of 9 measurements:

1	2	3	4	5	6	7	8	9
f1,p1	f1,p2	f1,p3	f2,p1	f2,p2	f2,p3	f3,p1	f3,p2	f3,p3

- **2D Sweep Frequency per Power** - Frequency is swept from start to stop at each specified power level as follows:

1	2	3	4	5	6	7	8	9
f1,p1	f2,p1	f3,p1	f1,p2	f2,p2	f3,p2	f1,p3	f2,p3	f3,p3

## Viewing and Saving 2D Data

It is NOT possible to plot ALL of the 2D measurement data on the PNA display. However, it can be saved to a \*.csv file and then read into an Excel spreadsheet. The initial S-parameter measurement data is not saved to this file. [Learn more.](#)

You can also view on the PNA all power sweep information at a selected frequency using the [Compression Analysis](#) feature.

## SMART Sweep

SMART Sweep is usually the fastest and most accurate method to measure Gain Compression. Unlike the 2D acquisition modes which measure all of the specified frequency / power points, SMART Sweep performs a series of power search iterations. At each frequency, an 'intelligent guess' of input power is made to find the compression level that is within tolerance. This guess is further refined with each successive power search iteration sweep.

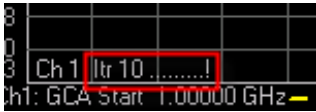
SMART Sweep continues to iterate until one of the following conditions occur:

1. ALL data points are within tolerance. When the compression level for a data point achieves the specified tolerance, it continues to be measured and input power changed to improve the measurement within tolerance.
2. The specified compression level can NOT be achieved for the remaining frequencies that are not in tolerance. Either the Start power is too high or the Stop power is too low.
3. Maximum iterations have been achieved. If a measured gain is not within the specified tolerance before the specified **Max** number of **Iterations** has been reached, then the **last** power reading is used as the compression point.

### The Iteration Counter, Dots, and Bangs

Next to the Ch indicator, in the lower left corner of a GCA window, the following annotation appears:

- An **iteration counter** is incremented each time input power is adjusted.
- A **dot** appears when another 10% of the frequency points are within tolerance.
- **!** (bangs) are displayed after the last iteration. Each bang represents 10% of the data points that are NOT within tolerance.



### SMART Sweep and Compression Method

The intelligent guess process works differently depending on the compression method. This is important because Backoff and X/Y compression methods subject the DUT to significant changes in input power during an iteration sweep. This can affect the DUT and the measurement results.

[Learn all about Backoff and X/Y compression methods.](#)

ALL GCA measurements begin by measuring S-parameters at the specified Linear Power level. Reverse parameters are measured ONLY if Full 2-port calibration is applied or if a reverse parameter is displayed. [Learn more about Cal choices.](#)

- **Backoff and XY** Because both compression methods specify the separation between the "linear" region and the "compressed" region, each iteration requires a single sweep at **two dramatically** different power levels over the same frequency range. The first half of the sweep measures the DUT at the **Backoff** or **X** power level. The second half of the sweep measures the DUT at the compressed power level, specified by the [Start and Stop](#) power range. At the beginning of the second half, the power level rises by the **Backoff** or **X** value. The specified [Settling Time](#) is applied at this point to allow the DUT time to react to this significant change in power level. [Safe Sweep](#) does **NOT** minimize this change in input power. However, Safe Sweep with Backoff and XY methods **DOES** prevent the DUT from being exposed to too much input power.
- **Compression From Linear Gain** After the reference gain is measured at the linear input power, the next iteration measures the DUT at a higher power level which attempts to push the DUT well into compression. Subsequent sweeps, depending upon the compression level of the DUT, either increases or decreases the power in order to reach the desired compression level. Usually, by the third iteration sweep, a curve-fit algorithm is utilized to precisely find the compression point.

**Note:** The DUT can be subject to significant changes in power from one iteration sweep to the next. This can be minimized by the use of [SAFE Sweep](#) and careful selection of the corresponding settings.

- **Compression from Max Gain** The maximum gain that is found at each frequency is stored and used to calculate the compression point. SMART Sweep does NOT perform extra iterations to search for the maximum possible gain of the amplifier at each frequency.
- **Compression from Saturation** The maximum power out that is found at each frequency is stored and used to calculate the compression point. SMART Sweep does NOT perform extra iterations to search for the maximum possible power out of the amplifier at each frequency.



## Using the Gain Compression Application

The following is a general procedure for performing a GCA measurement. The challenge with GCA is configuring a measurement that yields the true compression performance of YOUR DUT. This requires knowledge of the Gain Compression settings and knowledge of the DUT.

See specific dialog boxes below.

1. Disconnect the DUT if preset or default power levels may damage the PNA or DUT.
2. [Preset](#) the PNA, or configure a suitable [User Preset](#) that will be safe in case the DUT is connected.
3. Create a GCA channel. [Learn how](#). The default trace is S21.
4. Start [GCA Setup dialog](#) and configure the measurement settings based on the DUT, adapters, attenuators, booster amplifiers, and fixtures to be used in the measurement.
5. Save the [instrument state](#) (optional).
6. Connect DUT and apply bias and RF power as appropriate. The default measurement for a GCA channel is S21 (amplifier gain). Inspect the gain measurement to ensure the DUT is operating as expected.
7. Add GCA compression parameter traces. [Learn how](#).
8. Adjust the measurement settings to yield satisfactory compression parameters. [See GCA Measurement Tips](#).
9. Start and complete the [GCA Calibration wizard](#).

### How to start the Gain Compression Setup dialog

To provide quicker access, use the Setup softkey. [Learn how](#).

#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[Gain Compression Setup]**

#### Using a mouse with PNA Menus

1. Click **Stimulus**
2. then **Frequency**
3. then **Gain Compression Setup**

◀ **Programming Commands** ▶

The screenshot shows the 'Frequency' tab of a dialog box. It has three main sections: 'Sweep Type' with radio buttons for 'Linear Sweep' (selected), 'Log Sweep', and 'Segment Sweep'; 'Data Acquisition Mode' with radio buttons for 'SMART Sweep' (selected), 'Sweep Power Per Frequency (2D)', and 'Sweep Frequency Per Power (2D)'; and 'Sweep Settings' with input fields for 'Number Of Points' (201), 'IF Bandwidth' (10.000 kHz), 'Start' (1.000000000 GHz), 'Stop' (9.000000000 GHz), 'Center' (5.000000000 GHz), and 'Span' (8.000000000 GHz). On the right, summary statistics are displayed: 'Total Number of Points: 5829(20001)', 'Number of Power Points: 29', and 'Compression Method: X/Y Compression'.

## Frequency tab - Gain Compression -dialog box help

Configures the frequency settings over which Gain compression is to be measured, as well as the measurement method.

### Sweep Type

Choose a method in which to sweep frequency: Linear, Log, and Segment Sweeps. This setting applies to all data acquisition modes.

#### Segment Sweep Notes:

- The segment table shown on the dialog is **'READ-ONLY'**.
- Learn how to [Create and edit the Segment Sweep table](#).
- **Independent IFBW** and **Power** are NOT available.
- [X-axis point spacing](#) is available beginning with A.09.10.

**CW sweep** is NOT available. However, to see a traditional power sweep at a single frequency, use the [Compression Analysis](#) feature.

### Data Acquisition Mode

Specifies HOW the gain compression data is collected.

#### SMART Sweep

- At each frequency, input power is 'intelligently' adjusted to find a measured gain equal to the target gain.
- Faster and more accurate than 2D sweeps to measure Gain Compression point at a number of frequencies.
- [Learn ALL about SMART Sweep](#)

#### 2D (two-dimensional) Sweeps

- **Sweep Power per Frequency** Performs a series of power sweeps at each successive frequency.
- **Sweep Frequency per Power** Performs a series of frequency sweeps at each successive power level.
- [Learn ALL about 2D sweeps](#)

## Sweep Settings

Click each to learn more about these settings.

- **[Number of points](#)** Number of frequency points to measure. The Frequency points may be limited due to the number of specified Power points. [See Data Points Limit](#).
- **[IF Bandwidth](#)** Set this value to yield acceptable trace noise when measuring gain at the linear power level. This level of noise contributes directly to the accuracy of compression point. A lower value (narrower IFBW) allows for more accurate, but slower, measurements. [See GCA Measurement Tips](#) to see how to best set IFBW.
- **[Start / Stop, Center / Span](#)** frequencies. Set the frequency range over which to measure Gain compression.

## Data Points Limit

The maximum number of measurement data points depends on Acquisition method and Compression method as follows:

	SMART sweep	2D sweep
Compression method	Number of <b>frequency points</b> is reduced to ensure the total number of data points does not exceed the specified limit.	Number of <b>power points</b> is reduced to ensure the total number of data points does not exceed the specified limit.
• Compression from linear gain	Data points = <b>freq points</b> Max = 20,001	Data points = <b>(freq. points) * (power points)</b> Max power points = <b>2,001</b> Max data points = <b>20,000</b>
• Compression from max gain	Data points = <b>freq points</b> Max = 20,001	
• X/Y and Backoff	Data points = <b>2 * freq points</b> Max = 20,001	

**Note:** Although the dialog box will allow you to enter any number of frequency or power points, the values are checked when **OK** or **Apply** is pressed. If a limit is exceeded, the relevant data points are reduced to the maximum allowable number **without warning**.

## Power tab - Gain Compression dialog box help

Configures RF power and Power Sweep settings for Gain Compression measurement.

**Power ON (All channels)** Check to turn RF Power ON or clear to turn power OFF for all channels.

### Input Port

**Select** the PNA port that is connected to the DUT Input.

**Linear Power Level** The input power that yields the linear gain of the DUT. The linear gain is used as the reference gain when calculating the **Compression from Linear Gain**. Input match is also measured at this power level.

**Source Attenuator** Specifies the attenuator setting associated with the port connected to the input of the DUT. This attenuator will affect the range of available power into the DUT [Learn more about Source Attenuation](#).

All PNA channels in continuous sweep must have the same attenuation value. [Learn more](#).

**Receiver Attenuator** Specifies the attenuator setting for the receiver associated with the output of the DUT. When the power into the receiver test port is around +10 dBm, the PNA receiver may be in compression. However, with receiver attenuation, lower input power levels may become too noisy to make accurate power measurements. In this case, lower IFBW to reduce noise. [Learn more about Receiver Attenuation](#).

**Source Leveling** Specifies the leveling mode. Choose Internal or Receiver R1. [Learn how to configure Receiver Leveling](#). Open Loop should only be used when doing [Wideband Pulse measurements](#).

### Output Port

**Select** the PNA port that is connected to the DUT Output.

**Reverse Output Power** Sets power level into the output of the DUT for reverse sweeps. Port power is automatically uncoupled.

Reverse power is applied to the DUT ONLY under the following conditions. Otherwise, this setting is ignored.

- When Linear Output Match or Linear Reverse Isolation parameters are requested.

- When Full 2-port correction is used. You can perform a full 2-port cal and downgrade to an Enhanced Response Cal to prevent reverse power from being applied to the DUT. [Learn more.](#)

**Source Attenuator** Specifies the attenuator setting for the port connected to the DUT output. This setting will affect the range of available power at the DUT output port.

**Receiver Attenuator** Specifies the attenuator setting for the receiver associated with the DUT output port.

**Source Leveling** Specifies the leveling mode. Choose from: Internal (normal operation) or Open Loop (used only for [Wideband Pulse measurements](#)).

## Power Sweep

**Power Points** Number of power points to measure for 2D acquisition modes. The Power Points may be limited due to the number of frequency data points. [See Data Points Limit.](#) This setting is NOT available in SMART Sweep, which uses only enough power points to find the specified compression level.

### Start and Stop Power

- **2D sweep** In Backoff, X/Y, and Compression from Max Gain methods, sets the range of power levels that are applied to the DUT to find BOTH the [Reference Gain](#) and [Compression point](#). Make sure this range is wide enough to include both. For example, if the Backoff level is 10 dB, then the power range must be greater than 10dB. Otherwise, GCA will report a compression value using the closest reference gain and compression point, which may be inaccurate. In Compression from Linear Gain, the reference gain is measured at the Linear Power Level, so the Start and Stop power levels are used to find the compression point.
- **SMART sweep** Sets the range of power over which GCA will search for the compression point. The reference gain is found using the Linear Power Level, Backoff, and X values, depending on the Compression Method. To reduce the number of iterations that are required to find the compression point, limit the Start / Stop power range to the input levels that will achieve compression. Do not include the linear region.

**Note:** If your DUT requires more input power to achieve compression below 3.2 GHz, use the PNA-X **Hi-power mode**, available from the [RF Path Configuration](#) dialog. The disadvantage to this is higher harmonic content.

**Power Step (Size)** Calculated value from current Start, Stop, and Points settings. This setting can NOT be changed directly.

**Path Configuration** click to launch the [RF Path Configuration](#) dialog.

## Compression tab - Gain Compression dialog box help

### Compression Method

[Learn ALL about these Compression Methods](#)

- **Compression from Linear Gain** The specified compression level is measured from the linear gain. The linear gain is measured using the **Linear Power Level** that is specified on the [Power tab](#).
- **Compression from Max Gain** The specified compression level is measured from the maximum gain level. In SMART sweep, the Max Gain value is updated as each iteration occurs. To increase the chances of measuring the actual maximum gain of the amplifier, [Safe Sweep](#) should be invoked using low Coarse and Fine increments.
- **Compression from Back Off** This compression method uses the Compression Level and Back Off values for finding the compression point.
- **X/Y Compression** This compression method uses the specified parameters (X and Y) as the criterion for finding the compression point.
- **Compression from Saturation** Similar to Compression from Max Gain, except the specified compression level is measured from the maximum power out level. Use this method to better find the compression point when measuring amplifiers with non-monotonic gain. In SMART sweep, the Max power out value is updated as each iteration occurs. To increase the chances of measuring the actual maximum power out of the amplifier, [Safe Sweep](#) should be invoked using low Coarse and Fine increments

### 2D Sweep - Compression Point Interpolation

Check the box to calculate and display interpolated compression traces.

The [Target gain](#) is calculated using a complex linear ratio between the two closest measured values. All compression parameters are then interpolated using this same ratio.

Clear the box to display compression parameters for the closest compression point, either high or low, to the level specified in the Compression Method setting.

## SMART Sweep

[Learn ALL about Smart Sweep.](#)

**Tolerance** Specifies an acceptable range for measuring the compression level. Reducing this value can significantly increase the number of iterations that are required to find the compression point.

**Maximum Iterations** Specifies the maximum number of power search iterations SMART Sweep is allowed. Reducing this value can cause SMART sweep to terminate before all compression levels are found to within the specified tolerance.

**Show Iterations** When checked, the compression parameter traces are updated at the completion of each power search iteration. When cleared, compression parameter traces are updated when SMART Sweep completes the power search iteration process.

**End of Sweep Condition** Specifies the power level applied to the DUT at the completion of a GCA measurement.

GCA performs numerous power and frequency sweeps on the DUT during the overall measurement process. This setting has no affect on these intermediate sweeps. This setting only applies at the end of the very last sweep in the GCA channel.

In addition, this setting applies ONLY to the GCA channel. All other channels operate independently of this setting. Therefore, the power applied to the DUT after all channels have been measured may be different from this setting.

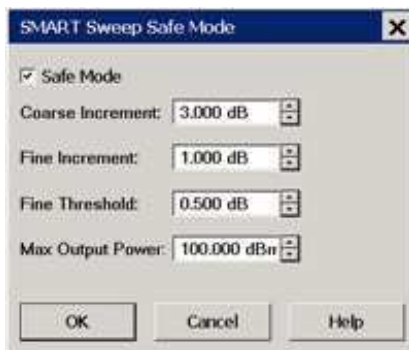
Choose from:

- **Default** Use the default PNA method. [Learn more.](#)
- **RF OFF** RF power is turned off when GCA completes a measurement cycle.
- **Start Power** RF power is set to the start power level.
- **Stop Power** RF power stays at the stop power level.

## Settling Time

Used ONLY in SMART Sweep when Back Off or X/Y compression algorithms are selected.

This setting allows additional dwell time when the input power changes from the back-off level to the compression level. [Learn more.](#)



## Safe Sweep Mode dialog box help

For use with SMART Sweep ONLY.

When enabled, Safe Sweep increases the input power to the DUT by the specified amounts, allowing the compression point to be achieved gradually. While this will increase the number of iterations required to achieve compression, it also minimizes the possibility of driving the DUT too far into compression.

**Note:** Safe Sweep does **NOT** minimize the dramatic change in input power with Backoff and XY method. However, Safe Sweep with Backoff and XY methods DOES prevent the DUT from being exposed to too much input power. [Learn more.](#)

**Safe Mode (Enable)** Check to enable Safe Sweep.

**Coarse Increment** Sets the maximum change in input power, up or down, which will be applied to the DUT from one iteration to the next. Default = 3.0 dB.

Without Safe Sweep, the maximum change in input power can be the entire Backoff or X value when using these compression methods.

**Fine Increment** Once the Fine Threshold has been achieved, this becomes the maximum change in input power, up or down, which will be applied to the DUT. Default = 1.00 dB

**Fine Threshold** Specifies the compression level in which Safe Sweep changes from the COARSE to the FINE increment. Default = 0.5 dB. This means that, by default, the PNA uses the Fine Increment adjustment when compression reaches 0.5 dB.

**Max Output Power** To protect the PNA from damage, when the PNA port that is connected to the DUT Output measures the specified value, the input power to the DUT is no longer incremented at that frequency. In these cases, the compression point would probably not be achieved.

## Compression Analysis

Compression Analysis changes the current trace into a power sweep trace at a specified CW frequency . The current parameter and acquisition method is unchanged. For example, with a CompGain21 trace displayed and SMART Sweep selected, enable Compression Analysis. The trace becomes a power sweep trace at the specified CW frequency. The Y-axis displays S21 Gain at each X-axis power point.

When Smart sweep is used, a complete power sweep is not performed, but only the data points that are required to find the compression point. To see a traditional power IN vs power OUT compression sweep, use one of the [2-D acquisition methods](#).

You can create PNOP or PSAT markers on a CompOut trace with Compression Analysis mode ON. [Learn more.](#)



## How to perform Compression Analysis

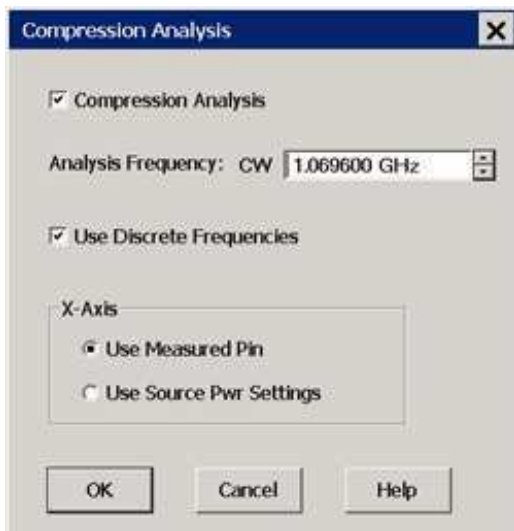
### Using front-panel HARDKEY [softkey] buttons

1. Press **ANALYSIS**
2. then [**Compression Analysis**]

### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Compression Analysis**

## Programming Commands



### Compression Analysis dialog box help

**Notes:** When an S21 or S11 trace is active, any [compression parameter](#) (such as CompGainS21) must also be displayed.

Compression Analysis is NOT allowed for S12 or S22 traces.

Scroll up to [learn more about Compression Analysis](#).

**Analysis Frequency: CW** Enter a frequency to use for the compression analysis trace.

**Compression Analysis** Check to perform compression analysis. A compression trace is displayed at the Analysis (CW) Frequency.

**Use Discrete Frequencies** Check to allow Analysis Frequencies at only the discrete points where data is measured. Clear to allow Analysis CW Frequencies that are interpolated from the data points. Then select ANY CW frequency between the start and stop frequencies of the GCA channel.

#### X-Axis

- **Use Measured Pin** The X-axis displays the actual power that is applied to the DUT after match correction and R-channel drift correction.

- **Use Source Pwr Settings** The X-axis displays the power level of the stimulus.

## Saving GCA Data

Beginning with PNA release A.08.20, GCA data can be saved to a \*.csv file in both 2D and SMART Sweep modes (previously only 2D modes). Also, a Delta Gain column has been added to the data.

### How to save GCA data

With a GCA [Compression](#) trace active:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[Save Data As]**
3. **File Type = GCA Sweep \*.csv**

#### Using a mouse with PNA Menus

1. Click **File**
2. then **Save Data As**
3. **File Type = GCA Sweep \*.csv**

### Programming Commands

## Notes

- This data type can be read by spreadsheet programs, such as Microsoft Excel.
- Data from the last **complete** sweep is saved to the specified \*.csv file.
- If calibration is turned **ON** when the file is saved, then all data is calibrated. Otherwise, raw data is saved.
- All \*.csv data saves include a reference power level sweep at the beginning of each frequency data.

	A	B	C	D	E	F	G	H	I	J	K
1	Agilent Technologies N5242A US46250053.2 08 04 12										
2	Date: Thursday, April 10, 2008 14:15:28										
3											
4	Calibration State OFF*										
5	Num Freq	3									
6	Num Iterat	5									
7											
8	Frequency S11	Pin Data		Pout Data		S21		Delta Gain			
9	<Hz>	<LogMag	<Phase (D	<LogMag	<Phase (D	<LogMag	<Phase (D	<LogMag	<Phase (D	<LogMag	<Phase (D
10	4.00E+09	-4.6375	152.421	-25	0	-17.2892	-34.3579	7.71076	-34.3579	4.44E-16	1.52E-15
11	4.00E+09	-4.6154	152.473	-19.7657	-132.092	-12.0009	-166.651	7.76485	-34.5593	0.054092	-0.20141
12	4.00E+09	-4.61091	152.572	-16.7746	-54.5945	-8.99388	-88.9208	7.78073	-34.3263	0.069968	0.031649
13	4.00E+09	-4.8083	152.424	-13.7818	160.191	-6.00358	125.667	7.77622	-34.5231	0.067459	-0.16519
14	4.00E+09	-4.59279	152.327	-10.7716	152.283	-3.00766	117.78	7.76391	-34.5024	0.053151	-0.14452
15	4.25E+09	-5.43633	-125.845	-25	0	-15.7527	4.97897	9.24726	4.97897	0	4.20E-16
16	4.25E+09	-5.46401	-125.925	-19.979	-118.327	-10.727	-113.371	9.252	4.96641	0.004746	-0.02256
17	4.25E+09	-5.45105	-125.983	-16.978	-141.044	-7.71819	-136.062	9.25881	4.98213	0.012559	0.003155
18	4.25E+09	-5.47133	-126.249	-13.9561	80.5926	-4.7118	85.3736	9.24425	4.78099	-0.003	-0.19798
19	4.25E+09	-5.46864	-126.271	-10.9576	-68.4214	-1.74774	-63.6055	9.2101	4.81595	-0.03716	-0.16302
20	4.50E+09	-6.37248	-34.7728	-25	0	-14.521	52.0538	10.4789	52.0538	-8.98E-16	9.26E-16
21	4.50E+09	-6.377	-35.0287	-20.0536	-142.773	-9.58433	-90.7563	10.4691	52.0174	-0.00982	-0.03642
22	4.50E+09	-6.349	-34.9226	-17.0636	-165.799	-6.56493	-113.678	10.4987	52.1201	0.019756	0.066347
23	4.50E+09	-6.36875	-35.0054	-14.055	55.4364	-3.57836	107.509	10.4766	52.073	-0.00234	0.019178
24	4.50E+09	-6.36475	-34.91	-11.0519	-94.0431	-0.59745	-41.9732	10.4545	52.0699	-0.02445	0.016094

The data are organized by frequency / Power IN (as shown above) regardless of the method used to acquire the

data. The above image shows SMART Sweep data with 5 iterations and 3 frequency points. The yellow highlight is added here for readability.

## GCA Measurement Tips

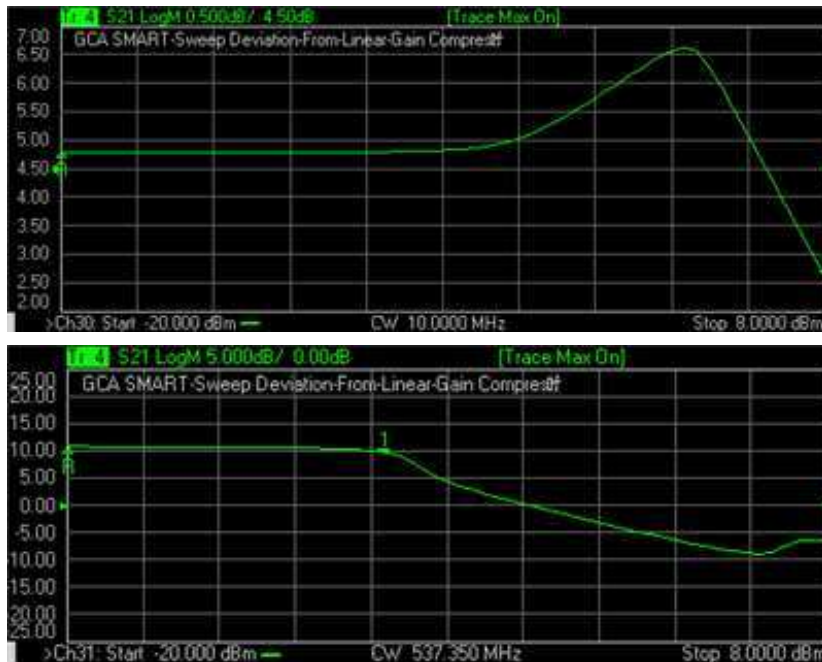
There are many settings in the Gain Compression Application. Here are a few tips when using GCA to learn as much as possible about the compression characteristics of your DUT in the most efficient manner.

### DUT Compression Characteristics and GCA

Although GCA provides excellent results with a wide variety of amplifiers, it works best with amplifiers which have a monotonic compression curve. In some cases where the compression curve is not monotonic, for example if the amplifier gain expands before it compresses, the correct compression level may not be found.

To help a SMART sweep find the correct compression point, limit the Start and Stop power levels around the anticipated compression point. [Learn more.](#)

The following two power-sweep traces are examples of non-monotonic gain:



### DeltaGain

A DeltaGain trace is the best way to see how closely GCA is actually measuring to the desired compression level. In addition, you can view the phase of DeltaGain to see the phase deviation between the [compressed gain](#) and the [reference gain](#). DeltaGain is calculated as:

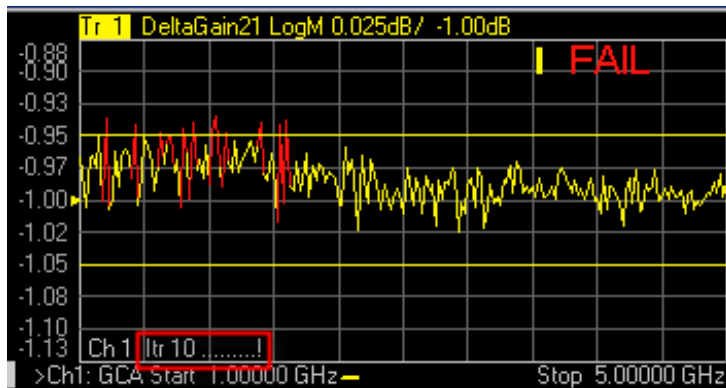
- $\text{DeltaGain} = \text{Measured Gain (watts)} / \text{Ref Gain (watts)}$
- In LogMag format:  $\text{DeltaGain} = (\text{Measured Gain}) - (\text{Ref Gain})$

With SMART Sweep, DeltaGain (in LogMag format) shows how soon certain frequencies achieve the specified tolerance. [Learn more.](#)

Some other settings which may be helpful:

- Trigger source: Manual allows you to analyze data and make adjustments while allowing the device to cool.
- Construct Limit Lines around the compression point at the tolerance level.

The following image shows a DeltaGain21 trace using SMART Sweep. The Limit Lines were added manually.



In the above image:

Relevant Settings	Method = Compression From Linear Gain Compression level = 1 Iteration Tolerance = 0.05 dB. Maximum Iterations = 10
Displayed Results	<p>A data point on -1.00 indicates that, at that frequency, the exact compression level (1 dB) was measured.</p> <p>Several frequencies did not achieve the specified tolerance (0.05 dB) before the Max Iterations (10) was reached.</p> <ul style="list-style-type: none"> <li>• FAIL and red data points outside the limit lines.</li> <li>• Nine dots (...) indicate that 90% of the data points achieved the specified compression level.</li> <li>• one ! indicates that 10% of the data points did not achieve compression.</li> <li>• <a href="#">Learn more about the Iteration Counter and annotation.</a></li> </ul>

### SMART Sweep Tips

- Compression from Linear Gain is the easiest compression method to understand and control in SMART Sweep. [Learn more.](#)
- If SMART Sweep requires more than twenty iterations, this is an indication that something is wrong. Try changing the Tolerance setting, Frequency Range, Start / Stop power range, IF bandwidth, or [Dwell Time](#).
- If the number of iterations required to achieve the desired compression level changes significantly from one set of measurements to the next, this could be due to other effects, such as heating. Try increasing the dwell

time or using a [wideband pulse](#) measurement configuration.

- If the DUT should not be significantly overdriven into compression, or the changes in the input power should be limited, use [Safe Sweep](#) mode with Deviation from Linear Gain compression method.

## Single Frequency Macros

**Note:** Beginning with PNA rev. A.09.00, the [Compression Analysis](#) feature provides an easier method of viewing a traditional power sweep at a single frequency than the GCA macros. However, the Macros are still maintained on the PNA hard drive.

The macros perform a single power sweep on the DUT using a standard channel with corresponding stimulus settings. The macro can show measurement differences from the compression analysis traces due to bias/thermal/settling effects of the DUT. So, the macro can help confirm a DUT is exhibiting some type of settling behavior which will need to be handled in some way.

Also, the macro is a great GCA programming example.

With a 2D sweep (NOT SMART Sweep) a script that is stored on the PNA hard drive automatically creates a traditional power sweep measurement in a standard channel using the same stimulus setting as the GCA channel. Use a marker in the GCA channel to specify the frequency for the measurement.

The script has two modes of operation:

1. **View Mode** displays all of the previous 2D sweep data at that frequency.
2. **Measure Mode** performs a new measurement at that frequency.

Both modes create a new S-Parameter channel using the same stimulus settings as the GCA channel, including port power, attenuator, IF Bandwidth, and dwell settings. The new channel does not support calibration or pulse characteristics.

To see noise on a measurement, use the **Measure** macro in continuous sweep. Adjust the IFBW and averaging until the noise versus sweep speed meets your needs.

To see other effects of your DUT at a specific frequency, use the **View** macro and the **Measure** macro with 2D sweep mode. Both macros present data using a standard channel. The View macro shows 2D data at a specific frequency, while the Measure macro shows freshly-measured data at the same frequency. Ideally, the data from these two would be identical. However, changes in your DUT behavior due to heating or other effects can cause these to be different. If significant differences exist, try:

- Using the 2D Frequency per Power setting rather than Power per Frequency
- Adjusting the dwell time
- Adjusting IFBW
- Use a [wideband pulse configuration](#)

## How to setup the Macros

Each macro must be setup separately.

1. Press **Macro**, then **Macro Setup**.

2. Select a blank line, then click **Edit**.
3. In **Macro Title**, type a short description such as Meas GCA or View GCA.
4. Click **Browse**, then navigate to C:/Program Files/Agilent/Network Analyzer/Applications/GCA/GCA.vbs
5. In Macro run string parameters:
  1. Type **M** for the Measure macro or **V** for View macro.
  2. Optional: Supply the following additional parameters in any order:
    - To run the program from a remote computer, specify the full computer name of the PNA .
    - Channel in which to create the measurement. If not specified, Measure is created in Ch30 and View is created in Ch31.
    - Example: Run string parameters for the Measure macro run from a remote computer in Channel 5.----  
M MyPNA 5.
6. Click **OK**.

### How to run the Macros

On a GCA channel:

1. Create a 2D sweep. Either Power per Freq or Freq per Power. Both macros always create a power sweep at the frequency of interest.
2. Create a Compln trace.
3. On the Compln trace, right-click and select **Add Marker**. Drag the marker to the frequency of interest.
4. Press **Macro**, then select either by the short description your provided in Step 3.

---

Last Modified:

17-Mar-2010	Added Max power and pulse support - removed some fixturing limitations
19-Nov-2009	Added X-axis point spacing (9.1)
13-Aug-2009	Added Compression Analysis
30-Mar-2009	Added ADC measurements
23-Feb-2009	Added link to compression marker
1-Dec-2008	Clarified SMART - Linear compression
9-May-2008	Edited for data save, backoff, and XY method
23-Aug-2007	New topic

## Gain Compression Calibration

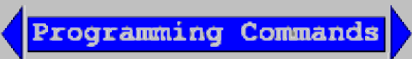
The GCA Calibration Wizard guides you through a calibration of the GCA channel. The procedure is the same regardless of the Gain Compression Settings.

- A [Source Power Calibration](#) and [Receiver Calibration](#) is performed first.
- Then, [your choice of a Full 2-port Cal or an Enhanced Response Cal](#).

### See Also

[Gain Compression Application](#)

[Calibration Programming commands](#)

How to start a GCA Calibration	
Using front-panel HARDKEY [softkey] buttons	Using a mouse with PNA Menus
1. Press CAL	1. Click <b>Response</b>
2. then [Start Cal]	2. then <b>Cal Wizard</b>
3. then [Cal Wizard]	
	

### Overview - GCA Source Power Cal

The GCA Source Power Calibration is a little different from a standard Source Power Cal. Although GCA measurements are performed at many power levels, the GCA source power cal is performed at a [single power level](#) over the specified frequency span of your GCA measurement. The required source correction from that single power level is applied to all power levels. This method ensures that the 'absolute' power level being applied to the DUT is within the PNA-X source power linearity specification.

Although it is important for GCA to be able to **set** the absolute power level to the DUT, it is MOST important to be able to exactly **measure** the actual incident power. Therefore, during the GCA Source Power Cal, a receiver calibration is applied to the port 1 reference receiver, and indirectly to both test port receivers during the S-parameter calibration, correcting for impedance mismatch between the power meter and the PNA source, and the DUT and the PNA source.

Although the cal process is also at a single power level, the dynamic accuracy of the PNA-X receivers is typically about +/- .05 dB, which is comparable to the accuracy of Agilent's best power sensors. This allows GCA to **very accurately** measure and report ALL power levels that are actually applied to the DUT.

### Full 2-port or [Enhanced Response \(ER\) Cal](#)

The following issues may help you decide between these two Cal types:



- **Accuracy** A full 2-port correction is more accurate than ER when GCA measures linear gain. However, for non-linear measurements, ER yields identical compression values as a full 2-port cal, so this may not be a significant factor.
- **Measurement speed** An ER correction only requires measurements in the forward direction. The reverse parameters (usually S22 and S12) are not measured unless requested. With a full 2-port cal applied, all four S-parameters are measured, which requires an additional reverse sweep. [Learn more.](#)
- **Ease** A full 2-port cal is easiest with an ECal module. An ER Cal requires a [Defined Thru](#) or a [Flush Thru Cal](#) method. If these are possible, then an ER cal is easiest when using a mechanical Cal Kit.
- **High power** The test port damage level of a standard PNA-X is +30 dBm. Therefore, external attenuation may be required on the output of high power amplifiers, which degrades calibration accuracy for reverse (full 2-port) measurements. In addition, the external attenuation improves the DUT output / load match error, which allows a better uncorrected response and makes an Enhanced Response Cal the better choice.
- **DUT limitations** With an ER Cal applied, reverse measurements on the DUT are not performed unless requested.

### How to select Enhanced Response Cal

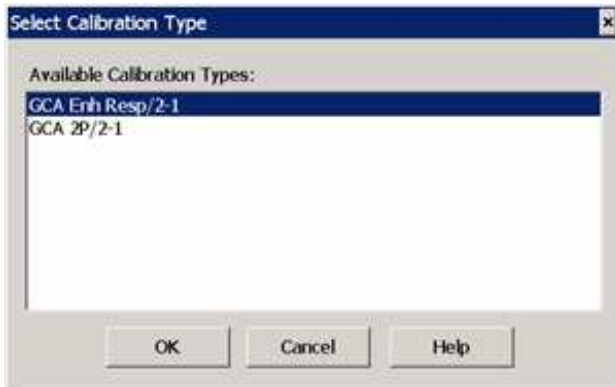
At the [Select DUT Connectors](#) page of the GCA Cal Wizard:

1. Check **Modify Cal**, then click **Next**.
2. A [Defined Thru](#) or a [Flush Thru Cal](#) method must be selected.
3. Click **Cal Type/Std**s
4. Under Calibration type, select **EnhResp** (2 <= 1 refers to the receive port 2 and source port 1).

**Downgrade a Full 2-port Cal to Enhanced Response Cal** if you prefer to perform a Full 2-port cal, but not perform reverse sweeps on the DUT.

To change the correction on the channel from Full 2-port to Enhanced Response:

1. Press **CAL**
2. then **[Manage Cal]**
3. then **[Cal Type]**
4. Select **GCA EnhResp**, then **OK**



## GCA Cal Wizard



### Select DUT Connectors and Cal Kits - GCA Cal dialog box help

This is a [standard Cal Wizard](#) page except for the following:

**Power Sensor** Specify the connector type and gender of the power sensor. When the power sensor connector is not the same type and gender as the DUT Port 1 connector, then for optimum accuracy, extra cal steps are required to measure and correct for the adapter that is used to connect the power sensor to the port 1 reference plane during the Source Power Cal. Select the Cal Kit that will be used for that process.

Select **Ignored** (at the bottom of the DUT Connectors list) to NOT compensate for the adapter.

**Modify Cal** Check, then click **Next**, to Modify Cal (Standards AND Thru Method).

**View/Modify Source Cal Settings** Click to launch the [Source Cal Settings](#) dialog.

[Learn more about GCA Source Power Calibration](#)



### Gain Compression Calibration Step 1 dialog box help

**Power Level** at which to perform the Source Power Cal.

It is usually best to perform the Source Power Cal at 0 dBm because the power sensor is calibrated at that level.

However, if the Gain Compression measurement is performed entirely below or above 0 dBm, then perform the Source Power Cal at the **Stop** power which probably has the lowest level of measurement noise.

[Learn more about GCA Source Power Calibration](#)

The remaining Gain Compression Cal dialogs are the same as the standard [SmartCal dialogs](#).

Return to [Gain Compression Application](#).

---

Last Modified:

26-Nov-2007    MX New topic

## iTMSA (Integrated True Mode Stimulus Application)

---

This application is integrated into the PNA firmware as **Option 460**. A previous version of TMSA is run as a Macro on the PNA.

- [Features, Requirements, and Limitations](#)
- [True Mode Stimulus Concepts](#)
- [Using iTMSA](#)
  - [Create a Measurement](#)
  - [Set Power Level](#)
  - [Calibrate](#)
  - [Ratioed Receiver Measurements](#)

### Other PNA Applications

**Notes:** When the first iTMSA measurement is created, an IFMUX Cal is performed, which takes a couple of minutes. If an IF board is replaced, this Cal should be performed again. To make this happen, delete C:/Program Files/Agilent/Network Analyzer/IFMUX.txt. When the next iTMSA measurement is created, the file will be recreated with new data.

## Features, Requirements, and Limitations

### Features

- A seamless extension of existing PNA Balanced Measurement, but with True Mode stimulus.
- True Mode Stimulus measurements are performed in a standard S-parameter channel.

### Requirements

- [4-port PNA-X](#) (opt 400) with **Opt 460** (software option only); [must be enabled](#).

### Limitations with iTMSA

The following standard PNA features are **NOT** available with iTMSA measurements:

- [External Test Set Control](#) (Option 551)
- [Interface Control](#)

- Time Domain Pulse measurements in [Wideband Pulse](#) are NOT supported.
- [Unratioed reference receiver](#) measurements.
- [Segment Sweep](#)
- [Frequency Offset Measurements](#) (opt 080)
- [Receiver Calibration](#)

## True Mode Stimulus Concepts

A balanced device is designed to receive input simultaneously across two ports. Standard PNA Balanced measurements apply stimulus to one port at a time, measures the responses, and calculates the theoretical balanced responses. [Learn more about balanced measurements.](#)

True Mode Stimulus uses two PNA sources to apply either truly differential (180 degree out-of-phase) or truly common (in-phase) signals across the input of a balanced device. PNA receivers measure the single-ended response at the output of the device and calculate the balanced response.

When operating in non-linear regions, a device may respond differently to single-ended stimulus than to True Mode Stimulus. Thus, True Mode Stimulus capability allows you to understand when, and if, True Mode Stimulus is required.

For more detailed information on this measurement technique, see the following white papers (internet connection required):

- [New Methods & Non-Linear Measurements for Active Differential Devices](#)
- [New Measurement Results and Models for Non-linear Differential Amplifier Characterization](#)

## How iTMSA Works - Overview

The following is an overview of how iTMSA gathers and displays True Mode Stimulus data:

1. **Initial sweep** is performed to gather the initial phase deviation between the two sources at the reference plane.
2. **Measurement** The phase deviation between the two sources is set to the required value and the response is measured. For a differential sweep, the phase deviation of the two sources is set to 180° at the reference plane. For a common sweep, the phase deviation is set to 0° at the reference plan.

A forward direction measurement requires 2 sweeps for differential and 2 sweeps for common. A reverse direction measurement requires the same. Therefore, a complete measurement in both directions requires 8 sweeps. The iTMSA traces are updated when all of the necessary data is gathered in each direction.

iTMSA computes the raw S parameters with the following matrix:

$$\begin{bmatrix} \frac{b_1}{a_1} d1 & \frac{b_1}{a_2} d2 & \frac{b_1}{a_1} c1 & \frac{b_1}{a_2} c2 \\ \frac{b_2}{a_1} d1 & \frac{b_2}{a_2} d2 & \frac{b_2}{a_1} c1 & \frac{b_2}{a_2} c2 \\ \frac{b_3}{a_1} d1 & \frac{b_3}{a_2} d2 & \frac{b_3}{a_1} c1 & \frac{b_3}{a_2} c2 \\ \frac{b_4}{a_1} d1 & \frac{b_4}{a_2} d2 & \frac{b_4}{a_1} c1 & \frac{b_4}{a_2} c2 \end{bmatrix} \cdot \begin{bmatrix} \frac{a_1}{a_1} d1 & 0 & \frac{a_1}{a_1} c1 & 0 \\ 0 & \frac{a_2}{a_2} d2 & 0 & \frac{a_2}{a_2} c2 \\ \frac{a_3}{a_1} d1 & 0 & \frac{a_3}{a_1} c1 & 0 \\ 0 & \frac{a_4}{a_2} d2 & 0 & \frac{a_4}{a_2} c2 \end{bmatrix}^{-1} = \begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ s_{21} & s_{22} & s_{23} & s_{24} \\ s_{31} & s_{32} & s_{33} & s_{34} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix}$$

#### Legend:

- d1**: differential stimulus in balanced port 1.
- d2**: differential stimulus in balanced port 2
- c1**: common stimulus in balanced port 1.
- c2**: common stimulus in balanced port 2.

### Using iTMSA

A Standard (default) S-Parameter channel is used for iTMSA measurements.

#### How to create an iTMSA trace:

1. Preset the PNA
2. Then do the following:

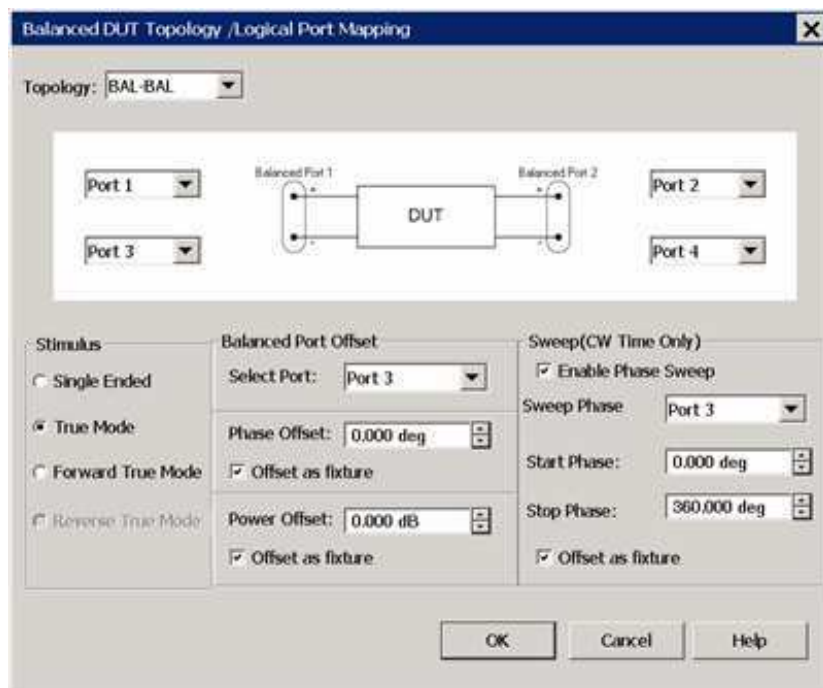
#### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then **[Balanced Parameters]**
3. then press the **Change** button

#### Using a mouse with PNA Menus

1. click **Response**
2. then **Measure**
3. then **Balanced Parameters**
4. then press the **Change** button

**Programming Commands**



### Balanced DUT Topology / Logical Port Mapping (with iTMSA Option 460 ) dialog box

[Learn about iTMSA.](#)

#### Topology

Only the following topologies and port mappings are available in True Mode Stimulus:

- **BAL-BAL** DUT has two balanced ports. The PNA ports are restricted to the following:
  - Balanced Port 1 = PNA ports 1 and 3
  - Balanced Port 2 = PNA ports 2 and 4
- **SE- BAL** DUT has one single-ended port and one balanced port. For a **BAL-SE** (balanced - single-ended) DUT, select this topology, and then select **Reverse True Mode**. The PNA ports are restricted to the following:
  - Single-ended Port 1 = PNA port 1.
  - Balanced Port 2 = PNA ports 2 and 4.

#### Stimulus

**Single Ended** Standard PNA mode. Each DUT port receives stimulus individually.

**True Mode** Each balanced port receives true balanced stimulus in both forward and reverse directions.

**Forward True Mode** Available ONLY in BAL- BAL topology. Only Balanced port 1 receives true balanced stimulus.

**Reverse True Mode** Available ONLY in SE- BAL topology. Only Balanced port 2 receives true balanced

stimulus.

## Balanced Port Offset

**Select Port** Select the physical PNA port to receive a phase or power offset. Only port 3 and port 4 are available for Phase or Power offset.

**Phase Offset** Specify offset within +/- 180° for the balanced INPUT port. This is in addition to the standard offsets, which are 180° offset for the differential stimulus sweeps, and 0° for the common stimulus sweeps.

### Offset (phase) as Fixture

When unchecked, output calculations are performed and displayed as though there is only the standard offset. Although additional Phase Offset is applied to the stimulus, it is ignored in the calculations of balanced differential and common mode output signals.

When checked, output calculations are performed and displayed using the actual phase offset that is applied to the DUT. Use this setting to compensate for a component or fixture that may present a phase delay before the DUT.

**Power Offset** Specify power offset. Range is +/- 2 dB. This is in addition to the power level that is specified using the [Power and Attenuators dialog](#). Offset Power is NOT reflected on the power dialog nor on the X-Axis during a power sweep.

### Offset (power) as Fixture

When unchecked, output calculations are performed and displayed as though there is no stimulus power offset.

When checked, output calculations are performed and displayed using the power offset that is applied to the DUT. Use this setting to compensate for a component or fixture that may present a magnitude loss before the DUT.

## Phase Sweep

- Available only when [Sweep type](#) = CW.
- Only PNA port 3 **OR** port 4 are eligible for Phase sweep.
- Available ONLY for balanced pairs.

The phase of the selected port (3 or 4) is swept relative to the phase of the other port (1 or 2) in the balanced pair.

Phase Sweep is similar to phase offset, except that for each data point, the phase 'offset' is incremented.

For example, with the topology that is selected in the above image: (Bal-Bal) Logical port 1 = PNA ports 1 and 3. For a phase sweep with 7 data points, from 0° to 180°, the phase difference between port 1 and port 3 increments 30° with each data point:

Data point	1	2	3	4	5	6	7
Delta phase	0°	30°	60°	90°	120°	150°	180°

**Enable Phase Sweep** Check to enable phase sweep for the specified port.

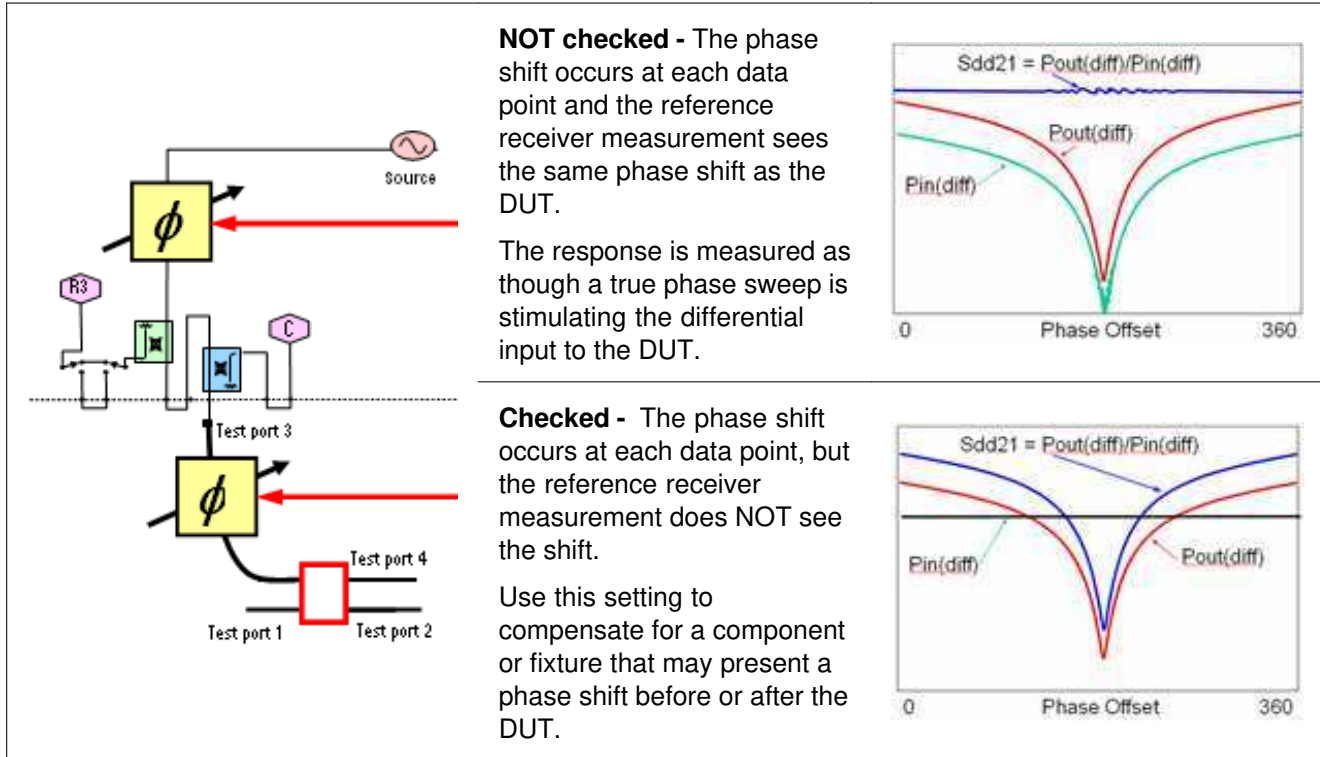


**Sweep Phase** Choose EITHER PNA Port 3 OR Port 4 to sweep. Only ONE port can be phase swept.

**Start / Stop Phase** Enter phase values in degrees. Each sweep will start and stop at these settings.

### Offset (phase) as Fixture

In the following image, the PNA phase sweep is shown as a phase shifter that is "virtually" located **EITHER** before the reference receiver (top) **OR** after the reference receiver (bottom) - **NOT BOTH**. This image is useful as a model ONLY.



### One use for Phase Sweep

Do the following to find the best operating point:

1. Phase sweep port 3, from 0° to 180°. Note the peak of Sdd21. This finds the phase offset of the input differential port.
2. Set port 3 **fixed offset** to the phase that has the highest Sdd21.
3. Then, phase sweep port 4 (balanced port 2) to find the phase offset of the output differential port.
4. Then set the port 4 to fixed phase offset found.

This measures the best possible case Sdd21 for the DUT.

### Power and Attenuators Settings

Set stimulus power levels using the standard PNA Power and Attenuators dialog.

To set power, press **STIMULUS**, then **[Power and Attenuators]**

<input checked="" type="checkbox"/> Power ON (All Channels)		<input checked="" type="checkbox"/> Port Powers Coupled					
Name	State	Port Power	Start Power	Stop Power	Auto Rang	Atten. Contr	Leveling Mod
Bal Port 1	Auto	-2.00 dBm	-7.00 dBm	3.00 dBm	<input checked="" type="checkbox"/>	0 dB	Internal
Bal Port 2	Auto	-2.00 dBm	-7.00 dBm	3.00 dBm	<input checked="" type="checkbox"/>	0 dB	Internal
Port 1	Auto	-5.00 dBm	-10.00 dBm	0.00 dBm	<input type="checkbox"/>	0 dB	Internal
Port 2	Auto	-5.00 dBm	-10.00 dBm	0.00 dBm	<input type="checkbox"/>	0 dB	Internal
Port 3	Auto	-5.00 dBm	-10.00 dBm	0.00 dBm	<input type="checkbox"/>	0 dB	Internal
Port 4	Auto	-5.00 dBm	-10.00 dBm	0.00 dBm	<input type="checkbox"/>	0 dB	Internal

### Power and Attenuation (with True Mode Stimulus) dialog box help

When a True Mode Stimulus is selected in the [Balanced DUT Topology dialog](#), the balanced ports are shown on the top rows of the dialog. The individual PNA port settings are displayed but can not be changed directly.

**Port Powers Coupled** Check to couple all power settings for Balanced Port 1 and Balanced Port 2. Clear this box to make independent power settings for these logical ports.

**State** Should be left in Auto for iTMSA measurements. [Learn more about his setting.](#)

**Port Power** Set power for the balanced port. Power for PNA ports, shown below balanced ports, are set to 3 dB less, or half power. [Power Offset](#) is made in addition to this setting, and on only PNA port 3 and port 4. Offset Power is NOT reflected on this dialog nor on the X-Axis.

**Start and Stop Power** Available when sweep type is set to [power sweep](#).

**Auto Range** Check to allow the PNA to select the [optimum attenuation value](#) to achieve the specified test port power. Clear to manually set the attenuation for each port. Type or select the attenuation value in the adjacent Attenuator Control box. When the attenuator setting of a logical port is changed, then the attenuators of the PNA ports associated with that logical port change to the same value.

**Leveling Mode** Open Loop leveling is available only on ports 1 and 3. [Learn more.](#)

**Note:** The range of leveled power (ALC range) for balanced ports is 3 dB higher than it is for each individual PNA port. For example, if a PNA source port with 0 dB attenuation will supply leveled power from -30dBm to +15dBm, then the balanced logical port has a range of -27dBm to +18dBm. [Learn more about Leveled Power.](#)

### Calibration with iTMSA

**Note:** Uncalibrated True Mode Stimulus are **NOT** at all accurate.

Perform or recall a SMART Cal exactly like any other 3 or 4-port cal.

Press **CAL**, then **[Cal Wizard]**.

- All Cals are performed as single-ended.
- Supports all Fixturing and Port Extension features.
- [Receiver Calibration](#) is NOT allowed.
- Supports [Enhanced Response Cal](#) and [Source Power Cal](#).

### Source Power Cal

Perform a Source Power Cal on:

- **Port 3** for BAL Port 1

- **Port 4** for BAL Port 2

Port 1 and Port 2 power levels are adjusted to these ports in the [initial sweep](#).

## Receiver Measurements

Any pair of receivers can be viewed as a ratio using the following dialog.

Unratioed measurements using reference receivers (an 'R' receiver as numerator and **1** as denominator) are NOT accurate.

To select these measurements, press **MEAS**, then **[Receivers]**

S-Parameter	Balanced	<b>Receivers</b>
Activate: <input checked="" type="checkbox"/>	Numerator R3	Denominator R1
	/	Source Port Port 1

### Create / Change Receiver Measurements (with True Mode Stimulus) dialog box help

Click **Activate**

- For **RATIOED** measurements, Select a receiver for the **Numerator** and select a receiver for the **Denominator**.
- For **UNRATIOED** measurements, Select a test port receiver (A, B, C, or D) for the **Numerator**. Reference receiver measurements are NOT accurate. Select **1** for the **Denominator**.

For example, with a Bal-Bal topology, the above selections show a R3/R1 measurement. R (reference) receivers measure the stimulus to the DUT. An R3/ R1 trace shows the difference between the two sources that comprise logical port 1. With Log format, a power offset between the two sources is visible. With Phase format, a phase offset is visible.

**Source Port** Specifies whether a Differential stimulus or Common mode stimulus is used for the measurement.

- With Source Port **1** or **2** selected, then **Differential** stimulus is used for the unratioed measurement which causes **180° offset** between the sources.
  - Source Port 1 = Differential stimulus on BAL1 port
  - Source Port 2 = Differential stimulus on BAL2 port
- With Source Port **3** or **4** selected, then **Common** mode stimulus is used for the unratioed measurement which causes **0° offset** between the sources.
  - Source Port 3 = Common-mode stimulus on BAL1 port
  - Source Port 4 = Common-mode stimulus on BAL2 port

Last Modified:

13-Nov-2008 Fixed allowed SE-BAL topology

19-May-2008 MX New topic

## Noise Figure Application (Opt 028 and Opt 029)

---

The Noise Figure Application makes fast, easy, and accurate noise figure measurements using the [PNA-X](#). This topic discusses Noise Figure on Amplifiers.

**See Also:** [Noise Figure on Converters \(NFX\)](#)

- [Noise Figure Options Explained \(029, 028, H29\)](#)
- [Features, Requirements, and Limitations](#)
- [Noise Concepts](#)
- [How the Noise Figure Application Works](#)
- [Scalar Noise Figure Measurements](#)
- [Noise Power Parameters](#)
- [Using Noise Figure App](#)
  - [Connect Tuner and Noise Source.](#)
  - [Create a Noise Figure Measurement.](#)
  - [Make Noise Figure Settings.](#)
  - [Perform Calibration](#)
- [Noise Figure Measurement Tips](#)
- [Noise Model and the Noise Correlation Matrix](#)

### See Also

[Noise Figure Calibration](#)

[Programming commands](#)

[Agilent Noise Figure App Note 57-1](#)

High-Accuracy Noise Figure Measurements Using the PNA-X (Search Agilent.com for App Note **1408-20**)

[See other PNA Applications](#)

## Noise Figure Options Explained (029, 028, H29)

- **029** - Includes two low-noise receivers and noise tuner bypass switch to enable noise figure measurements to 26.5 GHz. Also has Opt 028 capability.
- **028** - Uses standard PNA-X receivers to measure noise figure. A noise source is NOT required during

calibration. Any two PNA-X ports can be used. Use with DUTs that have sufficiently high gain and noise figure. Additional filtering may be necessary. [Learn more.](#)

- **H29** - Noise Figure on N5244A (43.5 GHz model) and N5245A (50 GHz model). Includes both:
  - Opt 029 (noise measurements to 26.5 GHz)
  - Opt 028 (noise measurements using standard receiver to 50 GHz) . [See H29 specifications.](#)

### Noise Figure Application Features

- Cold noise method includes correction for imperfect system source match, for highly accurate noise figure measurements.
- Measure noise figure, compression, intermodulation distortion, and more, all with a single connection to the DUT.
- Operates from 10 MHz to 50 GHz, with choice of low-noise receivers to 26.5 GHz or standard receivers to 50 GHz. (Opt 028)
- Opt 029 measures devices with Noise Figure values ranging from about 0 to 50 dB and devices with GAIN ranging from about -40 to +60 dB. [Learn more.](#)
- Measure noise figure of frequency translating devices. [Learn more.](#)
- During calibration, ENR values are interpolated for frequencies between the supplied data points.

### Noise Figure Application Requirements

- Agilent 346C Noise Source: Covers the same frequency range as the PNA-X. NOT required with Opt 028.
- An adapter may be necessary to connect the Noise Source to the PNA-X port 2 reference plane during [calibration](#).
- Noise Tuner (ECal Module - N4691B m-f recommended). Not required for [Scalar Noise Figure](#). Opt 029 provides an additional cable and adapter to connect the ECal module to the front-panel connectors. [Learn more.](#)
- Cal Kit (or second ECal module) with same connector type and gender as DUT connectors.
- Power Meter - Required when calibrating [NFX](#) (Noise Figure on Converters).
- Recommended: An accurate thermometer. [Learn more.](#)

### Limitations with the PNA-X Noise Figure Application

The following PNA-X features are NOT supported in a Noise Figure channel:

- [FCA \(opt 083\)](#) or [Frequency Offset \(opt 080\)](#).
- [Analog sweep](#). All frequency sweeps are STEPPED.

- [External Test Set Control](#) (Opt 550 or 551)
- [Receiver calibration](#).
- [Enhanced Response Cal](#)
- [ECal User Characterization](#).
- Some [Fixturing Features](#)
- [Copy Channels](#)
- [Auto Formatted Citifile](#) data.
- [Pulsed](#) Noise Figure measurements are supported with the following limitations:
  - Minimum 300 microsecond pulse width using 24 MHz noise bandwidth
  - Narrower noise bandwidths cause larger minimum pulse widths
  - A drop-out may occur at start of sweep and at 3 GHz. This is corrected by a 1 ms pulse width at 24 MHz Noise BW.

## Noise Concepts

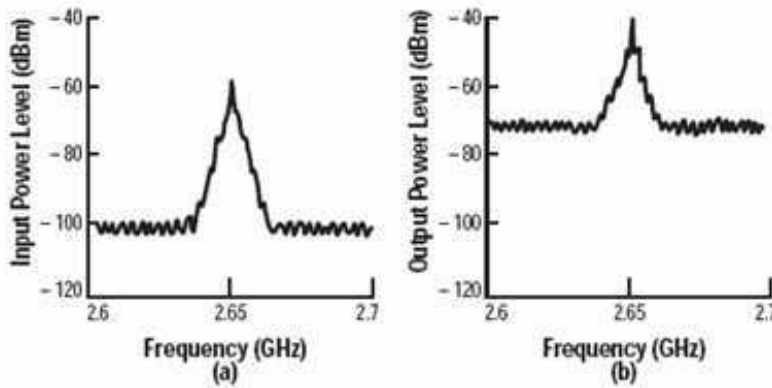
The following conceptual information is a short summary taken from the [Agilent Noise Figure App Note 57-1](#).

All electronic circuits have some degree of random noise. The most common form is thermal noise, which increases as the temperature of the circuit increases.

The signal-to-noise (S/N) ratio of components in a communications system is a very important parameter. To improve the S/N ratio, it is usually easier and more cost-effective to reduce noise than to increase signal power. In order to reduce noise, an accurate method to measure noise is required.

### Noise Figure

Noise figure is the degradation in the signal-to-noise ratio as a signal passes through a device. For example, in the following images:



**(a) At the INPUT of an amplifier:**

The noise floor is -100 dBm, the signal is at -60 dBm, **40 dB** above the noise floor.

**(b) At the OUTPUT of the same amplifier:**

The gain has boosted the signal AND the noise floor by 20 dB.

The amplifier then added 10 dB of its own noise.

The output signal is now only **30 dB** above the noise floor.

Since the degradation in signal-to-noise ratio is 10 dB, the amplifier has a **10 dB noise figure**.

For consistency, noise measurements are calculated as if using a 1 Hz bandwidth, although measurements are almost always made at higher bandwidths.

The following formula shows the lowest possible noise power in dBm at 290° K (room temperature). The only way to measure noise lower than this is to make the measurement at a lower temperature.

- $P = 10\text{LOG}(4.0 \times 10^{-21} \text{ watts}/.001 \text{ watt})$
- $P = -174 \text{ dBm} / \text{Hz}$

## How the Noise Figure Application Works

The goal of the Noise Figure application is to accurately measure the noise that is generated by the DUT. This may be done using special low-noise receivers or using the standard PNA-X receivers depending on whether the PNA-X has Options 029 or 028. [Learn more](#). The low-noise receivers are calibrated using a characterized noise source. The standard receivers are calibrated using a power meter and a measurement of the receivers effective noise bandwidth. [Learn more about the noise calibration process](#).

A source of noise measurement error is caused by a poor source match presented to the DUT input. Therefore, during every measurement, the Noise Figure Application uses an ECal module to present at least four different impedances at the input of the DUT. This "Noise Tuner" is connected to the PNA-X port 1 front-panel loops which is in the PNA-X internal source path. From the measurements at various impedance states, the PNA-X calculates the noise out of the DUT as though the PNA-X were exactly 50 ohms. No assumptions are made regarding the input impedance of the DUT.



Here is how a Vector Noise Figure measurement is made using Option 029. The sweep numbers are annotated on the PNA-X display as they occur.

1. With the noise tuner in the THRU state, S-parameter measurements are made to accurately characterize the gain of the DUT. This requires sweeps in both forward and reverse directions. (sweep #1 and #2).
2. The noise measurements are performed next. PNA-X source power is turned OFF and the noise tuner is switched to the first impedance state.
3. At each frequency, the noise receiver samples a large number of readings in order to attain **one** valid measurement. If [Noise Averaging](#) is selected, the specified number of measurements are made and averaged together to obtain one noise measurement. This continues for all frequencies (sweep #3).
4. The next noise tuner impedance state is switched IN and the noise measurements in step 3 are repeated. This occurs until measurements are made at all impedance states. At least four impedance states must be used. (sweeps #4, #5, #6+)
5. Calibration error terms are applied and calculations made to simulate the measurement with a perfect 50 ohm input impedance. The sweep result is plotted on the PNA-X display.
6. The PNA-X begins sweeping again with step 1.

## Scalar Noise Figure Measurements

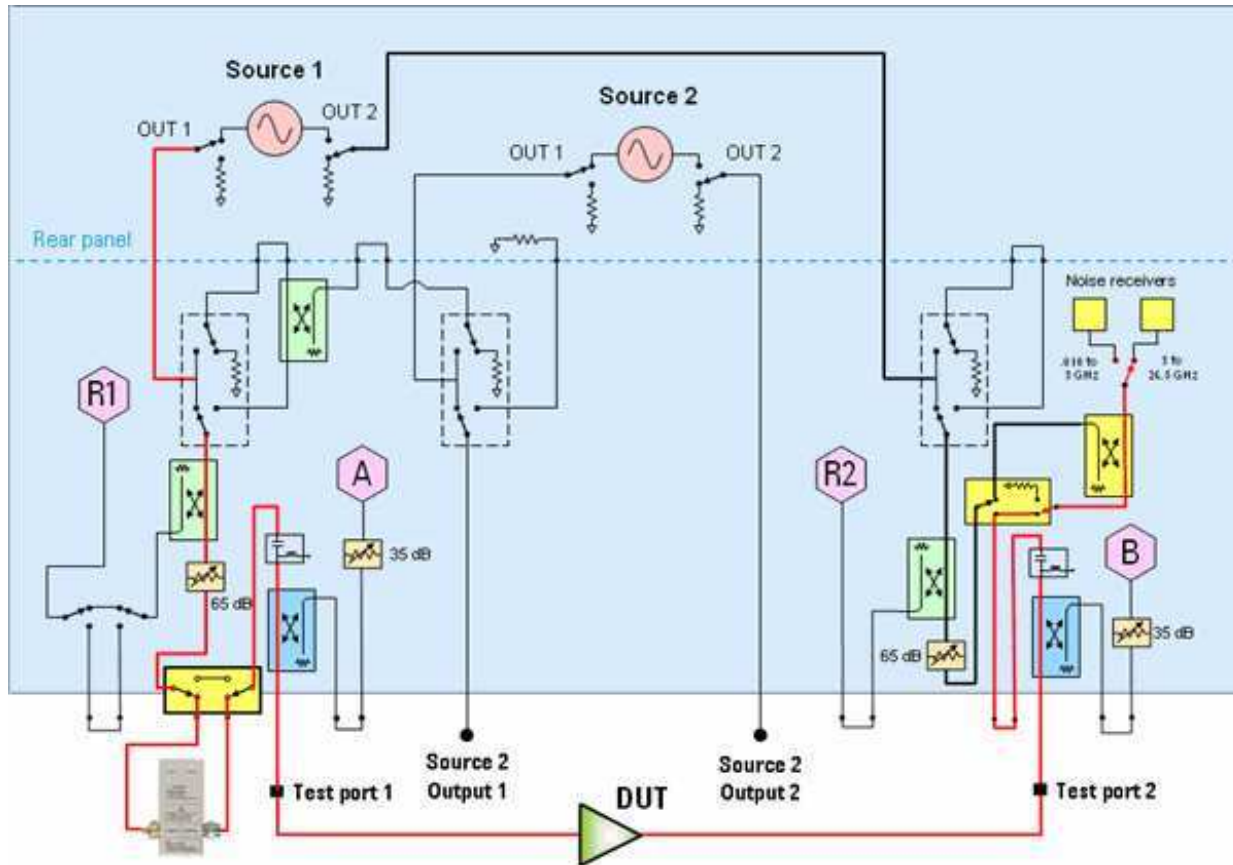
Scalar Noise Figure measurements can be made beginning with PNA-X Rev. A.08.50.

As described above, the noise tuner is switched to at least four different impedance states before a sweep is plotted. These sweeps are NOT made in a Scalar Noise Figure measurement, resulting in much faster sweeps. Of course, a Scalar Noise Figure measurement is NOT as accurate as a Vector Noise Figure measurement because Scalar Noise Figure measurements assumes that all impedances are 50 ohms. Measurement accuracy can be improved by adding an attenuator as close to the DUT input as possible. This improves the effective system source match. The effect of the attenuator loss is removed during the calibration process.

With Scalar Noise Figure, it is not necessary to connect the noise tuner. If a noise tuner remains connected, it is switched to the THRU state for Scalar Noise Figure measurements. This results in a small amount of loss which slightly degrades measurement accuracy. To increase measurement accuracy, manually switch the noise tuner switch to the INTERNAL position. [Learn how.](#)

Select Scalar Noise Figure at the first page of a [Noise Figure calibration](#).

## PNA-X Option 029 - Block Diagram with Noise Figure components



Noise Figure Components are shaded yellow

- At test port 1 front-panel loops, a **noise tuner bypass switch** connects the noise tuner (ECal module) in series with Source1 providing several different input impedances.
- At test port 2, a **switch** and **coupler** to route RF from the DUT output to **two noise receivers**. The appropriate receiver is automatically switched as required for frequency being measured.

### Making S-parameter measurements and the Noise Tuner Switch

The default setting for the port 1 noise tuner bypass switch is EXTERNAL, as shown in the above diagram. This setting always provides incident power through the front panel loop. When an ECal module is connected, it may NOT be in the THRU state, which is necessary for accurate S-parameter measurements. This can be changed in any of the following ways:

- Set the switch to INTERNAL for the S-parameter channel using the [path configuration](#) dialog.
- Set the switch to INTERNAL for the S-parameter channel using the following commands:
  - **SCPI** - [SENS:PATH:CONF:ELEM:STAT](#) Port1NoiseTuner, Internal
  - **COM** - [PathConfiguration.Element](#)(Port1NoiseTuner).Value = Internal
- Set the switch default to INTERNAL using a [preference setting](#).

- Set the Noise Tuner (ECal module) to the THRU state using SCPI: [CONT:ECAL:MOD:PATH:STATE](#).

### Noise Figure App vs Noise Figure Analyzer

In comparing the PNA-X Noise Figure App measurements with the [NFA Series](#) Noise Figure Analyzer measurements, you may obtain different results.

This is because the Noise Figure Analyzer assumes that the noise source has a perfect 50 ohm match. This assumption is often valid for low-ENR sources, but is NOT a good assumption for high-ENR sources, or when there are cables, adapters, switches, probes, etc. between the noise source and the DUT. Using advanced calibration methods, the PNA-X Noise Figure App calculates noise figure as though it were a perfect 50 ohm match.

In addition, the PNA-X measures the amplifier gain with vector error correction applied to reduce measurement uncertainty.

### Noise Power Parameters

Several noise power parameters, as well as standard parameters, can be measured in the same Noise channel.

#### How to add or change Noise Power Parameters

1. Create a Noise Figure channel.
2. Then do the following:

##### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then select a parameter

##### Using a mouse with PNA-X Menus

1. Click **Response**
2. then **Measure**
3. then select a parameter

**Programming Commands**

### Noise Figure Parameters that are offered:

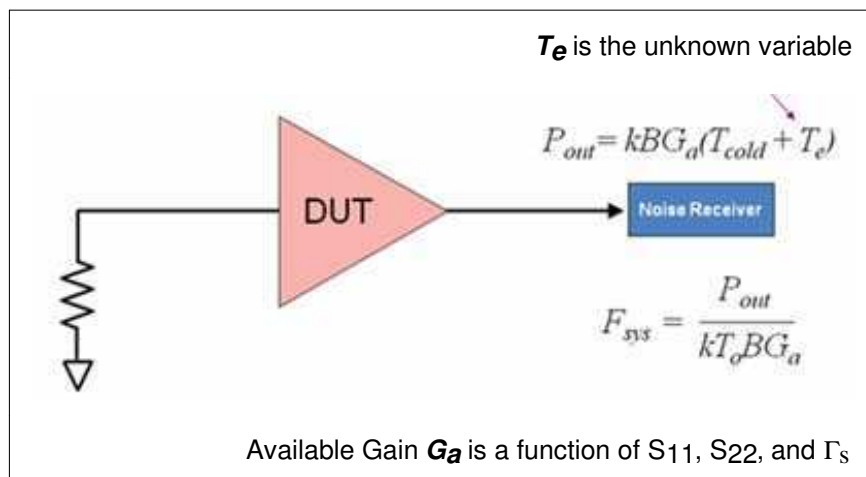
- **Noise Figure (NF)** - Explained in [Noise concepts](#).
- **Excess Noise Ratio** - Select when measuring the noise source. Compare with the ENR table to validate accuracy of the system. ENR is calculated as:

$$\text{ENR (in dB)} = 10 \log_{10}((T_{\text{hot}} T_{\text{cold}}) / T_0), \text{ where } T_0 = 290\text{K}.$$

Learn more about the ENR table and Noise Source. [Learn more about Noise Source ENR measurements.](#)

- **T-Effective** - The effective temperature, in Kelvin, of the measured noise level. For example:

$$290^\circ \text{ K} = -174 \text{ dBm/Hz}.$$



The Noise Power parameters below are offered in the following two formats:

- **Available Noise Power** The calculated power that is based on an ideal impedance match at the output of the DUT. These parameters have always been offered in the PNA-X Noise Figure App.
- **Incident Noise Power** - An 'I' is appended to the end of the Available Noise Power parameter. The calculated power into a perfect 50 ohm noise receiver, regardless of the output impedance of the DUT.

### Noise Power Parameters

- **SYSNPD / SYSNPDI** - System Noise Power Density: Total noise power available at the ADC, including the noise contributed by both the DUT and the internal noise receiver. This is generally expressed as an absolute power measurement in dBm, but can also be expressed in Watts or Kelvin.

$$\text{dBm} = 10 \log_{10}(k * T * B * 1000)$$

where:

$k$  = Boltzmann's constant

$T$  = the measured noise temperature

$B$  = bandwidth

1000 = conversion from milliwatts

- **SYSRNP / SYSRNPI** - System Relative Noise Power: The noise temperature of the combined DUT and receiver relative to 290 Kelvin. This is generally reported as a ratio in dB. Therefore a perfectly quiet device would render a trace at 0 dB.

$$\text{dB} = 10 \log_{10}(T/290)$$

- **DUTNPD / DUTNPDI** - DUT Noise Power Density: When correction is ON, this trace exhibits the available noise power, best described as the maximum power available from the DUT where the impedance of the noise port is equal to the output match of the DUT. To be more precise, this occurs when the noise port match is equal to the conjugate of the output match of the DUT. The noise power contributed by the receiver is removed.

When correction is OFF, the trace exhibits what is more accurately described as delivered power. Delivered power is the power actually seen by the ADC. Any mismatch between the receiver and the DUT is ignored. The noise power contributed by the receiver is removed.

This measurement is generally expressed in dBm:

$$\text{dBm} = 10 \log_{10}((\text{DUT Temperature} - \text{Receiver Temperature}) * B * 1000)$$

where:

B = bandwidth

1000 = conversion from milliwatts

- **DUTRNP / DUTRNPI** - DUT Relative Noise Power: This measurement is rendered as a ratio of the DUT temperature to 290 Kelvin. It is generally expressed in dB. The same comments apply with respect to available versus delivered power as described above for DUTNPD.

$$\text{dB} = 10 \log_{10}(\text{DUT Temperature} - \text{Receiver Temperature})$$

### Standard Parameters that are offered

- **S-parameters:** S11, S21, S22, S12
- **Unratioed parameters** using the following notation: (Receiver, source port). These parameters REPLACE the active measurement. To do this (from front-panel ONLY), press **MEAS**, then **[More]**, then **[Receivers]**.
  - (R1,1), (R2,2), (A,1), (A,2), (B,1), (B,2)

### Using the Noise Figure Application

Use the following general procedure to make Noise Figure measurements:

1. [Connect Tuner and Noise Source.](#)
2. [Create a Noise Figure Measurement.](#)
3. [Make Noise Figure Settings.](#)
4. For Opt 029 and H29, copy your Noise Source ENR file to the PNA C:/Program Files/Agilent/Network Analyzer/Noise folder.
5. [Perform Calibration](#)

6. Connect the DUT:

- **Opt 029** - DUT input to PNA-X port 1. DUT output to PNA-X port 2. For highest Noise Figure accuracy, there should be the least amount of electrical loss possible between the DUT output and the PNA-X Port 2.
- **Opt 028** - Use any PNA-X ports for the DUT input and output. Select the ports on the [Noise Figure setup tab](#).

7. Measure Noise Figure.

8. **Optional** Click File, then Save to save Noise Figure data in the following [formats](#): (available ONLY when NF correction is ON.)

- \*.CTI Citifile
- \*.PRN
- \*.nco Noise Correlation Matrix data in S2P format. [See Noise Model](#).

See Also: [Measurement Tips](#)

### Connect Noise Tuner and Noise Source

1. Connect the **noise source** to the [28V connector](#) on the PNA-X rear panel. **NOT required for Opt 028**. The Noise Source is turned ON and OFF automatically as needed during a calibration. Connect the noise source to Port 2 reference plane when prompted during calibration.
2. Connect the **noise tuner** (ECal module). NOT required for [Scalar Noise Figure](#) measurements.
  - a. On the PNA-X front panel, remove the **Port 1** jumper cable SOURCE OUT / CPLR THRU. Opt 028 allows Noise Figure measurements using any two PNA ports. Connect the noise tuner to the front-panel jumpers for the source (DUT input) port.
  - b. Connect M-F tuner (N4691B-M0F) using the supplied cable (N5242-20137) and adapter (85052-60013).
  - c. When using F-F ECal module (N4691B-00F), order a 3.5 mm M-M adapter (85052-60014).



### Create a Noise Figure Measurement

1. On the PNA-X front panel, press **Meas**, then **[Measurement Class]**
2. Select **Noise Figure Cold Source**, then either:
  - **OK** delete the existing measurement, or
  - **New Channel** to create the measurement in a new channel.
3. A Noise Figure measurement is displayed. To select additional parameters to display, click **Response**, then **Measure**, then select a parameter from the list.

### How to start the Noise Figure Setup dialog

To provide quicker access, use the Setup softkey. [Learn how.](#)

#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[Noise Figure Setup]**

#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Measure**
3. then **Noise Setup**

### Programming Commands

The screenshot shows the 'Noise Figure Setup' dialog box with the following settings:

- Bandwidth/Average:** Noise Bandwidth: 4.0 MHz, Average Number: 1, Average ON: ☐
- Noise Receiver:** ☐ NA Receiver (Port 2), ☒ Noise Receiver
- Receiver Gain:** ☒ High, ☐ Medium, ☐ Low
- Ambient Temperature:** 295.00 K
- Impedance States:** Noise Tuner: None selected, Max Acquired Impedance States: 4

### Noise Figure Setup dialog box help

**Note:** In this topic, the term **Jitter** is used to describe the trace-to-trace fluctuations in a measurement. In other topics, this is called 'trace noise'.

#### Bandwidth/Average

The following two settings work together to achieve the optimum balance of measurement accuracy versus speed. Both settings can be changed after calibration to make faster measurements with minimal effect on calibration accuracy.

**Noise Bandwidth** Increase the bandwidth to make faster measurements. However, a wider setting reduces the frequency resolution of the measurement. More frequencies are essentially smoothed together to produce a flatter response, which could hide the actual noise performance of the DUT.

**Noise Averaging Factor** Increase the number of averages to reduce jitter. This also increases measurement speed. For maximum accuracy, increase the averaging factor for the noise calibration. It can then be reduced to improve measurement speed.

### Noise Receiver

**NA (Network Analyzer) Receiver** (Opt 028) - Use a standard PNA receiver to measure noise figure.

- Connect the DUT to any PNA-X ports. For Vector NF measurements, [connect the noise tuner](#) to the source port.
- The sum of the gain and noise figure of the DUT must be at least 40 dB. This ensures that there is sufficient DUT noise power for the PNA-X to measure. [Learn more.](#)
- Additional filtering may be required. [Learn more.](#)

**Noise Receiver** (Opt 029) - Use internal low-noise receivers to measure noise figure.

- Opt 029 measures devices with Noise Figure values ranging from about 0 to 50 dB and devices with GAIN ranging from about -40 to +60 dB. [Learn more.](#)
- Amplifiers with higher gain can be measured by adding an attenuator to the output of DUT and using [fixture de-embedding](#) to remove the attenuator loss. An alternative for measuring high-gain devices is to use the standard receivers (Opt 028) as they have a higher compression level.

### Receiver Gain

This setting is NOT available when Noise Receiver is set to **NA (Network Analyzer) Receiver** (Opt 028).

With knowledge of your DUT gain, set the appropriate amount of receiver gain in order to optimize the power level at the noise receiver.

The following values reflect the SUM of the DUT gain (dB) **PLUS** NF (dB). For example: DUT gain = 20 dB; NF = 10 dB; SUM = 30 dB.

- Select **High** if the SUM is relatively low (<30 dB).
- Select **Medium** if the SUM is about average (20 dB to 45 dB).
- Select **Low** if the SUM is relatively high (>35 dB).

There is considerable overlap in these settings. Because all three gain settings are calibrated with each Noise Calibration, this setting can be changed after calibration to achieve the least amount of jitter without overpowering the noise receiver.

One of following messages appears when too much power is detected at the noise receiver:

- **Compression in noise receiver: excess signal** - The noise receiver is likely compressing. NF results are possibly not accurate. Select a lower gain setting.



- **Compression in noise receiver: gain has been limited** - The gain has been limited to avoid damage to the receiver. NF results are NOT accurate. Select a lower gain setting.
- **ADC over-range in noise receiver: excess signal** - Often caused by a CW signal, an oscillation, or LO feedthru during an NF measurement. Find and correct the cause, or try a lower gain setting.

Only ONE gain setting can be used for the entire frequency range of your noise measurement. Therefore, it may be necessary to use two noise channels with different frequency ranges and gain settings to achieve the very highest noise figure accuracy.

### Ambient Temperature

Enter the room temperature at the time of the measurement, in Kelvin. For best results, use a thermometer to read the temperature at the PNA-X test port 1 or the DUT input cable.

This ambient temperature number has an inverse relationship to the noise figure. When using the effective noise temperature ( $T_e$ ) format, a 3 degree increase in the ambient temperature will make the overall measurement result drop 3 degrees.

### Impedance States

**Noise Tuner** Displays the ECal module to be used as a noise tuner. Select the Noise Tuner during calibration on the [Select Cal Method](#) dialog.

**Max Acquired Impedance States** Select the number of impedance states in which to make noise measurements. At least FOUR impedance states are required. [Learn more](#)

The screenshot shows the 'Frequency' tab of a measurement setup interface. It includes a 'Sweep Type' section with radio buttons for Linear Sweep (selected), Log Sweep, Power Sweep, CW Frequency, and Segment Sweep. Below this is a 'Sweep Settings' section with input fields for Number Of Points (201), IF Bandwidth (1.000 kHz), Start (10.000000 MHz), Stop (26.50000000 GHz), Center (13.255000000 GHz), and Span (26.490000000 GHz).

## Frequency Tab - Noise Figure dialog box help

These settings can also be made from the normal PNA-X setting locations. Click links below to learn how.

### Sweep Type

Choose a sweep type. [Learn more.](#)

#### Segment Sweep Notes:

- The segment table shown on the dialog is **'READ-ONLY'**.
- Learn how to [Create and edit the Segment Sweep table](#).
- **Independent IFBW** and **Power** are NOT available.
- [X-axis point spacing](#) is available beginning with A.09.10.

### Sweep Settings

Click each to learn more about these settings.

- [Number of points](#)
- [IF Bandwidth](#) This setting is important for improving noise measurement accuracy. [Learn more.](#)
- [Start / Stop](#), [Center / Span](#) frequencies.

## Power Tab - Noise Figure dialog box help

**Note:** S-parameter power settings are critical for accurate Noise Figure measurements. [See Noise Figure Measurement Tips.](#)

Configures RF power settings for the S-parameter measurements that occur before noise measurements. Input power to the DUT is turned OFF during noise measurements.

These settings can also be made from the normal [Power setting](#) locations.

**Power ON (All channels)** Check to turn RF Power ON for all channels.

### DUT Input Port

Opt 028 -Select a PNA port to be connected to the DUT input.

**Note:** Input power levels are critical for accurate Noise Figure measurements. [Learn more.](#)

**Power Level** The input power to the DUT during S-parameter measurements.

**Source Attenuator Auto** Check to automatically select the correct attenuation to achieve the specified input power. Clear, then select attenuator setting that is used achieve the specified Power Level. [Learn more about Source Attenuation.](#)

All PNA channels in continuous sweep must have the same attenuation value. [Learn more.](#)

**Receiver Attenuator** Specifies the receiver attenuator setting for port 1.

**Source Leveling** Specifies the leveling mode. Choose Internal. Open Loop should only be used when doing [Wideband Pulse measurements](#) (not available with Noise figure measurements).

### DUT Output Port

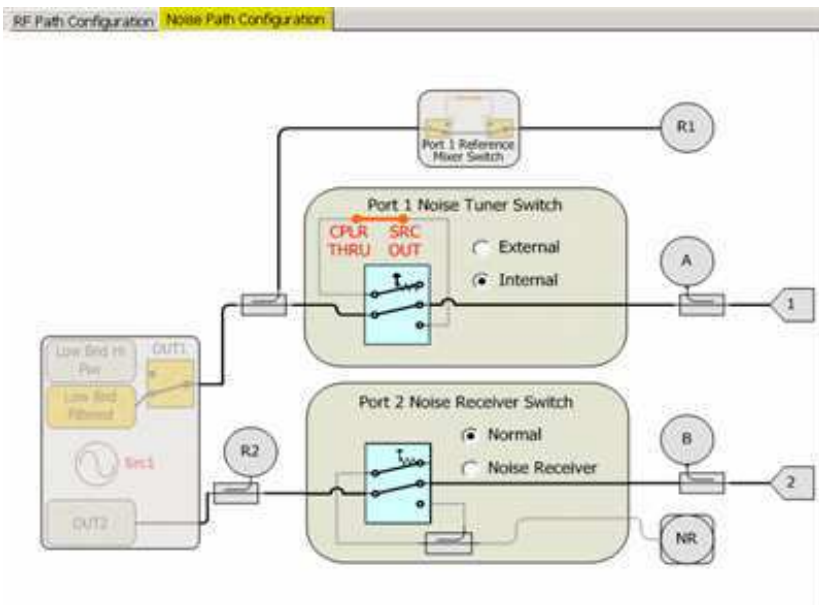
Opt 028 -Select a PNA port to be connected to the DUT output.

**Output Power** Sets power level in to the output port for reverse sweeps. Port power is automatically uncoupled. Reverse sweeps are always applied to the DUT when Full 2-port correction is applied. [Enhanced Response Cal](#) is NOT available for Noise Figure measurements.

**Source Attenuator** Specifies the source attenuator setting for reverse power.

**Receiver Attenuator** Specifies the receiver attenuator setting for the output port. [Learn more about Receiver Attenuation.](#)

**Source Leveling** Specifies the leveling mode. Choose Internal.



### Noise Path Configurator dialog box help

**Port 1 Noise Tuner Switch (Opt 029)** The orange line between CPLR THRU and SRC OUT represents the Noise Tuner. The **External** setting switches IN the Noise Tuner when making noise measurements.

**Port 2 Noise Receiver Switch (Opt 029)** allows you to make Noise Receiver measurements.

To prevent premature wear on the above two Noise switches, the PNA does not allow these switches to be thrown when sweeping a Noise channel and non-Noise channel. To make Noise Figure measurements and non-Noise Figure measurements in different channels and continuously trigger both, set these switches to the same state as the Noise channel:

- With the **non-Noise Figure channel** active, go to [Noise Path Configurator](#).
- Set Noise Tuner switch to **External**. This routes source power to the front-panel loops, and to the Noise Tuner when connected. Use [CONT:ECAL:MOD:PATH:STATE](#) to set the internal state of the Noise Tuner to THRU, which creates a small amount of additional loss in the source path.
- Set Noise Receiver Switch to **Noise Receiver**.

### Noise Figure Measurement Tips

**Note:** In this topic, the term **Jitter** is used to describe the trace-to-trace fluctuations in a measurement. In other topics, this is called 'trace noise'.

#### Option 029

[See Opt 028 \(NF with Standard Receiver\)](#)

- measures devices with Noise Figure values ranging from about 0 to 50 dB and devices with GAIN ranging from about -40 to +60 dB.
- Highest Noise Figure accuracy is attained when the sum of device Noise Figure + GAIN is between 0 dB to

+70 dB.

- For highest Noise Figure accuracy and stability, there should be the least amount of electrical loss possible between the DUT output and PNA Port 2.

### Power level at the DUT Output - Opt 029

S-parameters are used to measure the gain of the DUT before each series of noise measurements. Jitter in the S-parameter measurements corresponds directly to jitter in the noise measurements.

For best noise figure accuracy, the power level out of the DUT should be between 15 dB and 20 dB below the compression point of the DUT during the S-parameter portion of the noise figure measurement.

To reduce jitter, the power level at the B receiver (port 2) should be above approximately -20 dBm. Much below this level, S-parameter measurements have more jitter. Power must be below +10 dBm as the B receiver starts to compress at this point, although there is no warning or annotation that shows this condition is occurring in S-parameter measurements.

The best way to monitor power at the B receiver is to display a [B,1 measurement](#). With your DUT in place and powered ON, change the input power to the device and note the power at the B receiver.

- For low-gain amplifiers, use 5 dB of source attenuation to improve the uncorrected match of port 1.
- For high-gain amplifiers, source and receiver attenuation may be required. Use the lowest possible attenuation values.

Attaining the optimum power level **during calibration** can also be challenging since calibration is performed without the DUT in place. Because of this, it is often necessary to set source power higher during the calibration than during the measurement. This will cause the '[C](#)' annotation on the status bar. Measurement results are accurate as long as the step attenuators and other configuration switches are in the same position and all receivers remain in their linear range (below +10 dBm).

It is best to find the optimum power and attenuation settings for both the calibration and subsequent noise measurements **before** performing a calibration.

### IF Bandwidth

Jitter is further reduced by narrowing the IF bandwidth. If the calibration needs to be performed at a low source power, or with receiver attenuation due to high DUT gain, the IF bandwidth should be reduced during the calibration to reduce jitter. The IF bandwidth can then be increased to improve measurement speed. The [C annotation](#) can be ignored when changing IFBW after calibration.

### Noise Settings

See [Noise Figure dialog box help](#) for a complete description of these important settings.

### Temperature

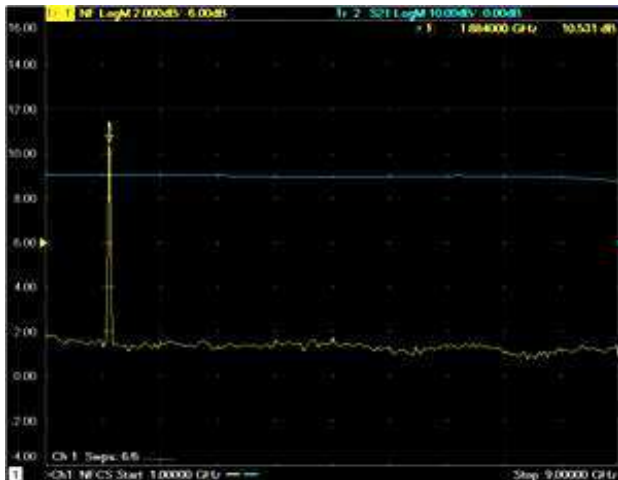
Noise Figure measurements are extremely sensitive to temperature. As such, there are two settings that require an accurate temperature measurement: At the [DUT input](#), and at the [Noise Source connector](#).

### Interference

When measuring the noise figure of an unshielded device, like an amplifier on a printed-circuit board, it is very common to pick up interference from external signals such as cellular phones, wireless LAN, or mobile radios. This

interference shows up as non-repeatable spikes in the measurement, as shown below.

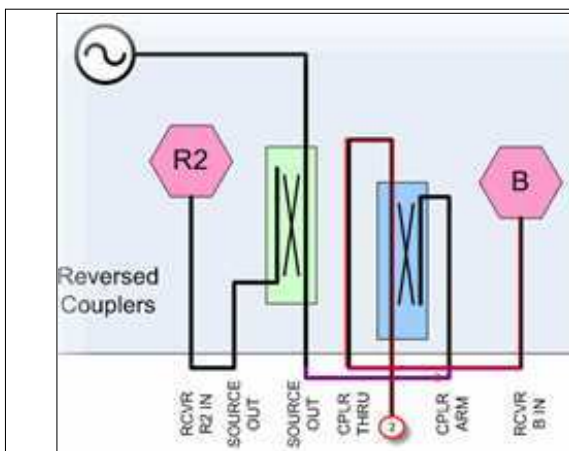
Usually, the interference adversely affects the noise figure measurement only at the frequency where it occurs. However, if the interference is large enough and present all of the time, it can cause the noise receivers to compress, which results in inaccurate measurements at many frequencies. In this case, the noise figure measurements should be done in a shielded environment like a screen room.



### Option 028

**Noise Figure of PNA receiver** - Option 028 gives you the flexibility to measure noise figure using a standard PNA-X receiver. For best measurement accuracy, the DUT excess noise power (gain plus noise figure in dB) should meet or exceed the noise figure of the receiver (typically about 40 dB, reference to the test port). For devices that do not meet this requirement, there are two alternatives:

1. Reverse the main arm and coupled arm of the test-port coupler (see following images). This increases the signal to the receiver port by about 15 dB, while lowering the available port power by the same amount. This is a good tradeoff for noise figure measurements, as it lowers the excess noise threshold mentioned above from 40 dB to about 25 dB.

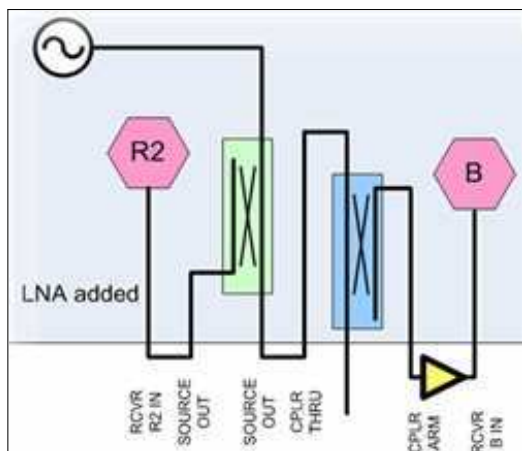


Block diagram showing port 2 through coupler main arm to B receiver.



Re-configure the **receiver port** front-panel loops to a vertical orientation.

2. Add an low-noise amplifier (LNA) to the receiver loop (see following image). This boosts the noise power at the receiver by the gain of the LNA. The disadvantage is the possibility of measurement drift and receiver compression. Any change in the gain of the LNA will have an impact on measurements that use the receiver with the LNA, so frequent calibration may be required. Care should also be taken when setting the channel power (used during the gain portion of the measurement) to ensure that the added gain of the LNA does not cause receiver damage or compression. A filter is also required on the output of the LNA (not shown). [Learn more](#).



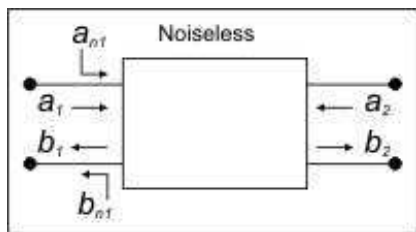
### Filtering Requirement (Option 028)

Opt 029 includes noise receivers with filtering to keep mixing-product noise out of the low-noise receivers. These filters are not available when measuring noise with the standard PNA-X receiver. Therefore, for best measurement accuracy, a filter should be used at the output of the DUT (or LNA preamp if used).

- A bandpass filter at the frequencies of interest can always be used.
- A lowpass filter can be used when the PNA-X is doing fundamental mixing (up to 26.5 GHz). The lowpass filter must pass the fundamental frequency of the measurement but suppress the third harmonic. A measurement at 1 GHz would need a lowpass filter with a cutoff below 3 GHz, while a 5 GHz measurement would need a filter with a cutoff below 15 GHz.
- A single highpass filter can be used when the PNA-X is doing 3rd harmonic mixing (from 26.5 to 50 GHz). Use a highpass filter with cutoff about 18 GHz.

### Noise Model and the Noise Correlation Matrix

The noise wave model of any linear 2-port network may be represented by the following image:



This shows a noiseless 2-port network with noise waves ( $a_{n1}$  and  $b_{n1}$ ) added to the input terminals.

The noise correlation matrix relates to the noise waves as follows:

$$C_t = \begin{bmatrix} \overline{|a_{n1}|^2} & \overline{a_{n1} b_{n1}^*} \\ \overline{b_{n1} a_{n1}^*} & \overline{|b_{n1}|^2} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

Where:

- $\overline{|a_{n1}|^2}$  and  $\overline{|b_{n1}|^2}$  are time-averaged noise power in 1 Hz bandwidth.
- $\overline{a_{n1} b_{n1}^*}$  and  $\overline{b_{n1} a_{n1}^*}$  are time-averaged cross correlation terms, correlation of  $a_{n1}$  to  $b_{n1}$ .

### Radio-Frequency Electromagnetic Field Immunity

When a 3Vm-1 radio-frequency electromagnetic field is applied to an PNA-X with Opt 029 according to IEC 61000-4-3:1995, degradation of performance may be observed. When the frequency of the incident field matches the frequency of a measured noise figure or gain, the values displayed will deviate from those expected. This phenomenon will only affect that specific frequency, and the analyzer will continue to perform to the specification at all other frequency sample points.

The PNA-X with Opt 029 may be unable to calibrate a chosen frequency sample point if the frequency matches that of an incident electromagnetic field.

Last Modified:

28-Apr-2010	Added port mapping (9.22)
12-Mar-2010	Added 028 and H29 - removed some fixturing limitations (9.20)
19-Nov-2009	Added X-axis point spacing (9.1)
31-Jul-2009	Added interference tip
15-Jul-2009	Added reference to other tuner models
6-Apr-2009	Replaced N5242a with PNA-X
25-Feb-2009	Added Incident parameters and Scalar (A.08.50)
18-Feb-2009	Added Noise model
29-Sep-2008	Added Port Ext to NOT supported list
22-Apr-2008	Added link to preferences
12-Mar-2008	Modified Noise Params
29-Jun-2007	MX New topic



## Noise Figure and NFX Calibration

---

This topic discusses calibration for both Noise Figure on Amplifiers and Noise Figure on Converters (NFX).

- [Overview](#)
- [How to Perform a Noise Figure Cal](#)
  - [Select Calibration Method](#)
  - [Configure Noise Source](#)
  - [Select DUT Connectors and Cal Kits](#)
  - [Measure Standards Steps](#)
  - [Validate Noise Source Cal](#)

See [Noise Figure](#) or [NFX Applications](#)

### Overview

Noise Figure Cal is very similar for both amplifiers and converters (NFX).

- NFX does NOT offer '[S-params ONLY](#)' calibration.
- NFX includes an optional [LO Power Cal](#).
- NFX and [NF with a standard receiver](#) includes a [Power Sensor Cal](#) on the source port.

**Note:** Noise Figure results are NOT at all accurate without a Noise Figure Calibration.

Before a Noise Figure calibration can be performed, a Noise Figure measurement must be active.

For Vector Noise Figure calibration, the Noise Figure Tuner must be connected to the PNA during calibration.

The following explains how a Noise Figure Calibration works:

To begin the Noise Figure Cal, a standard 2-port calibration is performed on the Noise Figure channel. This is because S-parameters are measured at each frequency step before a noise measurement.

Following the 2-port cal, the noise receivers are calibrated using a Noise Source. A Noise Source is a device that is specifically designed to generate two very consistent levels of noise over its operating frequency range:

1. Hot (On) the Noise Source is biased in order to provide a high level of noise.
2. Cold (Off) the Noise Source is unbiased to provide ambient temperature noise level.

These levels are measured by the Noise Source manufacturer and provided in table and electronic format with each Noise Source by serial number. The electronic file is known as the ENR (Excess Noise Ratio) file.

During the PNA calibration with Opt 029, the Noise Source is connected to the PNA noise receiver through test port 2. Each frequency and noise level is measured by the noise receivers. The differences between the known ENR noise levels and the measured noise levels are the noise error terms. These values are removed from subsequent noise measurements.

**Note:** For highest accuracy, the noise source should be connected as close as possible (the least amount of electrical loss) to the PNA port 2 connector. This causes the largest difference between the Noise Source HOT (on) and COLD (off) settings.

During the Noise Source measurements, noise averaging and noise bandwidth is automatically turned ON to the values that you specify. [Learn more about Noise Averaging.](#)

After calibration, correction is automatically turned ON. The PNA [status bar](#) shows **VNC\_2P** (for Vector) and **SMC\_2P** (for Scalar).

### How to Perform a Noise Figure Calibration

With a noise figure measurement already present:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **CAL**
2. then **[Cal Wizard]**

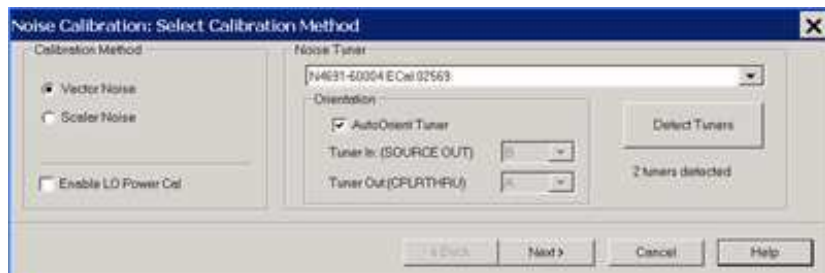
#### Using a mouse with PNA Menus

1. Click **Response**
2. then **Cal**

**Programming Commands**

The following Cal Wizard pages are unique to Noise Figure Calibration. The remaining pages that are presented are the same as those in the standard [Cal Wizard SmartCal](#).

### Select Calibration Method dialog box help



#### Calibration Method

- Vector Full - Comprehensive Noise Figure calibration
- S-Parameter Only - **Amplifiers ONLY**. Does NOT calibrate the noise receivers.
- Scalar Full - Noise Figure calibration for Scalar Noise Figure measurements. [Learn more.](#)

**Enable LO Power Cal - NFX ONLY.** Check to cause the Cal Wizard to guide you through a Power Calibration

on the LO source.

**Note:** NO correction is provided for an adapter that may be used to connect the power sensor to the LO source.

### Noise Tuner

Not available when Scalar Full is selected.

Select from the ECal modules that are connected to the USB.

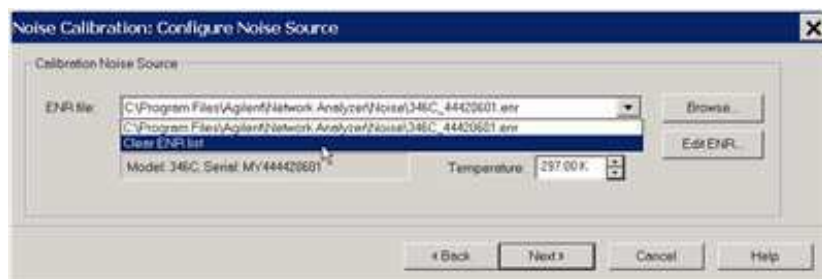
### Orientation

**AutoOrient Tuner** Check to allow the noise tuner orientation to be auto-detected. When cleared, use the following two fields to provide manual orientation of the noise tuner.

**Tuner In (SOURCE OUT) / Tuner Out (CPLRTHRU):** Specify the ECal module labels that are connected to the PNA-X front panel jumper connectors. [Learn how to connect the noise tuner.](#)

**Detect Tuners** Click to re-detect the Noise Tuners (ECal modules) that are connected to the USB. If the ECal module is not detected, check the USB connection, then click this button. The label below the button indicates the total number of ECal modules that are connected to the USB.

### Configure Noise Source (Opt 029) dialog box help

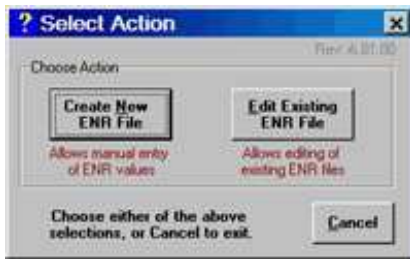


**ENR File** Select the Noise Source ENR file. If not already there, copy your Noise Source ENR file to the PNA C:/Program Files/Agilent/Network Analyzer/Noise folder. Then click Browse to find the ENR file.

**Clear ENR List** Scroll to the bottom of the ENR list, then click to removes the selected ENR file. Then browse or select to find a new file.

**Edit ENR** Click to launch the [ENR Editor](#) dialog box which is used to change or create ENR files. This is NOT usually necessary.

**Temperature** Specify the current temperature at the Noise Source connector. The Noise source is kept ON during Noise Figure measurements. This results in the Noise Source being 4 to 5 degrees warmer than Ambient temperature, and a more accurate calibration. See [Noise Figure tips](#) to learn more about the significance of temperature.



Click either Create or Edit to launch the same dialog box, shown below.

- **Edit** populates all fields with existing data which can then be edited and stored.
- **Create** has empty fields except for frequencies.

#### Edit / Create ENR File dialog box help

The 'ENR Editor' dialog box has a title bar with a pencil icon and 'x'. It is divided into two main sections: 'ENR Numeric Data' and 'Identifying Data'.  
**ENR Numeric Data:** Includes fields for 'Entry #', 'Frequency (MHz)', and 'ENR (dB)'. The 'Entry #' field shows '1 of 29'. The 'Frequency (MHz)' field shows '10'. The 'ENR (dB)' field shows '13.816'. There are 'Done', 'Prev', 'Next', and 'Enter' buttons. A red text note says: 'Use Prev/Next keys to make any changes to the ENR values for each listed frequency. When finished editing all ENR data, click "Done".'  
**Identifying Data:** Includes fields for 'Model #', 'Serial #', 'Temperature', 'Humidity', and 'Cal Date'. The 'Model #' field shows '346C'. The 'Serial #' field shows 'MY44420700' with '(required)' next to it. The 'Temperature' field shows '24 °C' and the 'Humidity' field shows '40 %'. The 'Cal Date' field shows '20070731' with 'yyyymmdd' below it. There are 'KeyBd' and 'Store ENR File' buttons.  
 On the right side, there is a list of frequencies and ENR values. At the bottom, there are 'Help' and 'Cancel' buttons. The version 'Revision A.01.00' is shown in the bottom left.

MHz	ENR
10	13.816
100	13.642
1000	13.536
2000	13.321
3000	13.282
4000	13.316
5000	13.425
6000	13.609
7000	13.862
8000	14.142
9000	14.49
10000	14.71
11000	14.792
12000	14.993
13000	14.934
14000	15.041
15000	15.019
16000	14.961
17000	14.747
18000	14.679
19000	14.597
20000	14.65
21000	14.914

#### ENR Numeric Data

Use **Previous** and **Next** buttons to scroll to **Entry #** to edit. Type **ENR** value in dB, then press **Enter**.

**Done** Click when finished editing all values. Then click **Store ENR File** to save the file.

#### Identifying Data

**Model #** of the Noise Source. This can NOT be changed.

**Serial #** of the Noise Source.

**Temperature and Humidity** in which the Noise Source was calibrated. This is for information only. The ENR data is always normalized to 290 Kelvin.

**KeyBd** launches a mouse-driven keyboard.

**Store ENR File** Click to launch a dialog to save the new or edited ENR file.

## Select DUT connectors and Cal Kits dialog box help



### Port 1 and Port 2

**DUT (Device Under Test) Connectors** Specify the connector and gender of the **DUT**.

**Cal Kits** Select the Cal Kit to be used to calibrate each test port. The list for each DUT Port displays kits having the same connector type as the DUT. Using incorrect calibration standards can significantly degrade measurement accuracy. [Learn more.](#)

**Power Sensor** **NFX Cal** and **NF with a standard receiver ONLY**. Used to calibrate the source port.

**Noise Src** Used to calibrate the noise receivers. The Agilent 346C has an "APC 3.5 male" connector.

**Note:** For highest accuracy, the noise source should be connected as close as possible to the PNA port 2 connector. This causes the largest difference between the Noise Source HOT (on) and COLD (off) settings.

For both Cal devices (power sensor and noise source, specify the connector type and gender. When the Cal device connector is **NOT** the same type and gender as the DUT Port connector, then for optimum accuracy, extra cal steps are used to measure and correct for the adapter that is used to connect the Cal device to the reference plane.

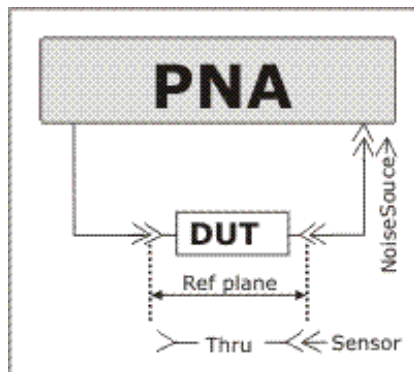
Select **Ignored** (at the bottom of the DUT Connectors list) to NOT compensate for the adapter.

Select the Cal Kit that will be used for that process.

**De-embed power sensor adapter / noise source adapter** The PNA uses the connector type and gender of the DUT along with the connector type and gender of the cal device to determine if an adapter removal operation is taking place AND whether or not that removal operation requires an additional cal step.

However, the use of the connector type can, in special cases, hide the need for the extra cal step. Check the "De-embed..." box in these cases to inform the PNA that the extra step is needed.

Such a case is illustrated below where the noise source is connected close to test port 2 for higher accuracy. If unchecked, the PNA would assume in this case that the Noise Source is connected to the Thru standard at the port 1 (DUT input) reference plane.



**Source Cal Settings** Click to launch the [Source Power Cal \(for apps\)](#) dialog. This dialog is used to set Power Meter / Sensor settings for both the Port 1 Power Cal, and the optional LO Power Cal.

**Modify Cal** Check, then click **Next**, to Modify Cal (Standards AND Thru Method).

**Note:** [Enhanced Response Calibration](#) is NOT supported with Noise Figure.

### Measure Standards Steps dialog box help



**Power Level** at which to perform the Port 1 Power Cal.

It is usually best to set power level to 0 dBm at the power sensor because the power sensor is calibrated at that level. Lower power levels will yield a slower and noisier calibration.

However, with 20 dB of source attenuation (default NF setting), the PNA may not be capable of achieving this power level at higher frequencies. To check the max leveled power, view an [R1 \(port 1 reference receiver\) trace](#) over the frequency range of interest, then increase the power until roll-off appears. Power levels at the test port may be approximately 2 dB lower than at the R1 receiver.

If an external component is used between the PNA-X test port and the calibration reference plane, then adjust the power level so that the power at the sensor is about 0 dBm if possible.

The current source attenuation value is shown on the dialog.

**LO Power Cal (Optional)** When [enabled](#), perform a Source Power Cal at the DUT LO input. An LO must already be selected. [Learn how](#). The power level of the LO source calibration is set on the [NFX \(LO\) Power Tab](#).

### Connect Noise Source to the Port 2 measurement (reference) plane

When the ["De-embed Adapter.."](#) boxes are checked, additional cal steps are required.

### Subsequent Steps

**Connect Port 1 to Port 2** - Connect port 1 reference plane to the port 2 reference plane using the required Thru standard or adapter.

**Connect ECal to Ports 1 and 2** - Connect the ECal module between the port 1 reference plane and the port 2 reference plane.

After calibration, correction is automatically turned ON. The PNA [status bar](#) shows VNC\_2P (for Vector) and SMC\_2P (for Scalar).

### Validate Noise Source Cal

To validate a Noise Source Calibration, connect the Noise Source to Port 2 and measure [ENR](#).

Compare the measured values to the values in the ENR table.

### How to manually turn the Noise Source ON | OFF

1. Press **POWER**
2. then **[Noise Source ON | OFF]**

---

#### Last Modified:

30-Apr-2010	Added Opt 028 and port mapping
21-Oct-2009	Added NFX Cal
27-Feb-2009	Added Scalar (A.08.50)
9-May-2008	Added validation
23-Aug-2007	New topic

## Swept IMD and IM Spectrum Concepts

---

- [Swept IMD Concepts](#)
- [Swept IMD for Converters](#)
- [Swept IMD Parameters](#)
- [How the PNA-X Measures IMD](#)
- [How an IM Spectrum Channel Works](#)
- [IM Spectrum Parameters](#)

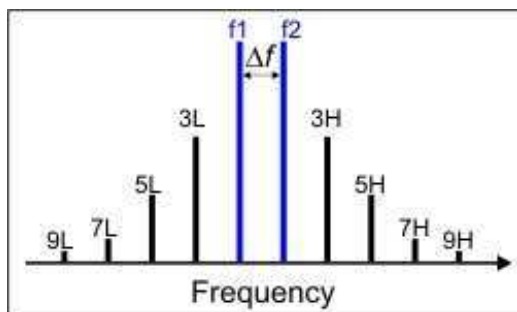
### Other IMD topics

## Swept IMD (Intermodulation Distortion) Concepts

When a device or system is subjected to multiple input frequencies, the non-linearity of the DUT can generate undesired outputs at other frequencies. Typically, two input tones of equal power separated in frequency by a specified amount are used to stimulate the device while observing the resulting frequency spectra at the output. A variety of measurements can then be utilized to determine the intermodulation distortion characteristics of the device.

The frequencies of the resulting distortion products are predictable. While many mixing products can be generated, the high and low signals of the "odd order" products (3rd, 5th, and so forth) are close enough to the original two signals to potentially interfere with adjacent communication channels. With the exception of the 2nd order product, the higher "even order" products are usually far enough away to be of no interest.

The following image and table shows two equal-power main-tones and the nearby odd-order distortion products, as well as the 2nd order product (not shown in the image). Notice that the frequency separation between adjacent odd-order products is the same as the separation of the main tones (Delta F) frequency. For most devices, these distortion products become worse as the device is pushed further into compression.



**Two main tones ( $f_1$  and  $f_2$ ) with odd order intermodulation products.**

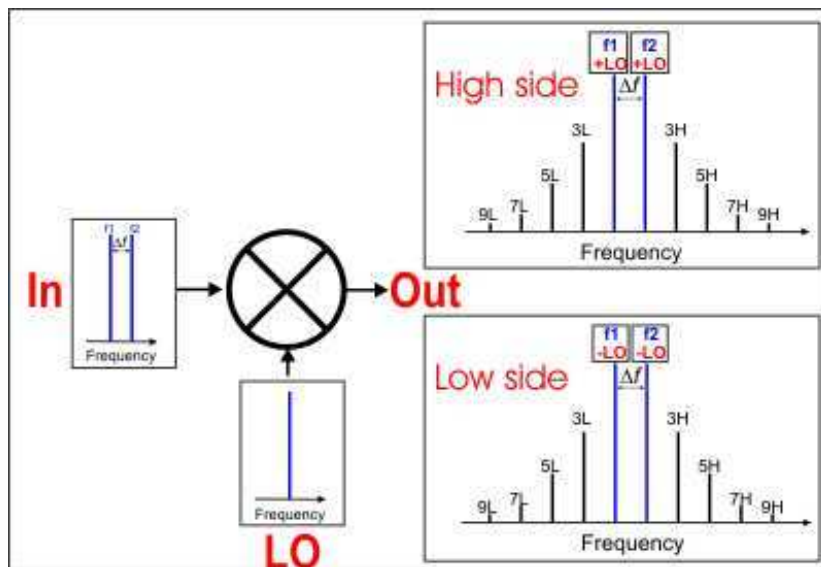
The following table shows the calculations and example frequencies ([Blue text](#)) of the intermodulation products that are closest to the two main tones.



Product	Low (L)	High (H)
Main tones	f1 100 MHz	f2 120 MHz
3	2f1-f2 80 MHz	2f2-f1 140 MHz
5	3f1-2f2 60 MHz	3f2-2f1 160 MHz
7	4f1-3f2 40 MHz	4f2-3f1 180 MHz
9	5f1-4f2 20 MHz	5f2-4f1 200 MHz
2	f2-f1 20 MHz	f1+f2 220 MHz

## Swept IMDx Concepts for Converters

The IMDx concepts for Converters is like IMD Concepts for Amps, except that there are two sets of products on the DUT output.




This image shows the simplest measurement configuration: the Input, LO, and Output are all CW frequencies.

- DUT **INPUT** - two fundamental tones (f1 and f2)
- DUT **LO** - a single frequency.

- DUT **OUTPUT** - two sets of frequencies

In IMDx for Converters, you can view EITHER the High side or Low side products.

	High side (Input <b>PLUS</b> LO and all ordered products)
	Low side (Input <b>MINUS</b> LO and all ordered products)

Make this selection on the [Mixer Frequency tab](#).

## Swept IMD Parameters

The following basic parameters, offered for both Amps and Converters, are expanded to over 150 by selecting specific product tones (2,3,5,7,9), the Low-side, High-side, or Average of these tones, measured at the Input or Output of the DUT.

- [Tone Power](#)
- [Tone Gain](#)
- [Intermodulation Distortion](#)
- [Intercept Point Parameters](#)
- [Composite Triple Beat \(CTB\)](#)
- [Composite Second-order Beat \(CSO\)](#)
- [Cross-Modulation Distortion](#)

[Learn how to select these IMD parameters.](#)

## Tone Power Parameters

Tone Power parameters measure the **absolute** power level of the main tones, odd-order product tones up to the 9th order, and the 2nd order product tones. These tone powers can be measured at the input and output of the DUT. Because the tones come in pairs, the **Low tone**, **High tone**, and the **Average** of the two can be measured and displayed.

The Average Tone Power is calculated as follows:

$$\text{Avg} = (\text{High tone (dBm)} + \text{Low tone (dBm)}) / 2$$

When measuring the 2nd order products, only the Low tone and High tones are allowed. When the main tones are separated by less than 10 MHz, the Low tone (f2-f1) is below the frequency range of the PNA-X.

## Tone Gain

Tone Gain (in dB) calculates the main tone Output Tone power / Input Tone power. Because the tones come in pairs, Tone Gain can be calculated for the **Low tone**, **High tone**, and the **Average** of the two as indicated in the Average Tone Power calculation.

For IMDx for Converters, the Input and the Output tones are typically at different frequencies.

## Intermodulation Distortion Parameters

IMD parameters measure the **difference** in power level between the specified product tone and the main tones. These IMD parameters are calculated from the Tone Power measurements. IMD parameters can measure the odd-order product tones up to the 9th order, and the 2nd product tones, at the DUT Input or Output. For each specified product, the difference between the **Low** product and main tone, difference between the **High** product and main tone, and difference between the **Averages** of the product and main tone can be measured and displayed.

Swept IMD supports IMD parameters which are calculated as follows:

$$\begin{aligned} \text{IMxLo} &= \text{PwrXLo} - \text{PwrMainLo} \\ \text{IMxHi} &= \text{PwrXHi} - \text{PwrMainHi} \\ \text{IMx} &= \text{PwrX} - \text{PwrMain} \\ \text{IMxLoIn} &= \text{PwrXLoIn} - \text{PwrMainLoIn} \\ \text{IMxHiIn} &= \text{PwrXHiIn} - \text{PwrMainHiIn} \\ \text{IMxIn} &= \text{PwrXIn} - \text{PwrMainIn} \end{aligned}$$

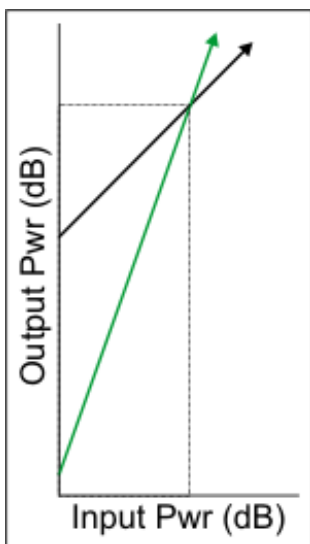
where:

- **x** = the IM product of interest (2, 3, 5, 7, 9)
- **Avg** is implied if **Hi** or **Lo** is not stated
- **Output** is implied when **In** is not stated

[Learn how to select IMD parameters.](#)

## Intercept Point Parameters

As the main tone output power increases (black arrow), output power in the specified product tone increases at a predictable, and steeper, rate (green arrow). At some point, the power in the product tone will be equal to the power in the main tone. The power level at which this occurs is known as the intercept point. Measuring this point directly is typically not possible. Therefore, it is calculated by measuring the main tone power and the specified product tone power.



The Swept IMD App can display either the DUT Input power or DUT Output power that is required to achieve the theoretical intercept point. This is called either Input Referred (IIP) or Output Referred (OIP).

This measurement can be made for the 2nd, 3rd, 5th, 7th, and 9th order intercept points. In addition, the measurements can be made for either the Low tone, the High tone, or the Average of the two. However, for the 2nd order intercept point, only Low and High tone parameters are supported; not Average.

Swept IMD supports Intercept Point parameters which are calculated as follows:

$$\begin{aligned} \text{OIPxHi} &= \text{PwrMain} - \text{IMxHi} / (x-1) \\ \text{OIPxHiIn} &= \text{PwrMain} - \text{IMxHiIn} / (x-1) \\ \text{OIPxLo} &= \text{PwrMain} - \text{IMxLo} / (x-1) \\ \text{OIPxLoIn} &= \text{PwrMain} - \text{IMxLoIn} / (x-1) \\ \text{OIPx} &= \text{PwrMain} - \text{IMx} / (x-1) \\ \text{OIPxIn} &= \text{PwrMain} - \text{IMxIn} / (x-1) \\ \text{IIPxHi} &= \text{PwrMainIn} - \text{IMxHi} / (x-1) \\ \text{IIPxHiIn} &= \text{PwrMainIn} - \text{IMxHiIn} / (x-1) \\ \text{IIPxLo} &= \text{PwrMainIn} - \text{IMxLo} / (x-1) \\ \text{IIPxLoIn} &= \text{PwrMainIn} - \text{IMxLoIn} / (x-1) \\ \text{IIPx} &= \text{PwrMainIn} - \text{IMx} / (x-1) \\ \text{IIPxIn} &= \text{PwrMainIn} - \text{IMxIn} / (x-1) \end{aligned}$$

where:

- **x** = the IM product of interest (2, 3, 5, 7, 9)
- **PwrMain** = average power of the main tones at the DUT Input or Output
- **IMx** = product tone power main tone power (from [above IMD parameter](#))
- **Avg** is implied if **Hi** or **Lo** is not stated
- **Output** is implied if **In** is not stated

[Learn how to select IMD parameters.](#)

### Composite Triple Beat (CTB)

From the NCTA Standard, composite triple beat is defined as the modulation beat of the target channel signal caused by triple beat resulting from the nonlinear characteristic of the DUT. Composite triple beat is expressed as the ratio of the target channel signal level to the maximum mean level of beat components dispersed around the carrier of that target channel.

Swept IMD supports two parameters of this type:

- CTB is based upon an approximation for the number of beats in mid-band
- CTBE is based upon an approximation for the number of beats at the band edge.

The equations for these two parameters are as follows:

$$\text{Mid-Band CTB (dB)} = -2 (P_i - P_s) + 6 + 10 \log(3N^2/8) + \text{CTB Offset}$$

$$\text{Band Edge CTBE (dB)} = -2 (P_i - P_s) + 6 + 10 \log(N^2/4) + \text{CTB Offset}$$

Where:

- **P<sub>i</sub>** = Output power level at the third order intercept point (dBm): OIP3 (Lo | Hi)
- **P<sub>s</sub>** = One of the following values based upon the Composite Normalization Mode:
  - For **PDBM** or **PDBMV** mode,  $P_s = \text{CompositeNormalizedCTBPower}$
  - For **Number of Carriers** mode,  $P_s = \text{PwrMain (AVG)} - 10 \log(N/2)$
  - For **None** mode,  $P_s = \text{PwrMain (AVG)}$
- **CTB Offset** = Offset value for CTB calculation
- **N** = Total number of carriers.

**Note:** **CTB Offset** and **N** values can ONLY be set using [SCPI or COM commands](#).

[Learn how to select IMD parameters.](#)

### Composite Second-Order (CSO)

From the NCTA Standard, composite second order is defined as the modulation beat of the target channel signal caused by second order beat resulting from the nonlinear characteristic of the DUT. Composite second order beat is expressed as the ratio of the target channel signal level to the maximum mean level of beat components dispersed around 0.75 MHz and 1.25 MHz above and below the carrier of that channel.

Swept IMD supports a CSO parameter which is calculated as follows:

$$\text{CSO (dB)} = (P_i - P_s) + 10 \log(N) + \text{CSO Offset}$$

Where:

- **P<sub>i</sub>** = Output power level at 2nd order intercept point: OIP2 (Lo | Hi)
- **P<sub>s</sub>** = One of the following values based upon the **Composite Normalization Mode**:
  - For **PdBm** or **PdBmV** mode,  $P_s = \text{CompositeNormalizedCSOPower}$
  - For **Number of Carriers** mode,  $P_s = \text{PwrMain (AVG)} - 10 \log(N/2)$
  - For **None** mode,  $P_s = \text{PwrMain (AVG)}$
- **CSO Offset** = Offset value for CSO calculation
- **N** = Number of distortion products.

**Note:** **CSO Offset** and **N** values can ONLY be set using [SCPI or COM commands](#).

[Learn how to select IMD parameters.](#)

## Cross-Modulation Distortion

From the NCTA Standard, cross modulation is defined as the distortion that causes modulated carrier components of undesired channels to amplitude-modulate the target channel carrier due to the nonlinear characteristic of the unit under test. Cross modulation distortion is expressed as the ratio of the target channel carrier level to the level of modulated components of the carrier of the target channel resulting from modulated signals of undesired channels.

Swept IMD supports an XMOD parameter which is calculated as follows:

$$XMOD = -2(P_i - P_s) + 6dB + 20\log(N)$$

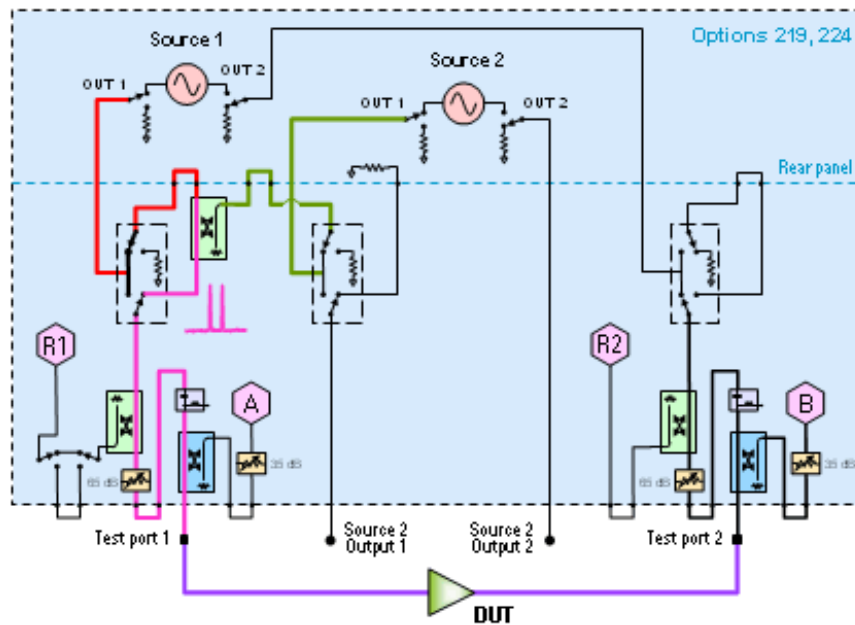
Where:

- $P_i$  = Output power level at third order intercept point: OIP3 (Lo | Hi)
- $P_s$  = Power level of each carrier: PwrMain (AVG)
- $N$  = Total number of carriers.

Make Cross Modulation settings using [SCPI or COM commands](#).

## How the IMD Application Works

The IMD App. requires a PNA-X with dual sources and the internal combiner. The following diagram illustrates how the PNA-X is configured to generate the two main tones.



**2-port PNA-X generates the f1 and f2 main tones.**

Depending on the specified parameters and sweep type, the sources and receivers are tuned to the appropriate frequencies in order to measure all of the required main and product tone powers. For example, an IM3 parameter requires the measurement of both main tones, and the 3rd order High and Low tone powers.

The [Narrowband IF path](#) is used for IMD measurements to help reduce spurious responses. Because the

narrowband filter has a bandwidth of about 28 kHz, using an IFBW greater than 30 kHz does nothing to improve measurement accuracy. [Learn how to set IFBW for IMD.](#)

### Limiting Stimulus Settings and Out of Range Product Tones

Because the main tones are generated by the PNA-X internal sources, the frequencies of the **main tones** must always be within the frequency range of the PNA. Sweep parameter values are adjusted when necessary to ensure that f1 and f2 frequencies are within these limits.

However, the PNA-X DOES allow you to make settings that cause the selected **IM products** to fall outside the frequency range of the PNA-X. For example, with the main tones at 10 MHz and 15 MHz, the PNA-X will allow you to select the parameter **IM3Lo** (3rd low side product tone). However, the frequency of this product will be at  $2f_1 - f_2$  or 5 MHz, which is below the frequency range of the PNA-X. In these cases, the trace data is set to zero, which converts to -200 dB in Log Mag format.

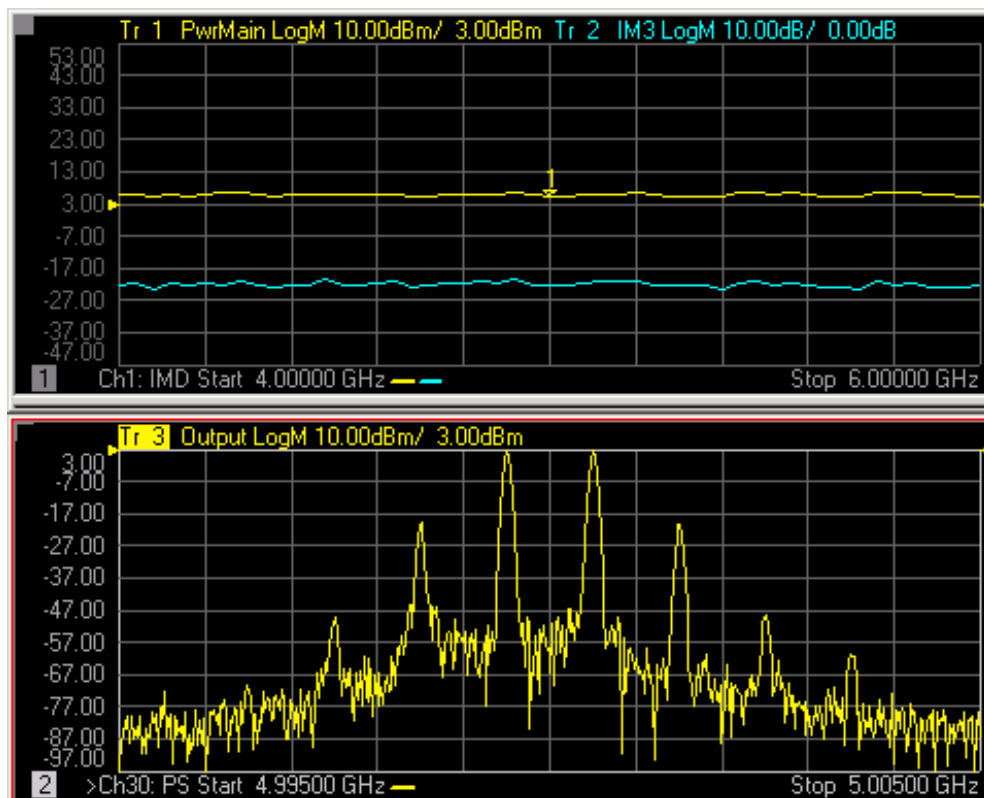
### Limited Number of Acquisitions

The total number of acquisitions per sweep can not exceed 32,001 points. The number of acquisitions is determined by multiplying the number of trace points, by the number of tones frequencies, then by 2 (for both Input and Output frequencies). The PNA-X will automatically reduce the number of trace points to ensure the total number of acquisition points does not exceed 32,001.

### How an IM Spectrum Channel Works

Before reading this topic, you should become familiar with [IMD Concepts](#).

The IM Spectrum channel provides a traditional spectrum analyzer view of the intermodulation distortion behavior of a device. Unlike the Swept IMD channel, the main tones (F1 and F2) are fixed while the receiver is swept over a frequency range of interest in order to generate a display as shown below.



**IM Spectrum trace (bottom) with Swept IMD traces (top)**

A typical spectrum analyzer does NOT have a signal source. This one does. The signal source, or stimulus, settings for the F1 and F2 main tones can be set in either the IM Spectrum channel or the Swept IMD channel. These settings include the frequencies and power levels of the main tones.

### Receiver Settings

The settings for the IM Spectrum receiver can be set ONLY in the IM Spectrum channel. These settings include how many tone products to view - which determines the center and frequency span - and the Resolution Bandwidth.

You can choose from several Resolution Bandwidths which also determines the number of data points used in the channel. The higher the Res BW, the fewer the number of data points. The formula for determining the number of points is:

$$N = (\text{Span}/\text{ResBW}) * 3$$

The IM Spectrum channel performs multiple measurements for each data point in order to reject unwanted images which are generated by the PNA-X internally. This provides a high degree of confidence that signals captured in a trace are real and are not spurious responses generated in the measurement process.

### IM Spectrum Parameters

You can select from three different IM Spectrum parameters.

1. The tones OUT of the DUT (default parameter).



2. The tones IN to the DUT (to be sure that the input signals are pure). **NOT supported** in IMx Spectrum
3. Reflected tones off the DUT input. **NOT supported** in IMx Spectrum

Learn how to select IM Spectrum parameters for [Amplifiers](#) or [Converters](#).

Learn all about IM Spectrum for [Amplifiers](#) or [Converters](#).

[See list of all IMD topics.](#)

---

Last Modified:

12-Mar-2009    Added Tone Gain (8.5)

9-Mar-2009    MX New topic

## Swept IMD for Amplifiers (Opt 087)

---

- [Features and Limitations](#)
- [Create a Swept IMD Measurement](#)
  - [Frequency tab](#)
  - [Power tab](#)
  - [How to specify IMD Parameters](#)
- [IMD Calibration](#)
- [Saving Swept IMD Power Data](#)

### Other IMD (Opt 087) Topics

## Features and Limitations

[See requirements.](#)

### Features

- Fast and easy setup for the measurement of a variety of distortion related parameters up to 9th order.
- Measurement at both input and output of a DUT.
- Supports a variety of sweep-modes for the main-tones: center-frequency linear and segment sweep, tone-separation sweep, power sweep, or CW sweep (fixed main-tone).
- Make very fast, accurate measurements using the PNA-X dual sources with high-power, high linearity, and low harmonics.
- Supports calibration and correction of Swept IMD parameters.
- Independently set IFBW for measuring main-tones versus product tones.

### Limitations

The following PNA-X features are NOT available with Swept IMD:

- 2-port non-frequency translating DUTs only
- [Number of points limited to 20,001](#)
- Analog Sweep ([Stepped sweep](#) mode only)

- [Log frequency](#) sweeps
- [Unratioed receiver measurements](#) (A, B, R)
- [ECal User Characterization](#)
- [Copy Channel](#)
- [Time Domain](#)
- [Balanced measurements](#)
- Save [Formatted Citifile](#) data.
- [Save SnP data](#).
- [External sources](#)
- [Interface Control](#)
- [Port extensions](#)
- [Some Fixturing Features](#)
- [External Test Set Control](#) (Option 551)
- [Integrated Narrowband](#) or [Narrowband Pulse App](#)
- [See Frequency limitations in a Swept IMD channel](#).

## Create a Swept IMD Measurement

1. On the PNA-X front panel, press **Meas** then **[Measurement Class]**
2. Select **Swept IMD**, then either:
  - **OK** delete the existing measurement, or
  - **New Channel** to create the measurement in a new channel.
3. A Swept IMD measurement is displayed. To select additional parameters to display, click **Response**, then **Measure**, then select a parameter from the list.

## How to start the Swept IMD Setup dialog

To provide quicker access, use the Setup softkey. [Learn how.](#)

### Using front-panel HARDKEY [softkey] buttons

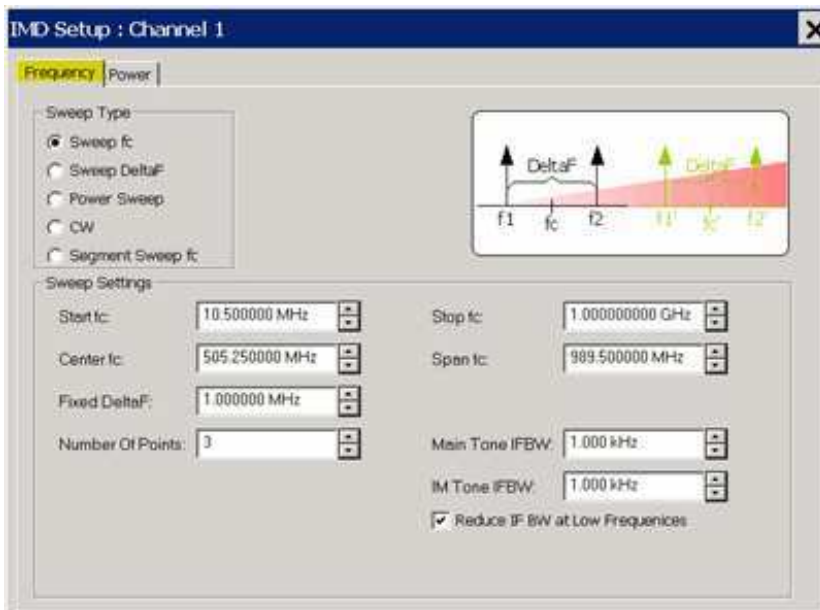
1. Press **FREQ**
2. then [**Swept IMD Setup**]

### Using a mouse with PNA-X Menus

1. Click **Stimulus**
2. then **Frequency**
3. then **Swept IMD Setup**

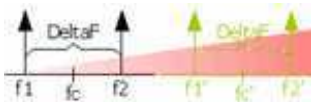
## Programming Commands

## Frequency tab - IMD Setup -dialog box help



Configures the Sweep Type and frequency range for IMD measurements.

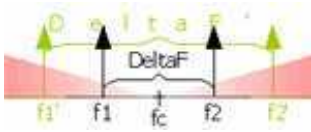
### Sweep Type and Sweep Settings



### Sweep $f_c$ (center frequency)

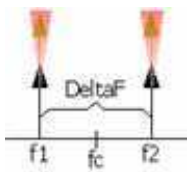
Maintaining a constant tone spacing (Fixed  $\Delta f$ ) and tone power,  $f_c$  is swept from Start  $f_c$  to Stop  $f_c$ . Center Frequency can also be specified as Center  $f_c$  and Span  $f_c$ .

At each  $f_c$ , the receivers are tuned to all of the required frequencies to measure the power of the appropriate tones.



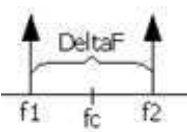
### Swept $\Delta f$ (tone spacing)

The specified  $f_c$  (center frequency) and tone power is held constant. The tone spacing is increased from Start  $\Delta f$  to Stop  $\Delta f$  in the specified number of points.



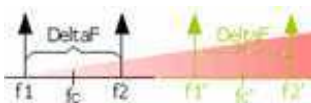
### Power Sweep

The main tone frequencies are specified as either  $f_1$  and  $f_2$ , or as  $f_c$  and  $\Delta f$ . These frequencies are held constant while the power of each main tone is varied from the Start-Power to Stop-Power in the specified number of power points. The power of each tone can be set (on the Power tab) individually or as a pair by checking Coupled Tone Power.



### CW

The main tone frequencies and power levels are held constant. Measurements are taken for the specified Number of Points. The X-axis is number of points.



### Segment Sweep $f_c$

Same as Sweep  $f_c$  except that the center frequencies are constructed using the standard segment table. [Learn how.](#)

### Segment Sweep Notes:

- The segment table shown on the dialog is **'READ-ONLY'**.
- Learn how to [Create and edit the Segment Sweep table](#).
- **Independent IFBW** and **Power** are NOT available.
- [X-axis point spacing](#) is available beginning with A.09.10.

**Number of Points** Enter the number of data points for each sweep. See Limited Number of Acquisitions.

### IFBW

The [Narrowband IF path](#) is used for IMD measurements to help reduce spurious responses. Because the narrowband filter has a bandwidth of about 28 kHz, using an IFBW greater than 30 kHz does nothing to improve measurement accuracy.

**Main Tone and IM Tone IFBW** IF Bandwidth is specified separately for the main tones ( $f_1$  and  $f_2$ ) and for

the intermodulation tones. This allows the higher-power main tones to be accurately measured at a higher - and faster - IFBW, while the lower-power product tones to be accurately measured a lower - and slower - IFBW.

**Reduce IF BW at Low Frequencies** - On the PNA-X, the trace noise becomes worse below 748 MHz. This is especially obvious between 10 MHz and 45 MHz. When this box is checked, the PNA-X uses a smaller IF Bandwidth than the selected value at frequencies below 748 MHz. [Learn more about the selected values.](#)

## Power tab - IMD Setup -dialog box help

The screenshot shows the 'Power tab - IMD Setup' dialog box. It has two main columns for DUT Input Port and DUT Output Port. The DUT Input Port column has checkboxes for 'Power On (All Channels)' and 'Coupled Tone Power'. Below these are dropdowns for 'Input Port' (Port 1) and 'Output Port' (Port 2). There are also sliders for 'Source Attenuator' (0 dB), 'Receiver Attenuator' (0 dB), 'Source Leveling Mode' (Internal), 'f1 Power' (-20.00 dBm), and 'f2 Power' (-28.00 dBm). The DUT Output Port column has a dropdown for 'Output Port' (Port 2) and sliders for 'Source Attenuator' (10 dB) and 'Receiver Attenuator' (0 dB). At the bottom, there is a 'Swept Tone Power Settings' section with 'Start', 'Stop', and 'Step' settings for f1 and f2 power, and a 'Path Configuration...' button.

Configures RF power and Power Sweep settings for IMD measurements.

**Power ON (All channels)** Check to turn RF Power ON or clear to turn power OFF for all channels.

**Coupled Tone Power** Check to set the same power level for each main tone using the f1 Power setting. Clear to set different f1 and f2 power levels.

### DUT Input Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Input Port** Choose Port 1 or Port 3 (with limitations).

**Source Attenuator** Specifies the port 1 attenuator. This attenuator affects the range of available power into the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** This attenuation setting protects the A receiver from damage.

**Source Leveling** Specifies the leveling mode. Choose Internal or Receiver R1. [Learn how to configure Receiver Leveling.](#) Open Loop should only be used when doing [Wideband Pulse measurements.](#)

**f1 Power** Specify the power level for f2. Choose a value between -30 dBm and +30 dBm. When Coupled Tone Power is checked, power is set for both f1 and f2 tones.

**f2 Power** Specify the power level for f2. Choose a value between -30 dBm and +30 dBm.

### DUT Output Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Output Port** Choose Port 2 or Port 4 (with limitations).

**Source Attenuator** This setting is used to improve the load match at the DUT output. Select 0 dB for power levels up to 10 dBm, and increase by 10 dB for every 10 dBm more output power.

**Receiver Attenuator** Specifies the attenuator setting for port 2. When the power into the receiver test port is around +10 dBm, the PNA-X receiver may be in compression. However, with receiver attenuation, lower input power levels may become too noisy to make accurate power measurements. In this case, lower IFBW for the IM tones to reduce noise. [Learn more about Receiver Attenuation.](#)

### Swept Tone Power Settings

**f1 and f2 Start and Stop Power** Available ONLY in Power Sweep. Sets the Start and Stop power levels for f1 and f2, either individually or together with Coupled Tone Power checked.

**Path Configuration** click to launch the [RF Path Configuration](#) dialog.

### How to specify IMD Parameters

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then **[More Meas...]**

#### Using a mouse with PNA-X Menus

1. Click **Response**
2. then **Measure**
3. then **More Meas...**

◀ **Programming Commands** ▶

IMD New Trace dialog box help

Param. Name	Type	Tone Select	Order	Measure At
<input type="checkbox"/> PwrMain	Tone Power	Avg	1	DUT OUT
<input type="checkbox"/> IM3	IMD Relative to Carrier	Avg	3	DUT OUT
<input type="checkbox"/> OIP3	Output Referred Intcpt. Pt.	Avg	3	DUT OUT
<input type="checkbox"/> IIP3	Input Referred Intcpt. Pt.	Avg	3	DUT OUT
<input type="checkbox"/> Pwr2	Tone Power	Avg	2	DUT OUT

Channel Number
 ☐ Create In New Window
 ☒ Auto-Creat Windows

Use this dialog to select IMD parameters to measure and display. Five parameters can be selected, then click **Apply** to create those traces, then select more without closing the dialog. There is no limit to the [number of traces and windows](#) allowed in the PNA.

**Note:** Calculations are NOT performed to determine if the frequency of a selected intermod (Order) product will be within the frequency range of the PNA-X. Measurements that fall outside of the frequency range of the PNA-X are displayed as -200 dB.

**Param Name** This name is built from the selected **Type** and **Order**. Once built, check to measure this parameter.

### Type

Select the type of Swept IMD measurement to make. The characters in (parenthesis) are used in the Param Name. Choose from the following:

- **Tone Power (Pwr)** Measures the absolute power of the specified (ordered) tone. [Learn more.](#)
- **IMD Relative to Carrier (IM)** Measures the power of the specified (ordered) tone relative to the power of the f1 or f2 tone. [Learn more.](#)
- **Input Referred Intercept Point (IIP)** From DUT measurements, calculates the theoretical power level at which the specified (ordered) intermod product will be the same power level as the carrier at the output of the DUT. The input power to the DUT at which this output power occurs is reported and displayed. [Learn more.](#)
- **Output Referred Intercept Point (OIP)** From DUT measurements, calculates the theoretical power level at which the specified (ordered) product will be the same power level as the main tone at the output of the DUT. This value is reported and displayed. [Learn more.](#)
- **CTB Band-Edge Distortion (CTBE)** Composite 'Triple Beat' Distortion - 3rd order DUT OUT only. [Learn more.](#)



- **CTB Mid-Band Distortion (CTB)** Composite 'Triple Beat' Distortion - 3rd order DUT OUT only. [Learn more.](#)
- **CSO Distortion (CSO)** 2nd order DUT OUT only. [Learn more.](#)
- **XMOD 3rd Order Crossmod.** 3rd order DUT OUT only. [Learn more.](#)
- **Tone Gain (ToneGain)** From tone power measurements, calculates the Output Tone power / Input Tone power for the specified tones. [Learn more.](#) The Input and Output tones are different frequencies.

### Tone Select

Select the tone and intermodulation product to be measured ([See image](#)).

**Hi** and **Lo** are appended to the Param Name when selected.

Nothing is appended to the Param Name when Avg is selected.

Choose from the following:

- **High** Measure the specified (ordered) tone on the High side of the main tones.
- **Low** Measure the specified (ordered) tone on the Low side of the main tones.
- **Avg** Measure both the High and Low tones and calculate the dB average power level.

### Important Notes

- If either the High or Low side falls outside the frequency range of the PNA, then **Avg** is NOT allowed, and the measurement shows as zeroes ( **-200 dB** in Log Mag format).
- When displaying 2nd order traces, **Avg** is NOT allowed, even when both the Lo and Hi products are displaying valid data. This is because 2 Low and 2 High products are usually very different from one another.

### Order

Specify the intermodulation product to measure. Choose from 1, 2, 3, 5, 7, 9. 2nd order products are more likely to be outside of the PNA-X frequency range than odd order products.

**Main** is appended to the Param Name when 1 is selected.

Otherwise, the tone number is appended to the parameter name.

### Measure At

Measure the selected parameter at either the DUT **Input** or **Output**.

- **In** is appended to the Param Name when Input is selected.
- Nothing is appended to the Param Name when Output is selected.

**Select All** Checks all available parameters.

**Clear All** Clears all checked parameters.

### Channel / Window Selections

**Channel Number** Select the channel for the new traces.

#### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the [PNA-X traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See [PNA-X number of windows limitation](#).

**Apply** Creates the measurements. The dialog remains open to allow you to create more measurements.

## IMD Calibration

### Overview

1. At the first page of the IMD Cal Wizard, you tell the calibration routine the frequencies at which the calibration is to be performed. Optionally, you can choose to perform the source power cal at only the center frequency midway between the main tones. For IMDx ONLY, you can also choose to perform a source power cal of the LO source.
2. The PNA-X calculates an array of source and receiver frequencies that incorporate all the main tone frequencies (low and high) and all the product tone frequencies.
3. Using a power meter at port 1, a source and receiver calibration is performed to calibrate the R1 reference receiver to be a fast and 'tunable' power meter.
4. The R1 reference receiver is then used to perform a source power cal of the main tone frequencies: first the Source 1 / Low tone then the Source 2 / High tone. Both sources are left ON while each tone is measured in order to duplicate the impedance match under which the measurement will be performed.
5. Then a standard 2-port SOLT cal is performed at all frequencies using either an ECal module or mechanical standards. The 2-port cal is used to correct the source calibration R1 tracking terms for the match of the power sensor. It is also used to transfer the R1 tracking term to the B receiver.

### Notes

- If the main tone frequencies change but are within the frequency range in which the calibration was performed, then the calibration becomes interpolated C\*. This can occur by changing the start/stop/center/span frequencies, the number of points, or the sweep type. Learn more about [Interpolation](#).
- [Receiver calibrations](#) that are performed in a standard channel can be applied to a Swept IMD channel. However, [Source Calibrations](#) can NOT be applied.

## See Also

[IM Spectrum Calibration](#)

### How to start a Swept IMD Calibration

#### Using front-panel HARDKEY [softkey] buttons

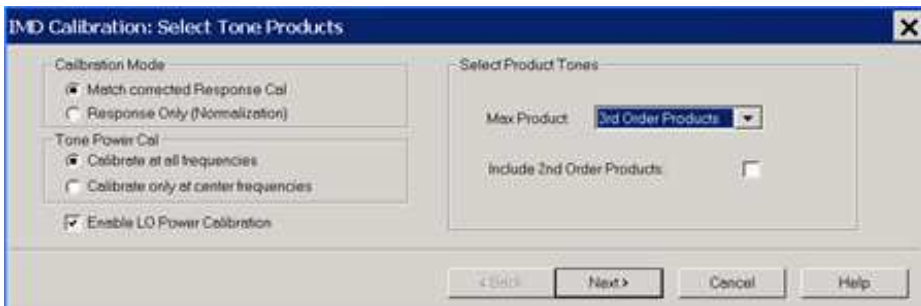
1. Press **CAL**
2. then **[Start Cal]**
3. then **[Cal Wizard]**

#### Using a mouse with PNA-X Menus

1. Click **Response**
2. then **Cal Wizard**

◀ **Programming Commands** ▶

### Select Tone Products - IMD Cal dialog box help



#### Calibration Mode

**Match corrected Response Cal** This selection performs a full IMD calibration as described in the above [Calibration 'Overview'](#).

**Response ONLY (Normalization)** This IMD Cal does NOT correct for the mismatch of the power sensor. Choose this if you have a test configuration that does not easily accommodate making match measurements. Instead of a standard 2 port SOLT cal ([Step 5 above](#)), only the transmission tracking term is measured and used to transfer the R1 receiver tracking term (produced by the power sweep) to the B receiver.

**Note:** For the Response (Normalization) Cal, it is assumed that a [zero-length THRU standard](#) is being used to connect port 1 to port 2. If an adapter is used, there is NO compensation for delay or loss of the adapter. This can NOT be changed.

#### Tone Power Cal

**Calibrate only at center frequencies** The source power cal portion is performed at only the center frequency, which is midway between the main tones. This cuts the source power calibration time (the slowest part of the calibration) in half. The measurement is interpolated although the C\* annotation is not shown in

the status bar.

**Calibrate at all frequencies** The source power cal portion is performed at all main tone frequencies.

**Enable LO Power Calibration** (IMDx ONLY) Check to perform a standard power calibration of the LO source as part of the calibration process.

### Select Product Tones

**Max Product** Select the highest product that you will be measuring. The low and high frequencies for that product, and all lower 'odd' order products will be calibrated. For example, when 5th Order Product is selected, the frequencies for the Main Tones, and the Low and High order products for the 3rd and 5th order products will be included in the calibration.

**Include 2nd Order Products** Check to calibrate the 2nd order products in the frequencies to be calibrated. The frequencies of these products are usually far from the main tones.

### Select DUT Connectors and Cal Kits - IMD Cal dialog box help

If **Response Cal Only** is selected on the previous page, click **View/Modify** to change the Source Cal settings, or click **Next>** to continue.

Otherwise, this is a [standard PNA-X Cal Wizard](#) page except for the following:

**Power Sensor** When the power sensor connector is not the same type as the DUT Port 1 connector, then for optimum accuracy, an extra cal step is required to measure the adapter that is used to connect the power sensor to the port 1 reference plane during the Source Power Cal. Specify the connector type and the Cal Kit that will be used for that step. Select **Ignored** to NOT compensate for the adapter.

**Modify Cal** Check, then click **Next**, to Modify Cal (Standards AND Thru Method).

**View/Modify Source Cal Settings** Click to launch the [Source Cal Settings](#) (for Apps) dialog.

[Learn more about IMD Calibration](#)

## IMD Calibration Step 1 dialog box help



**Power Level** at which to perform the Port 1 Source Power Cal.

It is usually best to perform the Source Power Cal at 0 dBm because the power sensor is calibrated at that level.

However, if a component is used between the PNA-X source and the calibration reference plane, then adjust the power level so that the power at the sensor is about 0 dBm if possible.

[Learn more about IMD Calibration](#)

The remaining dialog pages are the same as [the standard Cal Wizard](#).

## Saving Swept IMD Power Data

Swept IMD power data, Log Mag ONLY, can be saved to a \*.csv file. This data type can be read by spreadsheet programs, such as Microsoft Excel. Data from the last **complete** sweep is saved to the specified \*.csv file.

### How to save Swept IMD (and IMDx) data

With a Swept IMD or IMDx channel active...

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SAVE**
2. then **[Save Data As]**
3. then, **File Type = IMD Sweep Data (\*.csv)**

#### Using a mouse with PNA-X Menus

1. Click **File**
2. then **Save Data As**
3. then **File Type= IMD Sweep Data (\*.csv)**

◀ **Programming Commands** ▶

### Notes:

- For every tone, six power levels are saved in this order: OUT Avg | OUT Lo | OUT Hi | IN Avg | IN Lo | IN Hi
- Power levels for the Main tones are always saved, regardless of the active measurement.
- All tones that are displayed are also saved. For example, any displayed 3rd order tone parameter causes the

3rd tone power levels to be saved.

- Only tone powers are saved. Calculated parameters, such as **IMD Relative to Carrier (IM)** are NOT saved.
- If calibration is turned **ON** when the file is saved, then all data is calibrated. Otherwise, raw data is saved.

Agilent Technologies, N5242A, [REDACTED]						
Date: Tuesday, February, 24, 2009 19:53:42						
Calibration	OFF					
Num Freq F	201					
Num Tones	6					
Frequency	PwrMain	PwrMainLo	PwrMainHi	PwrMainIn	PwrMainLoIn	PwrMainHiIn
<Hz>	<LogMag	<LogMag(d	<LogMag(d	<LogMag(d	<LogMag(dBr	<LogMag(dBr
1.05E+07	-76.721	-74.1983	-79.2441	-66.6679	-21.2689	-112.067

This image shows the 6 Main tone power levels that are always saved.

In the power parameter labels, **Output** and **Avg** are implied as in the [parameter selection](#). For example:  
**PwrMain** = Average Output power of the main tones.

---

#### Last Modified:

- |             |   |
|-------------|---|
| 12-Apr-2010 | Removed some fixturing limitations        |
| 19-Nov-2009 | Added X-axis point spacing (9.1)          |
| 11-Aug-2009 | Added limited port mapping (A.09.00)      |
| 13-Jul-2009 | Added Tone select notes                   |
| 27-Feb-2009 | Added data save and modified cal for IMDx |
| 14-Aug-2008 | New topic                                 |

## IM Spectrum for Amplifiers (Opt 087)

---

Intermodulation Spectrum measurements can be made independently, or coupled with Swept IMD measurements. They are a distinct measurement class and therefore always made in a separate channel.

- [Requirements and Limitations](#)
- [Create an IM Spectrum Channel](#)
- [Select IM Spectrum Parameters](#)
- [Start the IM Spectrum Setup dialog](#)
  - [Frequency tab](#)
  - [Power tab](#)
  - [Tracking tab](#)
- [Calibration](#)

### See Also

- [IM Spectrum Concepts](#)
- [Swept IMD Measurements](#)
- [Programming commands](#)

---

### Requirements and Limitations with IM Spectrum

The IM Spectrum channel requires a PNA-X with dual sources and the internal combiner. This is the same configuration that is required for all IMD App channels. [See requirements](#)

The following PNA-X features are NOT available in an IM Spectrum channel:

- [Unratioed receiver measurements](#) (A, B, R)
- [ECal User Characterization](#)
- [FOM](#) or [FCA](#)
- [Copy Channel](#)
- [Time Domain](#)
- [Balanced measurements](#)
- Save [Formatted Citifile](#) data.

- [Save SnP data.](#)
- [External sources](#)
- [External Test Set Control](#) (Option 551)
- [Interface Control](#)
- [Port Extensions](#)
- [Integrated Narrowband](#) or [Narrowband Pulse App](#)

## Create an IM Spectrum Channel

An IM Spectrum channel can be created independently from a Swept IMD channel or coupled with the stimulus settings of an existing Swept IMD measurement.

### To create an independent IM Spectrum channel

1. On the PNA-X front panel, press **Meas** then **[Measurement Class]**
2. Select **IM Spectrum**, then either:
  - **OK** delete the existing measurement, or
  - **New Channel** to create the measurement in a new channel.

### To create an IM Spectrum channel from an existing Swept IMD channel

1. With a Swept IMD channel active, press the **MARKER** key on the front panel.
2. Move the marker to the data point of interest. It can be moved later.
3. Press **[Marker Functions]** then **[Marker -> IM Spectrum]**

This creates or configures an existing IM Spectrum channel based upon the configuration of the Swept IMD channel.

## Select IM Spectrum Parameters



## How to create a NEW (additional) IM Spectrum trace

With the IM Spectrum channel active and a **Tr1 Output** trace displayed:

### Using front-panel HARDKEY [softkey] buttons

- Press **TRACE 2**  
A new Tr2 Input trace appears.

### Using a mouse with PNA-X Menus

- Click **Trace/Chan**
  - then **New Trace**The [New Trace](#) dialog appears.

## To change the active (existing) IM Spectrum parameter

Select the trace to be replaced. Then:

- Press **MEAS**
- then **[select the parameter]**
- Click **Response**
  - then **Measure**
  - then **select the parameter**

◀ **Programming Commands** ▶

## New Trace - IM Spectrum -dialog box help



Check the IM Spectrum measurement to add to the display.

- **Output** View signals OUT of the DUT and into PNA-X port 2 (B receiver).
- **Input** View signals IN to the DUT (R1 receiver). Use this when measuring IM product frequencies to determine the power level of spurious signals into the DUT at those frequencies.
- **Reflection** View signals reflected off the DUT input and back into PNA-X port 1 (A receiver)

**Select All** Checks all available parameters.

**Clear All** Clears all checked parameters.

### Channel / Window Selections

**Channel Number** Select the channel for the new traces.

#### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the [PNA-X traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See [PNA-X number of windows limitation](#).

## IM Spectrum Setup Dialog

### How to start the IM Spectrum Setup dialog

To provide quicker access, use the Setup softkey. [Learn how](#).

With an IM Spectrum measurement active:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[IM Spectrum Setup]**

#### Using a mouse with PNA-X Menus

1. Click **Stimulus**
2. then **Frequency**
3. then **IM Spectrum Setup**

**Programming Commands**

**Frequency tab - IM Spectrum -dialog box help**

The screenshot shows a configuration window for IM Spectrum measurements. It is divided into three main sections:

- Sweep Type:** Includes radio buttons for Linear, 2nd Order Spectrum, 3rd Order Spectrum, and Nth Order Spectrum (which is selected). A dropdown for 'Order N' is set to 9.
- Resolution BW:** A text field set to 600.000 kHz.
- Stimulus Settings:** Includes fields for 'fc (Tone Center)' (1.000000000 GHz), 'Delta F' (10.000000 MHz), 'f1' (995.000000 MHz), and 'f2' (1.005000000 GHz).
- Response Settings:** Includes fields for 'Start Spectrum' (990.000000 MHz), 'Stop Spectrum' (1.050000000 GHz), 'Center Spectrum' (1.000000000 GHz), and 'Span Spectrum' (100.000000 MHz).

**Note:** The number of data points in an IM Spectrum channel =  $3 * \text{SPAN} / \text{ResBW}$ . This can NOT be changed directly.

Configures the Sweep Type and frequency range for IM Spectrum measurements.

### Sweep Type

Provides several methods to tune the IM Spectrum **receivers** (NOT stimulus) to view the power spectrum of arbitrary frequency ranges or various distortion products.

**Linear** Allows the start/stop or center/span receiver frequencies to be set arbitrarily for f1 and f2. Enter the Response Settings below.

**2nd Order** Couples the receiver frequency range to f1 and f2 to provide a convenient means of observing the spectrum surrounding ONLY the high-side 2nd order harmonic where Center =  $(f1 + f2)$ , Span =  $3 * (f2 - f1)$ . **Note:** The center frequency is NOT set on the main tones.

**3rd Order** Couples receiver frequency range to f1 and f2 to provide a convenient means of observing the spectrum surround the main-tones including the 3rd Order products where Center =  $(f1 + f2) / 2$ , Span =  $4 * (f2 - f1)$ .

**Nth Order** Couples receiver frequency range to f1 and f2, providing a convenient means of observing the spectrum surrounding the main-tones for an arbitrary span where Center =  $(f1 + f2) / 2$ , and Span =  $N * (f2 - f1)$ . This allows you to set the span arbitrarily, but have the center frequency track the main tone frequencies.

### Resolution BW

The IM Spectrum channel utilizes a set of Gaussian filters instead of the standard PNA-X IF filters in order to provide similar behavior to traditional spectrum analyzers.

The comprehensive list of filters that are available for IM spectrum are: 3 MHz, 1 MHz, 600 kHz, 300 kHz, 150 kHz, 100 kHz, 60 kHz, 10 kHz, 3 kHz, and 1 kHz.

**Note:** The 10 kHz Res BW filter can generate image signals which are not a product of the DUT. You can verify the integrity of a questionable signal by switching to the 3 kHz or 60 kHz filter and looking for the image signal in the same location.

Not all of these filters are available for all measurements. Narrower filters are available for use with narrower frequency spans, and wider filters are available for wider spans.

### Stimulus Settings

Allows configuration of the main-tone frequencies. Available ONLY when Tracking is OFF.

- **fc (main-tone center frequency)** =  $(f1 + f2) / 2$

- **DeltaF (main-tone frequency separation)** =  $(f2 - f1)$ .
- **f1** = Low-side main-tone frequency
- **f2** = High-side main-tone frequency

### Response Settings

Allows configuration of the frequency range to sweep the receivers in terms of start/stop or center/span. Available for Linear sweep types ONLY.

- **Start Spectrum** = first frequency point of the power spectrum sweep
- **Stop Spectrum** = last frequency point of the power spectrum sweep
- **Center Spectrum** =  $(\text{Start Spectrum} + \text{Stop Spectrum}) / 2$
- **Span Spectrum** =  $(\text{Stop Spectrum} - \text{Start Spectrum})$

### Power tab - IM Spectrum -dialog box help

Swept Tone Power Settings			
	Start	Stop	Step
f1 Power:	-20.00 dBm	-10.00 dBm	0.050 dB
f2 Power:	-20.00 dBm	-10.00 dBm	0.050 dB

Configures RF power settings for IM Spectrum measurements.

**Power ON (All channels)** Check to turn RF Power ON or clear to turn power OFF for all channels.

**Coupled Tone Power** Check to set the same power for each main tone using the f1 Power setting. Clear to set f1 and f2 power levels individually.

### DUT Input Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Input Port** Choose Port 1 or Port 3 (with limitations).

**Source Attenuator** Specifies the attenuator setting associated with the port connected to the input of the DUT. This attenuator will affect the range of available power into the DUT [Learn more about Source Attenuation.](#)

All PNA-X channels in continuous sweep must have the same attenuation value. [Learn more.](#)

**Receiver Attenuator** This attenuation setting protects the A receiver from damage.

**Source Leveling** Specifies the leveling mode. Choose Internal or Receiver R1. [Learn how to configure Receiver Leveling](#). Open Loop should only be used when doing [Wideband Pulse measurements](#).

**f1 Power** Power level for f1 and f2 when Coupled Tone Power is checked.

**f2 Power** Power level for f2. Available only when Coupled Tone Power is NOT checked.

### DUT Output Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more](#).

**Output Port** Choose Port 2 or Port 4 (with limitations).

**Source Attenuator** This setting is used to improve the load match at the DUT output. Select 0 dB for power levels up to +10 dBm, and increase by 10 dB for every 10 dBm more output power.

**Receiver Attenuator** Specifies the attenuator setting for the receiver associated with the DUT output port. When the power into the receiver test port is around +10 dBm, the PNA-X receiver may be in compression. However, with receiver attenuation, lower input power levels may become too noisy to make accurate power measurements. [Learn more about Receiver Attenuation](#).

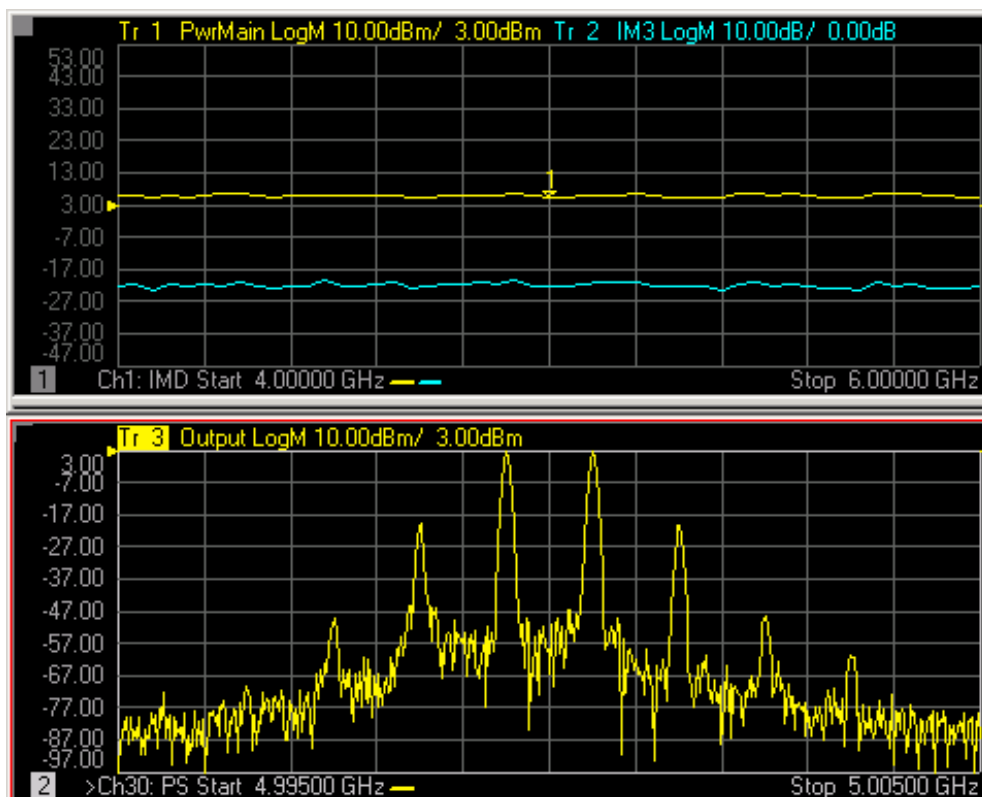
**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

### Tracking tab - IM Spectrum -dialog box help



Allows the IM Spectrum channel to use (track) the **stimulus** (Main Tone) settings of an existing Swept IMD channel. This selection is NOT available when there is no Swept IMD channel.

**Tracking Enable** Check to use the frequency and power stimulus settings from the specified IMD channel. When enabled, stimulus settings on the Frequency Tab are disabled and ALL stimulus settings, such as frequencies, power and attenuator settings, and calibration, are copied from the Swept IMD channel to the IM Spectrum channel.



*IM Spectrum trace (bottom) tracking with Swept IMD traces (top)*

- In the **top** Swept IMD window the main tones are swept from 4 GHz to 6 GHz with some specified delta F tone separation. The marker is on the center data point at **5 GHz**.
- The **bottom** IM Spectrum window center frequency is the same as the above marker: **5 GHz**, but it has a much narrower frequency range of +/- 500 kHz. The IM Spectrum channel sets the receiver to see the two main tones, plus the third, fifth, and seventh-order products.

## Step Mode

When tracking is enabled, set the method by which the IM spectrum measurement tracks the IMD channel.

**Manual Step** When selected, IM Spectrum measurements occur at only the specified IMD channel data point. **Stimulus Point** specifies the data point, by number, in the Swept IMD channel which has the stimulus settings to use for the IM Spectrum sweep.

**Automatic Step** When selected, causes the IM Spectrum channel to sequentially setup each of the stimulus conditions through which the Swept IMD channel sweeps the DUT. Each sweep of IM Spectrum is performed using the next set of stimulus conditions.

For example, in the above image, the first Swept IMD data point is at 4 GHz. The first IM Spectrum sweep uses a center frequency of 4 GHz. The following IM Spectrum sweep would be at the second Swept IMD data point or 4.01 GHz, and so forth. After the last data point in the sweep is reached, the IM Spectrum channel begins again at the first Swept IMD data point. The only indication that Tracking is enabled is in Automatic mode, you can see the center frequency increment with each IM Spectrum sweep.

---

## Calibration

A calibration of the IM Spectrum channel is NOT performed using a calibration wizard.

An IM Spectrum channel is calibrated from a Cal Set that is used on a Swept IMD channel. The Cal Set can be applied to the IM Spectrum channel using the Manage Cal Sets dialog ([Learn how](#)) or from the Marker =>IM Spectrum softkey ([Learn how](#)).

However, a Swept IMD channel with [Sweep Type = Power Sweep](#) can NOT be applied to a IM Spectrum channel. This is because a Cal Set for power sweep contains only a CW frequency and the IM Spectrum channel requires a swept frequency range. Zero Span is not supported in an IM Spectrum channel.

A calibrated IM Spectrum trace corrects the source and receiver power level accuracy of the displayed Tones.

[See Swept IMD Calibration](#)

---

Last Modified:

16-Sep-2009	Fixed calibration
11-Aug-2009	Added limited port mapping
18-Aug-2008	MX New topic

## Swept IMD for Converters (IMDx)

---

- [Limitations](#)
- [How to Configure your Hardware](#)
- [Create a Swept IMDx Measurement](#)
- [Valid Mixer Configuration / Sweep Type Combinations](#)
  - [Tone Frequency tab](#)
  - [Tone Power tab](#)
  - [Mixer Frequency tab](#)
  - [Mixer Power tab](#)
  - [Mixer Setup tab](#)
- [How to specify IMDx Parameters](#)
- [IMDx Calibration](#)
- [Saving IMDx Data](#) (Swept IMD topic)

### Other IMD Topics

## Limitations

- 1-Stage Converters ONLY

The following PNA-X features are **NOT** available with Swept IMDx:

- Analog Sweep ([Stepped sweep](#) mode only)
- [Log frequency](#) sweeps
- [Unratioed receiver measurements](#) (A, B, R)
- [ECal User Characterization](#)
- [Copy Channel](#)
- [Time Domain](#)
- [Balanced measurements](#)



- Save [Formatted Citifile](#) data.
- [Save SnP data](#).
- [Interface Control](#)
- [Port extensions](#)
- [Some Fixturing Features](#)
- [External Test Set Control](#) (Option 551)
- [Integrated Narrowband](#) or [Narrowband Pulse App](#)
- [See Frequency Limitations](#)

**Note:** Beginning with A.09.00, **Embedded LO** measurements are allowed in IMDx and IMSpectrum. Configure the measurement as you would with SMC. [Learn how](#).

## How to Configure your Hardware

The PNA-X is extremely versatile, and can be configured in many ways to make IMDx measurements. While not all conceivable configurations are documented here, a few of the most common examples are provided to show the basic concepts.

### DUT Configuration

- The DUT Input must be connected to PNA-X Port 1 which supplies the f1 and f2 tones.
- The DUT Output must be connected to PNA-X Port 2 which uses the PNA-X B receiver.
- See LO Source configuration below.

### Source Configuration

Three sources are required to make IMDx measurements: Two sources are PNA-X internal; the third is an external source.

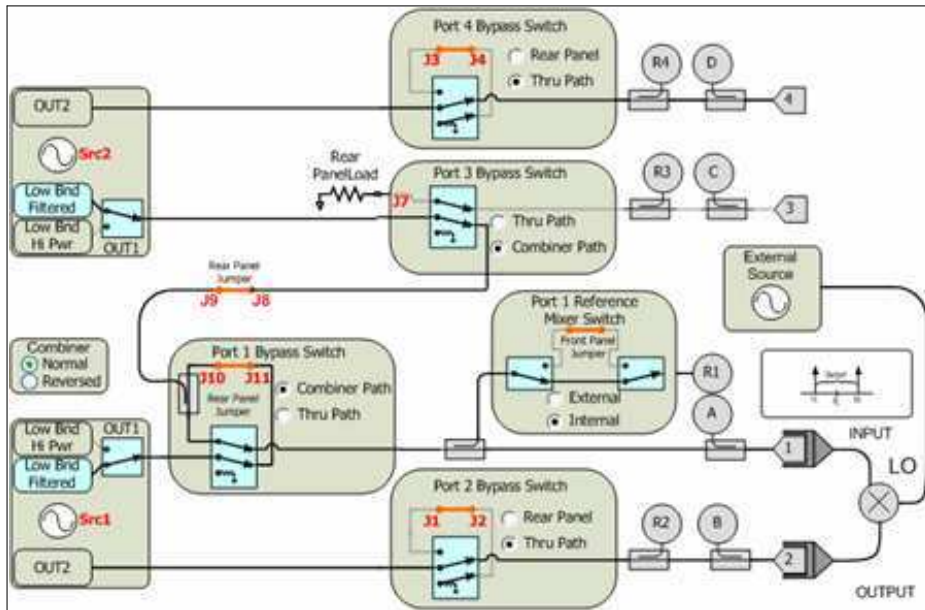
- **F1 tone** - Must come out PNA-X Port 1 Default is Src 1
- **F2 tone** - Must come out PNA-X Port 1 through the internal combiner. This source can come from an internal or external source. Default is the internal Src 2.
- **LO** - Can come from internal or external source. Default is Not controlled, set to 0 Hz.
  - If using the internal Src 2 source, the f2 tone must come from an external source through the rear-panel.
  - If using an external source, it can be connected directly to the DUT, or through the PNA-X Port 3 or Port 4 using the Path Configurator.

## Configuration Examples

Three configurations are shown below to illustrate how to use the [Path Configurator](#) to manually make switch settings.

IMDx and IMx Spectrum channels are ALWAYS configured using the default configuration switch settings shown in the image below. This is NOT the same default configuration that is used for an S-parameter channel. A manual switch setting is required whenever a custom configuration is used.

The manual switch setting, which must be done every time an IMDx or IMx Spectrum channel is created, can be saved using the **Store** button in the Path Configurator. Then save the entire IMDx setup as an [Instrument State](#). This will load the custom Path Configuration when the Instrument State is recalled.



### 1. Default Configuration No manual switching required.

- **f1** - Internal Src1
- **f2** - Internal Src2 through combiner in Normal
- **LO** - Connect external source directly to the DUT LO.

### 2. External LO through Port 3 or Port 4 (4-port PNA-X only)

Use this configuration to monitor LO power using R3 or R4. (Future parameters)

- **f1** - Internal Src1
- **f2** - Internal Src2 through combiner in Normal
- **LO** - Connect external source through the rear-panel ( J7 for Port 3; J3 for Port 4 ).
  - Connect the DUT LO to PNA-X Port 3 or Port 4.
  - For Port 3, NO switching is required.

- For Port 4, switch Port 4 Bypass Switch to **Rear Panel**.

### 3. Internal LO through Port 3 or Port 4; External f2 through rear-panel (4-port PNA-X only)

This configuration is commonly used for FCA measurements where the internal second source is used as the LO.

- **f1** - Internal Src1
- **f2** - External f2 through rear-panel J9. No switching required.
- **LO** - Internal Src2 through:
  - Port 3 Switch Port 3 Bypass Switch to **Thru Path**.. Connect the DUT LO to PNA-X Port 3
  - Port 4 Switch Port 4 Bypass Switch to **Thru Path**. Connect the DUT LO to PNA-X Port 4

## Create a Swept IMD Measurement

1. On the PNA-X front panel, press **Meas** then **[Measurement Class]**
2. Select **Swept IMD**, then either:
  - **OK** delete the existing measurement, or
  - **New Channel** to create the measurement in a new channel.
3. A Swept IMD measurement is displayed. To select additional parameters to display, click **Response**, then **Measure**, then select a parameter from the list.

### How to start the Swept IMD Setup dialog

To provide quicker access, use the Setup softkey. [Learn how.](#)

#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[Swept IMDx Setup]**

#### Using a mouse with PNA-X Menus

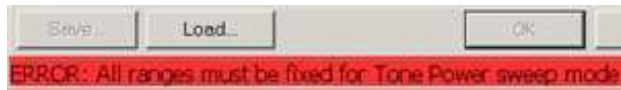
1. Click **Stimulus**
2. then **Frequency**
3. then **Swept IMDx Setup**

**Programming Commands**

## Valid Mixer Configuration / Sweep Type Combinations

Configuring the IMDx Setup dialog can be challenging at first. **RED** messages like this one appear at the bottom of

the IMDx Setup dialog to notify you of an invalid setup.



The following information can provide you with a 'mental model' of the VALID mixer configuration / sweep type combinations which can make it easier to setup your measurement.

The following are the **Valid Mixer Configurations** in table format:

Sweep Type	Input	LO	Output
<b>Swept Input (Fixed Tones)</b>			
Sweep fc	<b>Swept</b>	Swept	Fixed
	<b>Swept</b>	Fixed	Swept
<b>Fixed Input</b>			
CW	<b>Fixed</b>	Fixed	Fixed
Sweep Delta F			
(Tone) Power Sweep			
LO Power Sweep	<b>Fixed</b>	Swept	Swept
CW			

The following is an explanation of the table:

- **SWEPT Input** (There is ONLY one: **Sweep fc**) - Sweeps the center frequency of the tones, but the tone spacing remains fixed.
  - Either the LO or Output MUST also be swept.
  - On the Mixer Frequency tab, select Start/Stop, or Center/Span for each range to be swept.
- **FIXED Input** - The center frequency of the tones is fixed. On the Mixer Frequency tab - Input range, select Fixed for the following sweep types:
  - **Sweep Delta F** Tone separation changes. The LO and Output frequencies are always fixed.
  - **Power Sweep** - Tone power changes. The LO and Output frequencies are always fixed.
  - **LO Power Sweep** - All frequencies are fixed as the LO power is swept.
  - **CW** - Tones do NOT change. The LO or Output frequencies CAN be swept.

## Tips

Although you will soon become comfortable navigating these tabs, at first it may be best to complete the dialog in the following order:

1. On the [Tone Frequency tab](#), set the Sweep Type and Tone Frequencies.
  2. On the [Mixer Frequency tab](#), set a valid mixer configuration.
- Input center / fixed frequency CAN be set on both the [Tone Frequency tab](#) and the [Mixer Frequency tab](#). When you set one, the other is updated automatically.

The following tabs are available on the IMDx Setup dialog:

- [Tone Frequency tab](#)
- [Tone Power tab](#)
- [Mixer Frequency tab](#)
- [Mixer \(LO\) Power tab](#)
- [Mixer Setup tab](#)

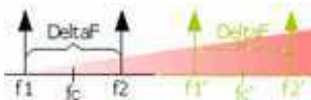
### Tone Frequency tab - IMDx Setup -dialog box help

The screenshot shows the 'Tone Frequency' tab of the 'IMDx Setup' dialog box. It is divided into several sections:

- Sweep Type:** A group box containing five radio buttons: 'Sweep f<sub>c</sub>' (selected), 'Sweep DeltaF', 'Power Sweep', 'CW', and 'LO Power Sweep'.
- X-Axis Display:** A section with an 'Annotation:' dropdown menu set to 'Input' and a small graph showing frequency components f<sub>1</sub>, f<sub>c</sub>, f<sub>2</sub>, f<sub>1</sub>Δ, f<sub>2</sub>Δ, and f<sub>2</sub>'.
- Sweep Settings:** A section with input fields for 'Start f<sub>c</sub>' (10.500000 MHz), 'Center f<sub>c</sub>' (13.25500000 GHz), 'Fixed DeltaF' (1.000000 MHz), 'Number Of Points' (201), 'Stop f<sub>c</sub>' (26.499500000 GHz), 'Span f<sub>c</sub>' (26.499000000 GHz), 'Main Tone IFBW' (1.000 kHz), and 'IM Tone IFBW' (1.000 kHz). There is also a checked checkbox for 'Reduce IF BW at Low Frequencies'.

Configures the Sweep Type and frequency range for IMD measurements.

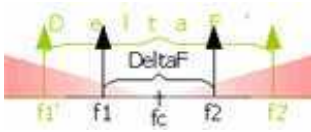
### Sweep Type and Sweep Settings



### Sweep $f_c$ (center frequency)

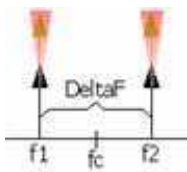
Maintaining a constant tone spacing (Fixed  $\Delta f$ ) and tone power,  $f_c$  is swept from Start  $f_c$  to Stop  $f_c$ . Center Frequency can also be specified as Center  $f_c$  and Span  $f_c$ .

At each  $f_c$ , the receivers are tuned to all of the required frequencies to measure the power of the appropriate tones.



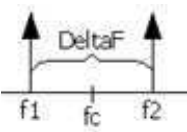
### Swept $\Delta f$ (tone spacing)

The specified  $f_c$  (center frequency) and tone power is held constant. The tone spacing is increased from Start  $\Delta f$  to Stop  $\Delta f$  in the specified number of points.



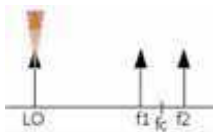
### Power Sweep

The main tone frequencies are specified as either  $f_1$  and  $f_2$ , or as  $f_c$  and  $\Delta f$ . These frequencies are held constant while the power of each main tone is varied from the Start-Power to Stop-Power in the specified number of power points. The power of each tone can be set (on the Power tab) individually or as a pair by checking Coupled Tone Power.



### CW

The main tone frequencies and power levels are held constant. Measurements are taken for the specified number of points. The X-axis is number of points.



### LO Power Sweep

The main tone frequencies and power levels are held constant. Measurements are taken for the specified number of points. The X-axis is LO Power.

**X-Axis Display** Select the mixer range to display on the X-Axis.

**Number of Points** Enter the number of data points for each sweep. See Limited Number of Acquisitions.

### IFBW

The [Narrowband IF path](#) is used for IMD measurements to help reduce spurious responses. Because the narrowband filter has a bandwidth of about 28 kHz, using an IFBW greater than 30 kHz does nothing to improve measurement accuracy.

**Main Tone and IM Tone IFBW** IF Bandwidth is specified separately for the main tones ( $f_1$  and  $f_2$ ) and for the intermodulation tones. This allows the higher-power main tones to be accurately measured at a higher - and faster - IFBW, while the lower-power product tones to be accurately measured a lower - and slower - IFBW.

**Reduce IF BW at Low Frequencies** - On the PNA-X, the trace noise becomes worse below 748 MHz. This is especially obvious between 10 MHz and 45 MHz. When this box is checked, the PNA-X uses a smaller IF Bandwidth than the selected value at frequencies below 748 MHz. [Learn more about the selected values.](#)

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## Tone Power tab - IMDx Setup -dialog box help

☒ Power On (All Channels) ☒ Coupled Tone Power

DUT Input Port  
Input Port: Port 1  
Source Attenuator: 0 dB  
Receiver Attenuator: 0 dB  
Source Leveling Mode: Internal  
f1 Power: -15.00 dBm  
f2 Power: -15.00 dBm

DUT Output Port  
Output Port: Port 2  
Source Attenuator: 10 dB  
Receiver Attenuator: 0 dB

RF2 tone power offset to compensate for combiner loss

Swept Tone Power Settings  
Start Stop Step  
f1 Power: -20.00 dBm -10.00 dBm 0.200 dB  
f2 Power: -20.00 dBm -10.00 dBm 0.200 dB

Path Configuration...

Configures RF tone power and tone power sweep settings for IMD measurements.

**Power ON (All channels)** Check to immediately turn ON or OFF ALL PNA-X internal RF Sources for all channels.

**Coupled Tone Power** Check to set the same power level for each main tone using the f1 Power setting. Clear to set different f1 and f2 power levels.

### DUT Input Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Input Port** Choose Port 1 or Port 3 (with limitations).

**Source Attenuator** Specifies the port 1 attenuator. This attenuator affects the range of available power into the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** This attenuation setting protects the A receiver from damage or compression.

**Source Leveling** Specifies the leveling mode. Choose Internal or Receiver R1. [Learn how to configure Receiver Leveling.](#) Open Loop should only be used when doing [Wideband Pulse measurements.](#)

**f1 Power** Specify the power level for f2. Choose a value between -30 dBm and +30 dBm. When Coupled Tone Power is checked, power is set for both f1 and f2 tones.

**f2 Power** Specify the power level for f2. Choose a value between -30 dBm and +30 dBm.

### DUT Output Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Output Port** Choose Port 2 or Port 4 (with limitations).

**Source Attenuator** This setting is used to improve the load match at the DUT output. Select 0 dB for power levels up to 10 dBm, and increase by 10 dB for every 10 dBm more output power.

**Receiver Attenuator** Specifies the attenuator setting for port 2. When the power into the receiver test port is around +10 dBm, the PNA-X receiver may be in compression. However, with receiver attenuation, lower input power levels may become too noisy to make accurate power measurements. In this case, lower IFBW for the IM tones to reduce noise. [Learn more about Receiver Attenuation.](#)

## Swept Tone Power Settings

**f1 and f2 Start and Stop Power** Available ONLY in Power Sweep. Sets the Start and Stop power levels for f1 and f2, either individually or together with Coupled Tone Power checked.

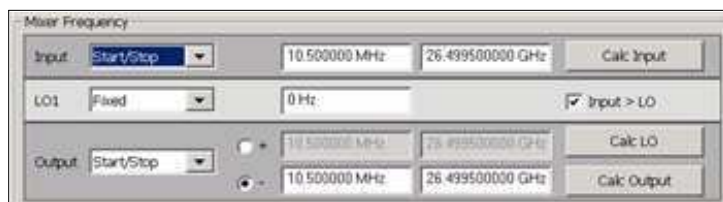
**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

### Highlighted Note: RF2 tone power offset to compensate for combiner loss.

This message appears when the f2 tone is being supplied by an external source. The tone power has been increased on the external source to compensate for loss through the internal combiner. For example, if the tone power at the DUT should be 0 dBm, the power out of the source will be about 15 dB higher.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## Mixer Frequency tab - IMDx Setup -dialog box help



The screenshot shows the 'Mixer Frequency' dialog box. It has three main sections: Input, LO1, and Output. Each section has a dropdown menu (Start/Stop or Fixed), a frequency input field, and a 'Calc' button. The Input section has a 'Calc Input' button. The LO1 section has a 'Calc LO' button. The Output section has a 'Calc Output' button. There is also a checkbox labeled 'Input > LO'.

### Rules for Configuring a Mixer

A **Red** message across the bottom of the dialog indicates that one or more of the following settings are invalid:

- A IMDx Sweep type that is NOT valid with the specified Input, LO, or Output sweep setting (Fixed or Swept). [Learn more](#).
- INPUT or OUTPUT frequencies cannot be outside the range of the PNA.
- Any combination of INPUT and LO which results in an OUTPUT that sweeps through zero Hz is NOT allowed.

**Frequency Format** The Input, LO1, and Output frequency information can be specified using Start/Stop (Swept) or Fixed. [See valid combinations](#).

**Frequencies** Enter the frequency values for each of the DUT ports.

**Input > LO** Removes ambiguity when using the Calculate button to determine the INPUT frequency.

- Check if the INPUT is GREATER THAN the LO
- Clear if the INPUT is LESS THAN the LO

This checkbox is only used when all 3 of the following conditions are TRUE:



(If ALL 3 are NOT true, the PNA-X does not read these boxes).

- Difference (Low) sideband is selected for the corresponding Calculate button AND
- Output frequency is less than the LO frequency AND
- Either Calculate button is used to calculate the Input frequency.

[To learn more, see this example.](#)



**Mixer-Product Selector** Determines whether the receivers will tune to the Sum (+) or the Difference ( - ) of the Input and LO frequencies.

**Calculate buttons** Calculates frequency settings based on your other mixer settings. For example, enter the Input frequency range and LO1 frequency range, then press **Calc Output**. The PNA-X will calculate and display the Output frequencies.

### About Mixer (\*.mxr) Files

The mixer setup files that are used with IMDx ARE compatible with the mixer setup files used with [FCA](#). However, IMDx files also contain IMD setup information that is NOT included with FCA files.

- When FCA files are recalled into IMDx, the PNA-X attempts a 'best guess' at missing information.
- When IMDx files are recalled into FCA, the extra information is ignored.
- When .mxr files are recalled into an IMx Spectrum channel with any range being swept, the recall is ignored and an error message appears.

**Note:** A .mxr file includes an LO source name. However, It does NOT include the LO Source configuration. Therefore, when using a .mxr file that was created on a different PNA, the PNA-X will display an error if does not find the LO Source configuration using EXACTLY the same LO source name.

**Load** Loads a previously-configured mixer attributes file (.mxr).

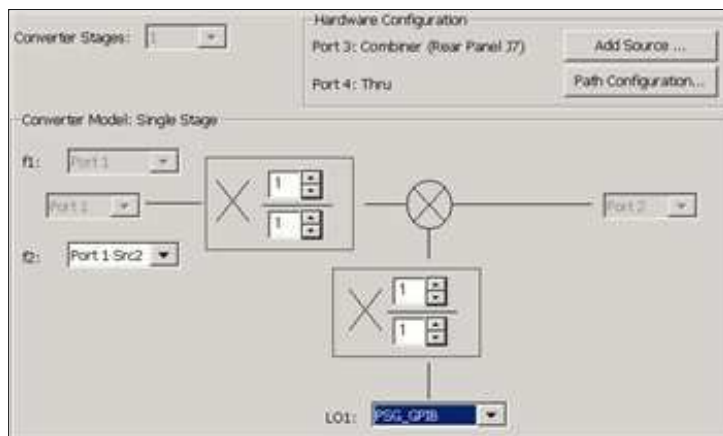
**Save** Saves the settings for your mixer/converter test setup to a mixer attributes file (.mxr).

**Apply** Applies the settings for your mixer/converter test setup to the measurement. The mixer setup dialog box remains OPEN.

**OK** Applies the settings for your mixer/converter test setup to the measurement. The mixer setup dialog box CLOSES.

**Cancel** Closes the mixer setup dialog box and does NOT apply the settings.

## Mixer Setup tab - IMD Setup -dialog box help



**Converter Stages** Only 1 stage converters are allowed.

### Hardware Configuration

Shows the path configuration for Port 3 and Port 4 switches. See [Hardware Configurations](#).

**Add Source** Click to start the [Select Sources dialog](#).

**Path Configuration** click to start the [Path Configuration dialog](#).

### Fractional Multiplier

The combination of (numerator / denominator) forms a fractional value that is multiplied by the input and LO frequency ranges. These values are used to Calculate the response frequency of the PNA-X receiver for the converter output. Use the fractional multipliers to:

- replicate the action of harmonic mixers
- replicate the action of multipliers and dividers that may exist in your test setup
- tune the PNA-X receiver frequency to a harmonic of the mixer/converter

The range for the numerator and denominator of a fractional multiplier is from +1 to +10. Negative values are NOT allowed.

**LO1** Select a source to be used as the LO.

**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## Mixer (LO) Power tab - IMD Setup -dialog box help

Power On (All Channels) ☒

LO1: MXG1 Power Settings

LO1 Power: -10.00 dBm

Source Leveling Mode: [Dropdown]

LO2: MXG2 Power Settings

LO2 Power: -10.00 dBm

Source Leveling Mode: [Dropdown]

Port Settings

Port 3

Source Attenuator: 0 dB

Receiver Attenuator: 0 dB

Port 4

Source Attenuator: 0 dB

Receiver Attenuator: 0 dB

Path Configuration...

Configures LO Power settings.

**Power ON (All channels)** Check to immediately turn ON or OFF ALL PNA-X internal RF Sources for all channels.

### LO1: <source> Power Settings

**LO1 Power** Sets the power level for LO1.

**Source Leveling** When using internal source for LO1, always set to Internal.

### LO2: <source> Power Settings

Only 1-stage converters are allowed.

### Port Settings

**Port 3** Used when an LO is supplied through the rear panel and out port 3. [See Hardware Configurations.](#)

**Source Attenuator** Specifies the port 3 attenuator. This attenuator affects the range of available power into the LO port of the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** For future use.

**Port 4** Used when an LO is supplied through the rear panel and out port 4. [See Hardware Configurations.](#)

**Source Attenuator** Specifies the port 4 attenuator. This attenuator affects the range of available power into the LO port of the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** For future use.

**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## How to specify IMDx Parameters

### Using front-panel HARDKEY [softkey] buttons

1. Press **MEAS**
2. then **[More Meas...]**

### Using a mouse with PNA-X Menus

1. Click **Response**
2. then **Measure**
3. then **More Meas...**

## Programming Commands

## IMDx New Trace dialog box help

Param. Name	Type	Tone Select	Order	Measure At
<input type="checkbox"/> PwrMain	Tone Power	Avg	1	DUT OUT
<input type="checkbox"/> IM3	IMD Relative to Carrier	Avg	3	DUT OUT
<input type="checkbox"/> OIP3	Output Referred Intcpt. Pt.	Avg	3	DUT OUT
<input type="checkbox"/> IIP3	Input Referred Intcpt. Pt.	Avg	3	DUT OUT
<input type="checkbox"/> Pwr2	Tone Power	Avg	2	DUT OUT

Select All Clear All

1 Channel Number ☐ Create In New Window ☒ Auto-Create Windows

OK Apply Cancel Help

Use this dialog to select IMDx parameters to measure and display. Up to five parameters can be selected at one time, then click **Apply** to create those traces, then select more without closing the dialog. There is no limit to the [number of traces and windows](#) allowed in the PNA.

**Note:** Calculations are NOT performed to determine if the frequency of a selected intermod (Order) product will be within the frequency range of the PNA-X. Measurements that fall outside of the frequency range of the PNA-X are displayed as -200 dB.

**Param Name** This name is built from the selected **Type** and **Order**. Once built, check to measure this parameter.

### Type

Select the type of Swept IMDx measurement to make. The characters in (parenthesis) are used in the Param Name. Choose from the following:

- **Tone Power (Pwr)** Measures the absolute power of the specified (ordered) tone. [Learn more.](#)
- **IMD Relative to Carrier (IM)** Measures the power of the specified (ordered) tone relative to the power of the f1 or f2 tone. [Learn more.](#)
- **Input Referred Intercept Point (IIP)** From DUT measurements, calculates the theoretical power level at which the specified (ordered) intermod product will be the same power level as the carrier at the output of the DUT. The input power to the DUT at which this output power occurs is reported and displayed. [Learn more.](#)
- **Output Referred Intercept Point (OIP)** From DUT measurements, calculates the theoretical power level at which the specified (ordered) product will be the same power level as the main tone at the output of the DUT. This value is reported and displayed. [Learn more.](#)
- **CTB Band-Edge Distortion (CTBE)** Composite 'Triple Beat' Distortion - 3rd order DUT OUT only. [Learn more.](#)
- **CTB Mid-Band Distortion (CTB)** Composite 'Triple Beat' Distortion - 3rd order DUT OUT only. [Learn more.](#)
- **CSO Distortion (CSO)** 2nd order DUT OUT only. [Learn more.](#)
- **XMOD 3rd Order Crossmod.** 3rd order DUT OUT only. [Learn more.](#)
- **Tone Gain (ToneGain)** From tone power measurements, calculates the Output Tone power / Input Tone power for the specified tones. [Learn more.](#)

### Tone Select

Select the tone and intermodulation product to be measured.

**Hi** and **Lo** are appended to the Param Name when selected.

Nothing is appended to the Param Name when Avg is selected.

Choose from the following:

- **High** Measure the specified (ordered) tone on the High side of the main tones.
- **Low** Measure the specified (ordered) tone on the Low side of the main tones.
- **Avg** Measure both the High and Low tones and calculate the dB average power level. If either the High or Low side falls outside the frequency range of the PNA, then Avg will not be available, and the measurement shows as zeroes ( -200 dB in Log Mag format).

### Important Notes

- If either the High or Low side falls outside the frequency range of the PNA, then **Avg** is NOT allowed, and the measurement shows as zeroes ( **-200 dB** in Log Mag format).
- When displaying 2nd order traces, **Avg** is NOT allowed, even when both the Lo and Hi products are displaying valid data. This is because 2 Low and 2 High products are usually very different from one another.

### Order

Specify the intermodulation product to measure. Choose from 1, 2, 3, 5, 7, 9. 2nd order products are more likely to be outside of the PNA-X frequency range than odd order products.

**Main** is appended to the Param Name when 1 is selected.

Otherwise, the tone number is appended to the parameter name.

### Measure At

Measure the selected parameter at either the DUT **Input** or **Output**.

- **In** is appended to the Param Name when Input is selected.
- Nothing is appended to the Param Name when Output is selected.

**Select All** Checks all available parameters.

**Clear All** Clears all checked parameters.

### Channel / Window Selections

**Channel Number** Select the channel for the new traces.

#### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the [PNA-X traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See [PNA-X number of windows limitation](#).

**Apply** Creates the measurements. The dialog remains open to allow you to create more measurements.

### IMDx Calibration

Calibration for IMDx is exactly the same as [calibration for IMD](#) with the following exception:

- You can choose to perform a source power cal of the LO source. If the LO is a fixed frequency, this step is performed very fast.

The results of an IMDx calibration are very similar to the results that are achieved from an [SMC calibration](#).

---

Last Modified:

12-Apr-2010	Removed some fixturing limitations
11-Aug-2009	Added limited port mapping
13-Jul-2009	Added 2nd order notes
2-Feb-2009	New topic

## IMx Spectrum for Converters

---

IMx Spectrum measurements are a distinct measurement class and therefore always made in a separate channel. This topic discusses all aspects of an IMx Spectrum measurement.

- [Requirements and Limitations](#)
- [How an IMx Spectrum Channel Works](#)
- [Create an IMx Spectrum Channel](#)
- [Select IMx Spectrum Parameters](#)
- [Start the IMx Spectrum Setup dialog](#)
  - [Frequency tab](#)
  - [Tone Power tab](#)
  - [Mixer Frequency tab](#)
  - [Mixer Power tab](#)
  - [Mixer Setup tab](#)
- [Calibration](#)

### See Also

- [Swept IMD Measurements](#)
  - [Programming commands](#)
- 

## Requirements and Limitations with IMx Spectrum

The IMx Spectrum channel requires a PNA-X with dual sources and the internal combiner. This is the same configuration that is required for all IMD App channels. [See requirements](#)

The following PNA-X features are NOT available in an IMx Spectrum channel:

- [Unratioed receiver measurements](#) (A, B, R)
- [ECal User Characterization](#)
- [FOM](#) or [FCA](#)
- [Copy Channel](#)
- [Time Domain](#)

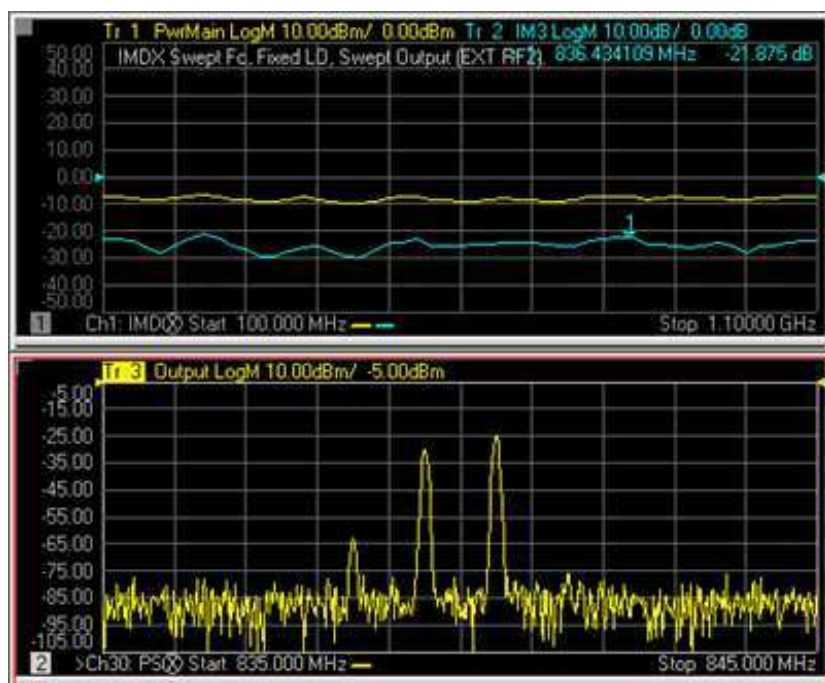


- [Balanced measurements](#)
- Save [Formatted Citifile](#) data.
- [Save SnP data.](#)
- [External sources](#)
- [External Test Set Control](#) (Option 551)
- [Interface Control](#)
- [Port Extensions](#)
- [Fixturing Simulator](#)
- [Integrated Narrowband](#) or [Narrowband Pulse App](#)

## How an IMx Spectrum for Converters Channel Works

Before reading this topic, you should become familiar with [IMD Concepts](#).

The IMx Spectrum channel provides a traditional Spectrum Analyzer view of the intermodulation distortion behavior of a converter output. Unlike the Swept IMDx channel, the main tones (F1 and F2) are fixed while the receiver is swept over a frequency range of interest in order to generate a display as shown below.



***IMx Spectrum trace (bottom) with Swept IMDx traces (top)***

A typical Spectrum Analyzer does NOT have a signal source. This one does. The signal source, or stimulus, settings for the F1 and F2 main tones can be set in either the IMx Spectrum channel or the Swept IMDx channel. These settings include the frequencies and power levels of the main tones.

## Receiver Settings

Although the stimulus settings can be set in either the IMx Spectrum channel or the Swept IMDx channel, the receiver settings are set ONLY in the Frequency tab of the [IMx Spectrum dialog](#). These settings include how many tone products to view, which determines the center and frequency span and the resolution bandwidth. You can also set a Linear sweep type, then enter an arbitrary receiver frequency range.

You can choose from several resolution bandwidths which also determines the number of data points used in the channel. The higher the Res BW, the fewer the number of data points.

The IMx Spectrum channel performs multiple measurements for each data point in order to reject unwanted images which are generated by the PNA-X internally. This provides a high degree of confidence that signals captured in a trace are real and are not spurious responses generated in the measurement process.

## Differences from IM Spectrum for Amplifiers

An IMx Spectrum for Converters channel has the following important differences from an [IM Spectrum channel for Amplifiers](#).

- In an IMx Spectrum channel, **all converter frequencies are fixed**. (Input, Output, LO, Tone Spacing, and Tone Power).
- The IMx Spectrum channel **CANNOT track** with the IMDx channel, as it can in IMD for Amplifiers.

## Create an IMx Spectrum Channel

An IMx Spectrum channel can be created independently, or from an existing Swept IMDx channel.

### To create an independent IMx Spectrum channel

1. On the PNA-X front panel, press **Meas** then **[Measurement Class]**
2. Select **IMx Spectrum**, then either:
  - **OK** delete the existing measurement, or
  - **New Channel** to create the measurement in a new channel.

### To create an IMx Spectrum channel from an existing Swept IMD channel

This creates or configures an existing IMx Spectrum channel based upon the configuration of the Swept IMD channel at the Marker frequency.

1. With a Swept IMD channel active, press the **MARKER** key on the front panel.
2. Move the marker to the data point of interest. It can be moved again later.
3. Press **[Marker Functions]** then **[Marker -> IMx Spectrum]**

## Select IMx Spectrum Parameters

## How to create a NEW (additional) IMx Spectrum trace

With the IMx Spectrum channel active and a **Tr1 Output** trace displayed:

### Using front-panel HARDKEY [softkey] buttons

- Press **TRACE 2**  
A new Tr2 Input trace appears.

### Using a mouse with PNA-X Menus

- Click **Trace/Chan**
  - then **New Trace**The [New Trace](#) dialog appears.

## To change the active (existing) IMx Spectrum parameter

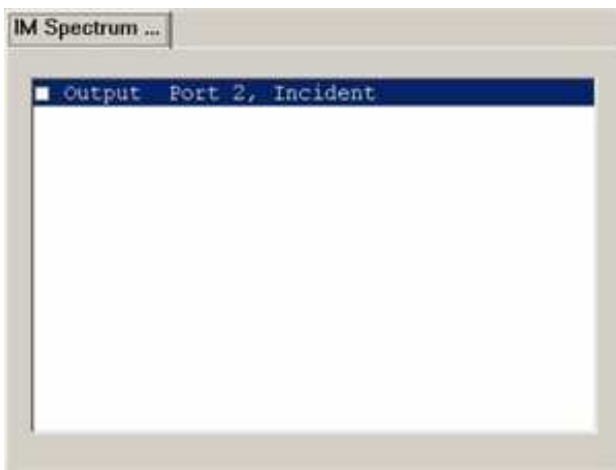
Select the trace to be replaced. Then:

- Press **MEAS**
- then **[select the parameter]**

- Click **Response**
  - then **Measure**
  - then **select the parameter**

◀ **Programming Commands** ▶

## New Trace - IMx Spectrum -dialog box help



**Note:** Only viewing the OUTPUT tones is supported.

- **Output** View signals OUT of the DUT and into PNA-X port 2 (B receiver).

**Select All** Checks all available parameters.

**Clear All** Clears all checked parameters.

### Channel / Window Selections

**Channel Number** Select the channel for the new traces.

#### Create in New Window

- Check to create new traces in a new window.
- Clear to create new traces in the active window. When the [PNA-X traces per window limitation](#) has been reached, no more traces are added.

**Auto-Create Windows** Check to create new traces in as many windows as necessary. See [PNA-X number of windows limitation](#).

## IMx Spectrum Setup Dialog

### How to start the IMx Spectrum Setup dialog

To provide quicker access, use the Setup softkey. [Learn how](#).

With an IMx Spectrum measurement active:

#### Using front-panel HARDKEY [softkey] buttons

1. Press **FREQ**
2. then **[IMx Spectrum Setup]**

#### Using a mouse with PNA-X Menus

1. Click **Stimulus**
2. then **Frequency**
3. then **IMx Spectrum Setup**

**Programming Commands**

### Frequency tab - IMx Spectrum -dialog box help

The screenshot shows the 'Frequency' tab of the IMx Spectrum dialog box. It is divided into several sections:

- Sweep Type:** Includes radio buttons for 'Linear', '2nd Order Spectrum', '3rd Order Spectrum', and 'Nth Order Spectrum'. The 'Nth Order Spectrum' option is selected, with 'Order N' set to 9.
- Resolution BW:** A text field showing '600.000 kHz'.
- Stimulus Settings:** Includes fields for 'fc (Tone Center)' (1.000000000 GHz), 'Delta F' (10.000000 MHz), 'f1' (995.000000 MHz), and 'f2' (1.005000000 GHz).
- Response Settings:** Includes fields for 'Start Spectrum' (950.000000 MHz), 'Stop Spectrum' (1.050000000 GHz), 'Center Spectrum' (1.000000000 GHz), and 'Span Spectrum' (100.000000 MHz).

**Note:** The number of data points in an IMx Spectrum channel =  $3 * \text{SPAN} / \text{ResBW}$ . This can NOT be changed directly.

Configures the Sweep Type and frequency range for IMx Spectrum measurements.

### Sweep Type

Provides several methods to tune the IMx Spectrum **receivers** (NOT stimulus) to view the power spectrum of arbitrary frequency ranges or various distortion products.

**Linear** Allows the start/stop or center/span receiver frequencies to be set arbitrarily for f1 and f2. Enter the Response Settings below.

**2nd Order** Couples the receiver frequency range to f1 and f2 to provide a convenient means of observing the spectrum surrounding ONLY the high-side 2nd order harmonic where Center =  $(f1 + f2)$ , Span =  $3 * (f2 - f1)$ . **Note:** The center frequency is NOT set on the main tones.

**3rd Order** Couples receiver frequency range to f1 and f2 to provide a convenient means of observing the spectrum surround the main-tones including the 3rd Order products where Center =  $(f1 + f2) / 2$ , Span =  $4 * (f2 - f1)$ .

**Nth Order** Couples receiver frequency range to f1 and f2, providing a convenient means of observing the spectrum surrounding the main-tones for an arbitrary span where Center =  $(f1 + f2) / 2$ , and Span =  $N * (f2 - f1)$ . This allows you to set the span arbitrarily, but have the center frequency track the main tone frequencies.

### Resolution BW

The IMx Spectrum channel utilizes a set of Gaussian filters instead of the standard PNA-X IF filters in order to provide similar behavior to traditional spectrum analyzers.

The comprehensive list of filters that are available for IMx Spectrum are: 3 MHz, 1 MHz, 600 kHz, 300 kHz, 150 kHz, 100 kHz, 60 kHz, **10 kHz**, 3 kHz, and 1 kHz.

**Note:** The **10 kHz** Res BW filter can generate image signals which are not a product of the DUT. You can verify the integrity of a questionable signal by switching to the 3 kHz or 60 kHz filter and looking for the image signal in the same location.

Not all of these filters are available for all measurements. Narrower filters are available for use with narrower frequency spans, and wider filters are available for wider spans.

### Stimulus Settings

Allows configuration of the main-tone frequencies.

- **fc (main-tone center frequency)** =  $(f1 + f2) / 2$
- **DeltaF (main-tone frequency separation)** =  $(f2 - f1)$ .
- **f1** = Low-side main-tone frequency
- **f2** = High-side main-tone frequency

### Response Settings

Allows arbitrary configuration of the start/stop or center/span frequency range of the receivers. Available ONLY for Linear sweep type.

- **Start Spectrum** = first frequency point of the power spectrum sweep
- **Stop Spectrum** = last frequency point of the power spectrum sweep
- **Center Spectrum** = (Start Spectrum + Stop Spectrum) / 2
- **Span Spectrum** = (Stop Spectrum – Start Spectrum)

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

### Tone Power tab - IMxSpectrum -dialog box help

Configures RF power settings for IMx Spectrum measurements.

**Power ON (All channels)** Check to immediately turn ON or OFF ALL PNA-X internal RF Sources for all channels.

**Coupled Tone Power** Check to set the same power for each main tone using the f1 Power setting. Clear to set f1 and f2 power levels individually.

#### DUT Input Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Input Port** Choose Port 1 or Port 3 (with limitations).

**Source Attenuator** Specifies the attenuator setting associated with the port connected to the input of the DUT. This attenuator will affect the range of available power into the DUT [Learn more about Source Attenuation.](#)

All PNA-X channels in continuous sweep must have the same attenuation value. [Learn more.](#)

**Receiver Attenuator** This attenuation setting protects the A receiver from damage.

**Source Leveling** Specifies the leveling mode. Choose Internal or Receiver R1. [Learn how to configure Receiver Leveling.](#) Open Loop should only be used when doing [Wideband Pulse measurements.](#)

**f1 Power** Power level for f1 and f2 when Coupled Tone Power is checked.

**f2 Power** Power level for f2. Available only when Coupled Tone Power is NOT checked.

## DUT Output Port

Beginning with PNA Rev. A.09.00, limited port mapping is available. [Learn more.](#)

**Output Port** Choose Port 2 or Port 4 (with limitations).

**Source Attenuator** This setting is used to improve the load match at the DUT output. Select 0 dB for power levels up to +10 dBm, and increase by 10 dB for every 10 dBm more output power.

**Receiver Attenuator** Specifies the attenuator setting for the receiver associated with the DUT output port. When the power into the receiver test port is around +10 dBm, the PNA-X receiver may be in compression. However, with receiver attenuation, lower input power levels may become too noisy to make accurate power measurements. [Learn more about Receiver Attenuation.](#)

### Note: RF2 tone power offset to compensate for combiner loss.

This message appears when the f2 tone is being supplied by an external source. The tone power has been increased on the external source to compensate for loss through the internal combiner. For example, if the tone power at the DUT should be 0 dBm, the power out of the source will be about 15 dB higher.

**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## Mixer Frequency tab - IMx Spectrum -dialog box help



ONLY Fixed Input, LO1, and Output settings are allowed.

**Frequencies** Enter the frequency values for each of the signals in your test setup.

**Input > LO** Removes ambiguity when using the Calculate button to determine the INPUT frequency.

- Check if the INPUT is GREATER THAN the LO
- Clear if the INPUT is LESS THAN the LO

This checkbox is only used when all 3 of the following conditions are TRUE:

(If ALL 3 are NOT true, the PNA-X does not read these boxes).

- Difference (Low) sideband is selected for the corresponding Calculate button AND
- Output frequency is less than the LO frequency AND
- Either Calculate button is used to calculate the Input frequency.

[To learn more, see this example.](#)



**Mixer-Product Selector** Determines whether the receivers will tune to the Sum (+) or the Difference ( - ) of the Input and LO frequencies.

**Calculate buttons** Calculates frequency settings based on your other mixer settings. For example, enter the Input frequency range and LO1 frequency range, then press **Calc Output**. The PNA-X will calculate and display the Output frequencies.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

## Mixer Power tab - IMx Spectrum -dialog box help

Power On (All Channels) ☒

LO1: MXG1 Power Settings

LO1 Power: -10.00 dBm

Source Leveling Mode: [Dropdown]

LO2: MXG2 Power Settings

LO2 Power: -10.00 dBm

Source Leveling Mode: [Dropdown]

Port Settings

Port 3

Source Attenuator: 0 dB

Receiver Attenuator: 0 dB

Port 4

Source Attenuator: 0 dB

Receiver Attenuator: 0 dB

Path Configuration...

Configures LO Power settings.

**Power ON (All channels)** Check to immediately turn ON or OFF ALL PNA-X internal RF Sources for all channels.

### LO1: <source> Power Settings

**LO1 Power** Sets the power level for LO1.

**Source Leveling** Not available. Always set to Internal.

### LO2: <source> Power Settings

Only 1-stage Converter are allowed.

### Port Settings

**Port 3** Used when an LO is supplied through the rear panel and out port 3. [See Hardware Configurations.](#)

**Source Attenuator** Specifies the port 3 attenuator. This attenuator affects the range of available power into the LO port of the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** For future use.

**Port 4** Used when an LO is supplied through the rear panel and out port 4. [See Hardware Configurations.](#)

**Source Attenuator** Specifies the port 4 attenuator. This attenuator affects the range of available power into



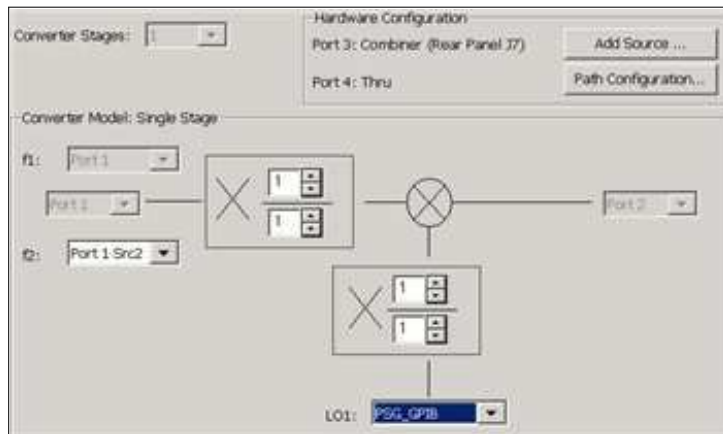
the LO port of the DUT [Learn more about Source Attenuation.](#)

**Receiver Attenuator** For future use.

**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

### Mixer Setup tab - IMx Spectrum -dialog box help



**Converter Stages** Only 1 stage converters are allowed.

#### Fractional Multiplier

The combination of (numerator / denominator) forms a fractional value that is multiplied by the input and LO frequency ranges. These values are used to Calculate the response frequency of the PNA-X receiver for the converter output. Use the fractional multipliers to:

- replicate the action of harmonic mixers
- replicate the action of multipliers and dividers that may exist in your test setup
- tune the PNA-X receiver frequency to a harmonic of the mixer/converter

The range for the numerator and denominator of a fractional multiplier is from +1 to +10. Negative values are NOT allowed.

**LO1** Select a source to be used as the LO.

**Path Configuration** Click to launch the [RF Path Configuration](#) dialog.

[Learn about the Load/Save \\*.mxr files](#), and other buttons across the bottom of all IMD Setup tabs.

### Calibration

A calibration of the IMx Spectrum channel is NOT performed using a calibration wizard.

An IMx Spectrum channel is calibrated from a Cal Set that is used on a Swept IMDx channel. The Cal Set can be applied to the IMx Spectrum channel using the Manage Cal Sets dialog ([Learn how](#)) or from the Marker =>IMx Spectrum softkey ([Learn how](#)).

However, a Swept IMDx channel with [Sweep Type = Power Sweep](#) can NOT be applied to a IMx Spectrum channel. This is because a Cal Set for power sweep contains only a CW frequency and the IMx Spectrum channel requires a swept frequency range. Zero Span is not supported in an IMx Spectrum channel.

A calibrated IMx Spectrum trace corrects the source and receiver power level accuracy of the displayed Tones.

[See Swept IMD Calibration](#)

[Learn how to apply a Cal Set to the IMx Spectrum channel](#)

---

Last Modified:

16-Sep-2009	Fixed Calibration
11-Aug-2009	Added limited port mapping
2-Feb-2009	New topic

## Time Domain

---

Time Domain allows you to view a device response as a function of time. The following are discussed in this topic:

- [Overview](#)
- [How the PNA Measures in the Time Domain](#)
- [Calibration for Time Domain](#)
- [Transmission Measurements](#)
- [Measurement Response Resolution](#)
- [Measurement Range and Alias Responses](#)
- [How to make Time Domain Settings](#)
- [Gating](#)
- [Window Settings](#)

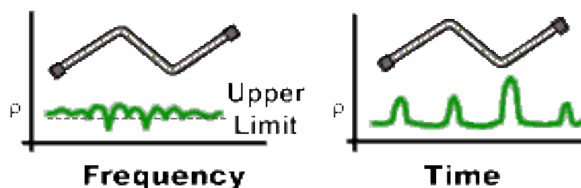
**Note:** Time Domain measurements are only available on PNAs with Option 010. See [PNA Options](#)

See the updated **App Note:** [Time Domain Analysis Using a Network Analyzer](#).

### Overview

In normal operation, the PNA measures the characteristics of a test device as a function of frequency. With Time Domain (opt 010), the frequency information is used to calculate the inverse Fourier transform and display measurements with time as the horizontal display axis. The response values appear separated in time, allowing a different perspective of the test device's performance and limitations.

The graphic below compares the same cable reflection measurement data in both the frequency and time domain. The cable has two bends. Each bend creates a mismatch or change in the line impedance.



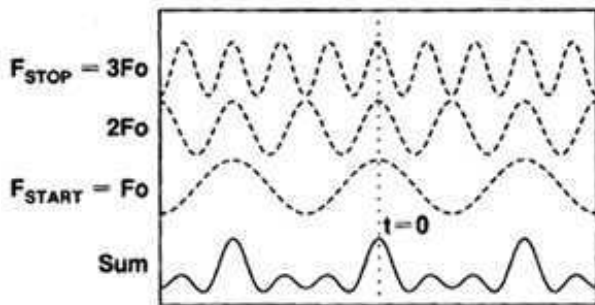
- The frequency domain S11 measurement shows reflections caused by mismatches in the cable. It is impossible to determine where the mismatches physically occur in the cable.
- The time domain response shows both the location and the magnitude of each mismatch. The responses indicate that the second cable bend is the location of a significant mismatch. This mismatch can be [gated out](#), allowing you to view the frequency domain response as if the mismatch were not present. Distance Markers can be used to pinpoint the distance of the mismatch from the reference plane.

## How the PNA Measures in the Time Domain

Time domain transform mode simulates traditional Time-Domain Reflectometry (TDR), which launches an impulse or step signal into the test device and displays the reflected energy on the TDR screen. By analyzing the magnitude, duration, and shape of the reflected waveform, you can determine the nature of the impedance variation in the test device.

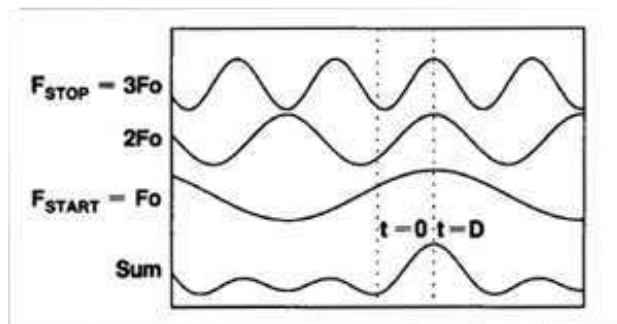
The PNA does not launch an actual incident impulse or step. Instead, a Fourier Transform algorithm is used to calculate time information from the frequency measurements. The following shows how this occurs.

A single frequency in the time domain appears as a sine wave. In the following graphic, as we add the fundamental frequency ( $F_0$ ), the first harmonic ( $2F_0$ ), and then the second harmonic ( $3F_0$ ), we can see a pulse taking shape in the Sum waveform. If we were to add more frequency components, the pulse would become sharper and narrower. When the PNA sends discrete frequencies to the test device, it is in effect, sending individual spectral pieces of a pulse separately to stimulate the test device.

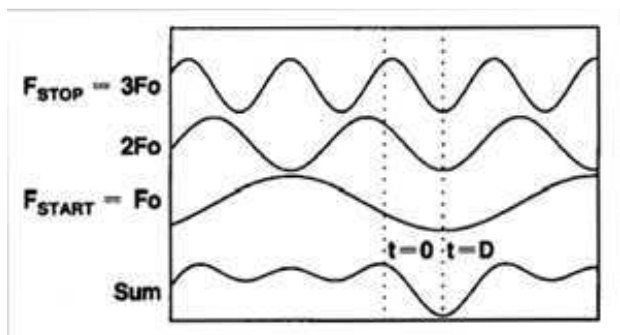


During an S11 reflection measurement, these incident signals reflect from the test device and are measured at the A receiver. This is when the time domain transform calculations are used to add the separate spectral pieces together.

For example, consider a short length of cable terminated with an open. All of the power in the incident signal is reflected, and the reflections are 'in-phase' with the incident signal. Each frequency component is added together, and we see the same pattern as the simulated incident would have looked (above). The magnitude of the reflection is related to the impedance mismatch and the delay is proportional to the distance to the mismatch. The x-axis (time) scale is changed from the above graphic to better show the delay.



Alternately, the same cable terminated with a short also reflects all of the incident power, but with a phase shift of 180 degrees. As the frequency components from the reflection are added together, the sum appears as a negative impulse delayed in time.



## Calibration for Time Domain

For simplicity, we have discussed incident signals reflecting off discontinuities in the test device. By far the most common network analyzer measurement to transform to time domain is a [ratioed](#) S11 measurement. An S11 reflection measurement does not simply display the reflections measured at the A receiver - it displays the ratio (or difference) of the A receiver to the Reference receiver. In addition, the S11 measurement can also be calibrated to remove [systematic errors](#) from the ratioed measurement. This is critical in the time domain as the measurement plane, the point of calibration, becomes zero on the X-axis time scale. All time and distance data is presented in reference to this point. As a result, both magnitude and time data are calibrated and very accurate.

The following shows where the time domain transform occurs in the PNA data flow: (see [Data Access Map](#))

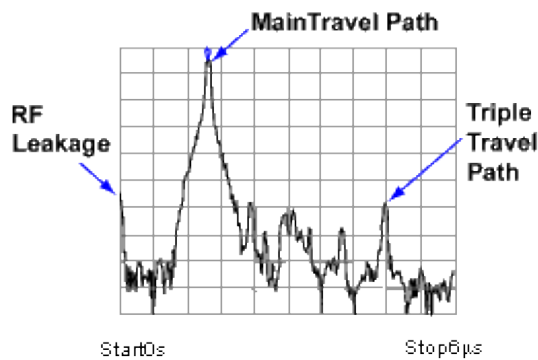
1. Acquire raw receiver (A and R1) data
2. Perform ratio (A/R1)
3. Apply calibration
4. Transform data to time domain
5. Display results

Therefore, although a time domain trace may be displayed, a calibration is always performed and applied to the frequency domain measurement which is not displayed.

## Transmission Measurements

The most common type of measurement to transform is an S11 reflection measurement. However, useful information can be gained about a test device from a transformed S21 transmission measurement. The frequency components pass through the test device and are measured at the B receiver. If there is more than one path through the device, they would appear as various pulses separated in time.

For example, the following transmission measurement shows multiple paths of travel within a Surface Acoustic Wave (SAW) filter. The largest pulse (close to zero time) represents the propagation time of the shortest path through the device. It may not be the largest pulse or represent the desired path. Each subsequent pulse represents another possible path from input to output.



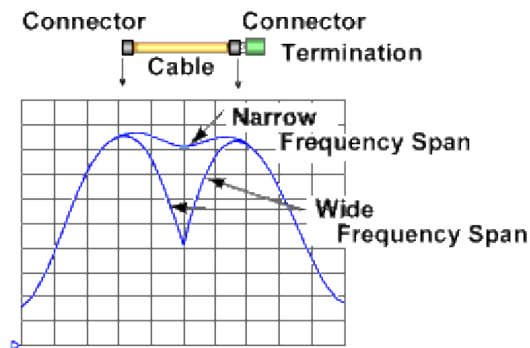
Triple travel is a term used to describe the reflected signal off the output, reflected again off the input, then finally reappearing at the output. This is best seen in a time domain S21 measurement.

## Measurement Response Resolution

In the previous paragraphs, we have seen that using more frequency components causes the assembled waveform to show more detail. This is known as measurement response resolution, which is defined as the ability to distinguish between two closely spaced responses.

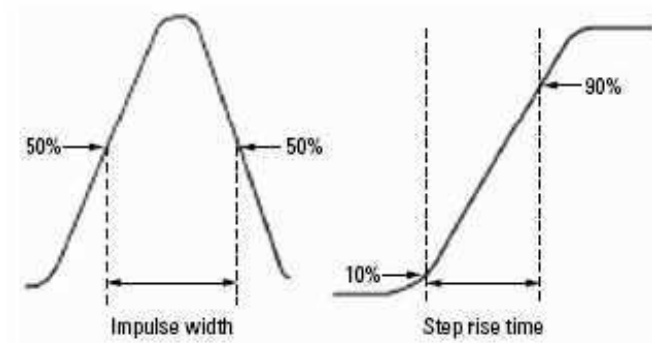
**Note:** Adjusting the [transform time settings](#) improves **display** resolution, but not measurement resolution.

The following graphic shows the effect of both a narrow and wide frequency span on the response resolution. The wider frequency span enables the analyzer to resolve the two connectors into separate, distinct responses.



## Resolution Formula

For responses of equal amplitude, the response resolution is equal to the 50% (-6 dB) points of the impulse width, or the step rise time which is defined as the 10 to 90% points as shown in the following image.



The following table shows the **approximated** relationship between the frequency span and the window selection on response resolution for responses of equal amplitude.

Window	Low-pass step (10% to 90%)	Low-pass impulse (50%)	Bandpass impulse
Minimum	0.45 / f span	0.60 / f span	1.20 / f span
Normal	0.99 / f span	0.98 / f span	1.95 / f span
Maximum	1.48 / f span	1.39 / f span	2.77 / f span

For example, using a 10 GHz wide frequency span and a normal window in Bandpass impulse mode, response resolution (in time) equals:

- Time Res = 1.95 / frequency span
- Time Res = 1.95 / 10 GHz
- Time Res = 195 ps

To calculate the physical separation (in distance) of the responses which can be resolved, multiply this value times the speed of light (c) and the relative velocity (Vf) of propagation in the actual transmission medium. In this case, Vf = 0.66 for polyethylene dielectric.

- Distance Res = 195 ps x c x Vf
- Distance Res = 195 ps x (2.997925 E8 m/s) x .66
- Distance Res = 38 mm

For reflection measurements, because of the 2-way travel time involved, this means that the minimum resolvable separation between discontinuities is half of this value or 19 mm.

Although a wider frequency span causes better measurement resolution, the [measurement range](#) becomes limited. Also, increasing the frequency range can cause a measurement calibration to become invalid. Be sure to adjust the frequency span BEFORE performing a calibration.

## Measurement Range and Alias Responses

Measurement range is the length in time in which true time domain responses can be seen. The measurement range should be large enough to see the entire test device response without encountering a repetition (alias) of the response. An alias response can hide a true time domain response.

To increase measurement range in both modes, change either of these settings:

- Increase the number of points
- Decrease the frequency span

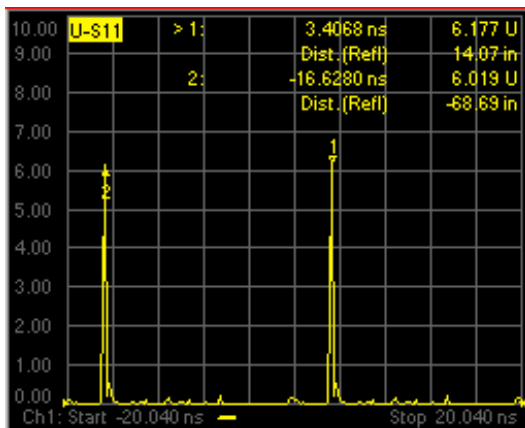
### Notes:

- After making these settings, you may need to adjust the [transform time settings](#) to see the new measurement range.
- Decreasing the frequency span degrades [measurement resolution](#).
- Make frequency span and number of points settings BEFORE calibrating.
- Maximum range also depends on loss through the test device. If the returning signal is too small to measure, the range is limited regardless of the frequency span.

## Alias Responses

An alias response is not a true device response. An alias response repeats because each time domain waveform has many periods and repeats with time (see [How the PNA Measures in the Time Domain](#)). Alias responses occur at time intervals that are equal to  $1/\text{frequency}$  step size.

The PNA adjusts the [transform time settings](#) so that you should only see one alias free range on either side (positive and negative) of zero time. However, these settings are updated only when one of the toolbar settings are changed.



To determine if a response is true, put a marker on the response and change the frequency span. A true device response will not move in time. An alias response will move.

For example, in the above graphic, the marker 1 response occurs at 14.07 inches. When the frequency span is changed, this response remains at 14.07 inches. The marker 2 response moves.

## Range Formula



You can calculate the alias-free measurement range (in meters) of the PNA using the following formula for **TDR** (reflection) measurements:

$$\text{Range (meters)} = (1/\Delta f) \times V_f \times c$$

Where:

- $\Delta f$  = frequency step size (frequency span/number of points-1)
- $V_f$  = the velocity factor in the transmission line
- $c$  = speed of light = 2.997925 E8 m/s

For example: For a measurement with 401 points and a span of 2.5 GHz, using a polyethylene cable ( $V_f = 0.66$ )

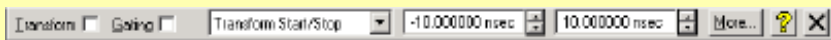
- Range =  $(1 / (2.5\text{E}9 / 400)) \times 2.997925 \text{ E}8 \text{ m/s} \times 0.66$
- Range =  $6.25\text{E}6 \times 2.997925 \text{ E}8 \text{ m/s} \times 0.66$
- Range = 32 meters

In this example, the range is 32 meters in physical length. To prevent the time domain responses from overlapping or aliasing, the test device must be 32 meters or less in physical length for a transmission measurement.

To calculate the one-way distance for a reflection measurement rather than round-trip distance, simply divide the length by 2. In this case, the alias-free range would be 16 meters.

### How to make Time Domain Settings

The following launches the [Time Domain toolbar](#)



On the toolbar, click **More...** to launch the Time Domain dialog box

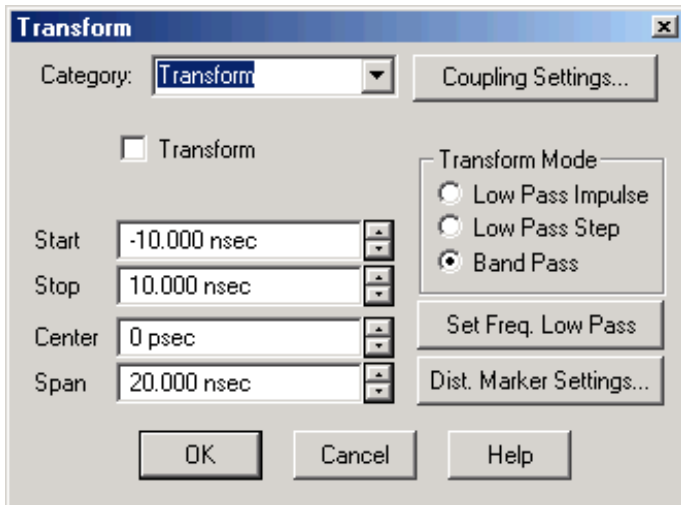
#### Using front-panel HARDKEY [softkey] buttons

1. Press **ANALYSIS**
2. then **[Transform]**
3. then **[More]**
4. then **[Transform Tool]**

#### Using a mouse with PNA Menus

1. Click **Marker/Analysis**
2. then **Transform**

◀ Programming Commands ▶



### Transform dialog box help

**Category** Select Transform, Window, or Gating

**Transform** Turns time domain transform ON and OFF.

**Coupling Settings** Launches the [Trace Coupling Settings](#) dialog box.

#### Time Settings

The following settings adjust the **display resolution**, allowing you to zoom IN or OUT on a response. They do NOT adjust [measurement range](#) or [measurement resolution](#).

These settings automatically update (when one of these values are updated) to limit the display to one [alias-free response](#) on either side of zero time.

**Start** Sets the transform start time that is displayed on the PNA screen.

**Note:** Zero (0) seconds is always the [measurement reference plane](#). Negative values are useful if moving the reference plane.

**Stop** Sets the transform stop time that is displayed on the PNA screen.

**Center** Sets the transform center time that is displayed in the center of the PNA screen.













**Span** Sets the transform span time that is split on either side of the Center value.

#### Transform Mode

Transform modes are three variations on how the time domain transform algorithm is applied to the frequency domain measurement. Each method has a unique application.

Mode	Benefit - application	Limitation
<b>Low pass Impulse</b>	Highest resolution. Most useful for seeing small responses in devices that pass low frequencies, such as cables.	In both Low pass modes, frequencies down to DC and negative frequencies are extrapolated. Therefore, the Start frequency is adjusted when you click <a href="#">Set Freq.Low Pass</a>  Because this will affect calibration accuracy, be sure to calibrate AFTER completely setting up your time domain measurement.
<b>Low pass Step</b>	Easiest to identify inductive and capacitive discontinuities in devices that pass low frequencies, such as cables.	
<b>Band pass Impulse</b>	Easiest method - can be used with any frequency sweep. Most useful for measuring band limited devices such as filters and DC blocked cables.	Does NOT show capacitive and inductive reactance  For the same frequency span and number of points, band pass mode has twice the impulse width, which hides closely spaced responses degrading the response resolution.

The following chart shows how to interpret results from various discontinuity impedances using Low pass Step and either Low pass or Band pass Impulse modes.

IMPEDANCE	STEP RESPONSE	IMPULSE RESPONSE
OPEN	 Unity Reflection	 Unity Reflection
SHORT	 Unity Reflection = 180	 Unity Reflection = 180
RESISTOR $R > Z_0$		
RESISTOR $R < Z_0$		
INDUCTOR		
CAPACITOR		

### Effect on Measurement Range

**Band pass mode** - measurement range is inversely proportional to frequency step size.

**Low pass mode** - measurement range is inversely proportional to the fundamental (start ) frequency AFTER clicking Set Freq. Low Pass.

### Set Freq. Low Pass USE ONLY IN LOW PASS MODES

Recomputes the start frequency and step frequencies to be harmonics of the start frequency. Start frequency is computed by the following formula: **Low Pass Start Frequency = Stop Frequency / Number of points.**

The computed value must always be greater than or equal to the analyzer's minimum frequency.

**Note:** The number of points or stop frequency may be changed in order to compute this value.

**Distance Marker Settings** Launches the [Distance Marker Settings](#) dialog box.

## Gating

Perhaps the most beneficial feature of time domain transform is the Gating function. When viewing the time domain response of a device, the gating function can be used to "virtually" remove undesired responses. You can then simultaneously view a frequency domain trace as if the undesired response did not exist.. This allows you to characterize devices without the effects of external devices such as connectors or adapters.

**Note:** When a discontinuity in a test device reflects energy, that energy will not reach subsequent discontinuities. This can "**MASK**", or hide, the true response which would have occurred if the previous discontinuity were not present. The PNA Gating feature does NOT compensate for this.

The following measurements images show a practical example how to use and perform gating. The test device is a 10inch cable, then a 6 dB attenuator, terminated with a short. The following four discontinuities are evident in window 2, from left to right:

1. A discontinuity in the test system cable which appeared after calibration. It is identified by marker 2 at -10.74 inches (behind the reference plane).
2. A discontinuity in the 10 inch device cable shortly after the reference plane.
3. The largest discontinuity is the attenuator and short shown by marker 1 at -12.67 dB ( 6 dB loss in both forward and reverse direction).
4. The last discontinuity is a re-reflection from the device cable.

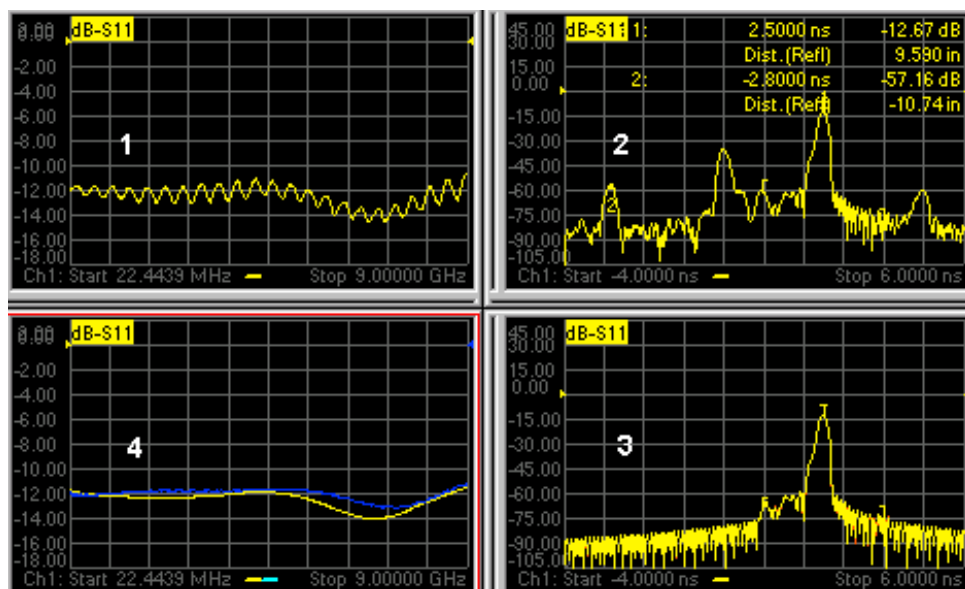
We will gate IN the attenuator response. All other responses will be gated OUT.

**Window 1.** Create original S11 frequency domain trace. Shows ripple from all of the reflections.

**Window 2.** Create a new S11 trace - same channel; new window. Turn Transform ON.

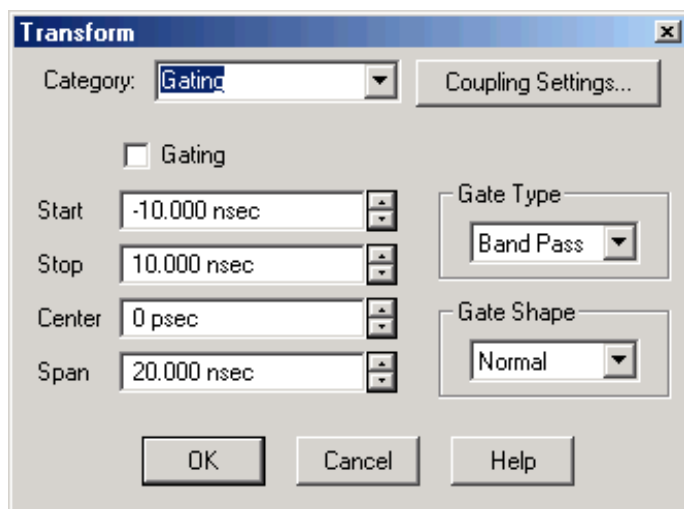
**Window 3.** On the transformed trace, turn gating ON. Center the gate on the large discontinuity (2.500ns). Adjust gate span to completely cover the discontinuity. Select Bandpass gating type.

**Window 4.** On the original frequency measurement, turn Gating ON (Transform remains OFF). View the measurement without the effects of the two unwanted discontinuities. The blue trace is a measurement of the 6 dB attenuator with the unwanted discontinuities PHYSICALLY removed. The difference between the two traces in window 4 is the effect of "[masking](#)".



[Learn how to launch the Transform dialog box](#)

## Programming Commands



### Transform Gating dialog box help

**Gating** Turns Gating ON and OFF.

**Coupling Settings** Launches the [Setup Trace Coupling](#) dialog box.

**Start** Specifies the start time for the gate.

**Stop** Specifies the stop time for the gate.

**Center** Specifies the value at the center of the area that is affected by the gating function. This value can be anywhere in the analyzer range.

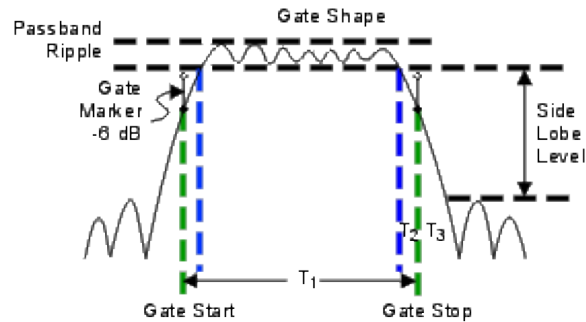
**Span** Specifies the range to either side of the center value of area that is affected by the gating function.

**Gate Type** Defines the type of filtering that will be performed for the gating function. The gate start and stop

flags on the display point toward the part of the trace you want to keep.

- **Bandpass** - KEEPS the responses within the gate span.
- **Notch** - REMOVES the responses with the gate span.

**Gate Shape** Defines the filter characteristics of the gate function. Choose from Minimum, Normal, Wide, Maximum



Gate Shape	Passband Ripple	Sidelobe Levels	Cutoff Time	Minimum Gate Span
Minimum	$\pm 0.1$ dB	-48 dB	$1.4/\text{Freq Span}$	$2.8/\text{Freq Span}$
Normal	$\pm 0.1$ dB	-68 dB	$2.8/\text{Freq Span}$	$5.6/\text{Freq Span}$
Wide	$\pm 0.1$ dB	-57 dB	$4.4/\text{Freq Span}$	$8.8/\text{Freq Span}$
Maximum	$\pm 0.01$ dB	-70 dB	$12.7/\text{Freq Span}$	$25.4/\text{Freq Span}$

**Cutoff time** -- is the time between the stop time (-6 dB on the filter skirt) and the peak of the first sidelobe. The diagram below shows the overall gate shape and lists the characteristics for each gate shape.

- $T_1$  is the gate span, which is equal to the stop time minus the start time.
- $T_2$  is the time between the edge of the passband and the 6 dB point, representing the cutoff rate of the filter.
- $T_3$  is the time between the 6 dB point and the edge of the gate stopband.
- For all filter shapes  $T_2$  is equal to  $T_3$ , and the filter is the same on both sides of the center time.

**Minimum gate span** -- is twice the cutoff time. Each gate shape has a minimum recommended gate span for proper operation. This is a consequence of the finite cutoff rate of the gate. If you specify a gate span that is smaller than the minimum span, the response will show the following effects:

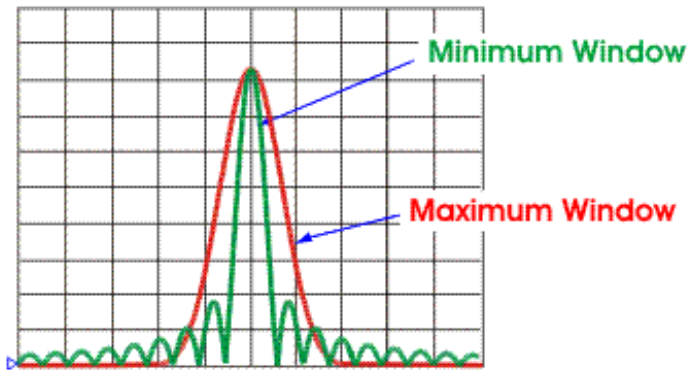
- distorted gate shape that has no passband
- distorted shape
- incorrect indications of start and stop times

- may have increased sidelobe levels

## Window Settings

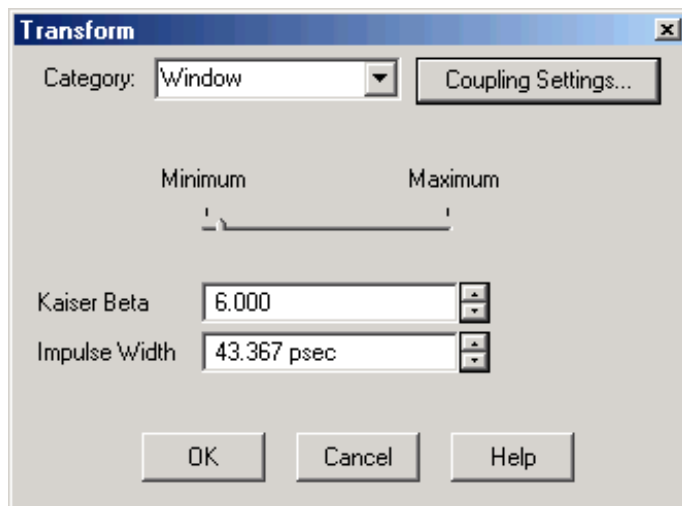
There are abrupt transitions in a frequency domain measurement at the start and stop frequencies, causing overshoot and ringing in a time domain response. The window feature is helpful in lessening the abruptness of the frequency domain transitions. This causes you to make a tradeoff in the time domain response. Choose between the following:

- **Minimum Window = Better Response Resolution** - the ability resolve between two closely spaced responses.
- **Maximum Window = Dynamic Range** - the ability to measure low-level responses.



[Learn how to launch the Transform dialog box](#)

### Programming Commands



## Transform - Window dialog box help

**Coupling Settings** Launches the [Setup Trace Coupling](#) dialog box.

The window settings balance response resolution versus dynamic range.

- Minimum Window = Best Response Resolution
- Maximum Window = Best Dynamic Range

The following three methods all the set window size. For best results, view the time domain response while making these settings.

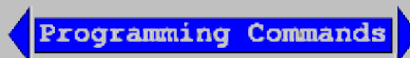
- **Minimum - Maximum** Move the slider with a mouse to change the window size
- **Kaiser Beta** Changes window size using a Kaiser Beta value
- **Impulse Width** Changes window size using an Impulse Width value

Learn more about [Windowing](#) (top)

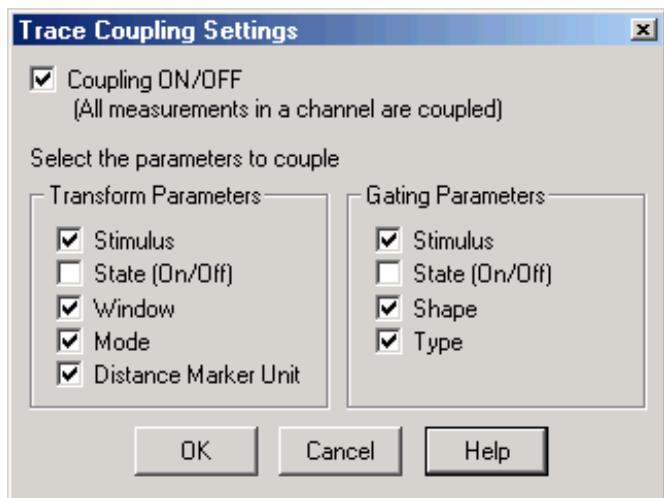
## How to make Trace Coupling Settings

You can launch the **Trace Coupling Settings** dialog box from any of the following dialog boxes:

- [Transform](#)
- [Gating](#)
- [Window](#)







### Trace Coupling Settings dialog box help

Trace coupling allows you to change time domain parameters on a measurement, and have the same changes occur for all other measurements in the channel.

For example:

If you are simultaneously viewing a frequency domain measurement and time domain measurement, and **Coupling** is enabled in this dialog box, and ALL **Gating Parameters** are checked in this dialog box, and on the time domain measurement you change the **Gate Span** parameter,

Then the frequency domain measurement will automatically change to reflect the time domain gated span.

**Coupling ON/OFF** Check to enable coupling. All of the measurements in the active channel are coupled.

The following parameters are available for coupling:

#### Transform Parameters

**Stimulus** Start, Stop, Center, and Span TIME settings.

**State** (On/Off) Transform ON and OFF

[Window](#) Kaiser Beta / Impulse Width

[Mode](#) Low Pass Impulse, Low Pass Step, Band Pass

#### Gating Parameters

**Stimulus** Start, Stop, Center, and Span TIME settings.

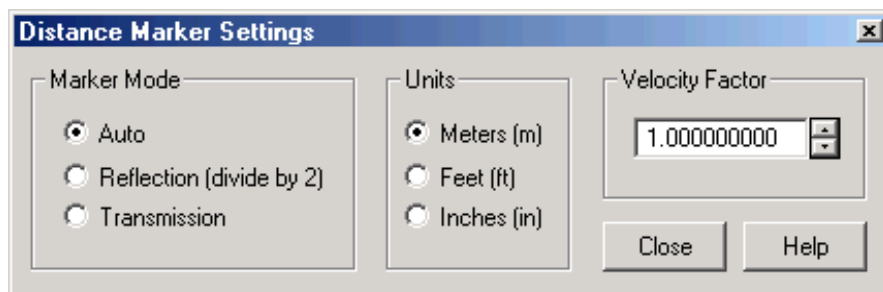
**State** (On/Off) Gating ON and OFF

[Shape](#) Minimum, Normal, Wide, and Maximum

[Type](#) Bandpass and Notch

To launch the Distance marker dialog box, click **Dist. Marker Settings** on the [Transform](#) dialog box.

## Programming Commands



### Distance Marker Settings dialog box help

When markers are present on a time domain measurement, distance is automatically displayed on the marker readout, [marker table](#), and [print copy](#). To learn how to create markers on your measurement see [marker settings](#).

This dialog box allows you to customize the time domain distance marker readings.

These settings affect the display of ALL markers for only the ACTIVE measurement (unless **Distance Marker Unit** is coupled on the [Trace Coupling dialog box](#).

**Marker Mode** Specifies the measurement type in order to determine the correct marker distance.

- Select **Auto** for [S-Parameter](#) measurements.
- Select **Reflection** or **Transmission** for [arbitrary ratio](#) or [unratioed](#) measurements.

**Auto** If the active measurement is an S-Parameter, automatically chooses reflection or transmission. If the active measurement is a non S-Parameter, reflection is chosen.

**Reflection** Displays the distance from the source to the receiver and back divided by two (to compensate for the return trip.)

**Transmission** Displays the distance from the source to the receiver.

**Units** Specifies the unit of measure for the display of marker distance values.

**Velocity Factor** Specifies the velocity factor that applies to the medium of the device that was inserted after the measurement calibration. The value for a polyethylene dielectric cable is 0.66 and 0.7 for Teflon dielectric. 1.0 corresponds to the speed of light in a vacuum. This is useful in Time Domain for accurate display of time and distance markers.

This setting can also be made from the [Electrical Delay](#) and [Port Extensions](#) dialog boxes.

Last Modified:

3-Sep-2008    Removed legacy content

11-Sep-2007    Edits to resolution and rage formulas

## Integrated Pulse Measurements

---

Beginning with PNA Rev. A.09.20, the Pulse Setup dialogs shown in this topic are integrated in the PNA-X firmware and are available with Opt 008 or Opt H08.

Previously, setup was performed with the [Narrowband](#) or [Wideband](#) pulse programs (E836x models can continue to use these programs) and with [SCPI and COM commands](#). With the appropriate hardware options (Opt 021, 022, 025) these commands are still available without Opt 008 or H08.

In this topic

- [Pulse Setup](#)
- [Pulse Generator Setup](#)
- [Pulse Trigger Tab](#)
- [Pulse Gens and IF Block Diagram](#)
- [Calibration in Pulse](#)

### See Also (separate topics)

- [IF Path Configuration](#)
- [Programming commands](#)
- [Narrowband Pulsed Application](#) (Opt H08)
- [WB Pulsed App](#)
- [IF Access User Interface Settings](#) (PNA "C" models)

### How to start the Pulse Setup dialog

To provide quicker access, use the Setup softkey. [Learn how](#).

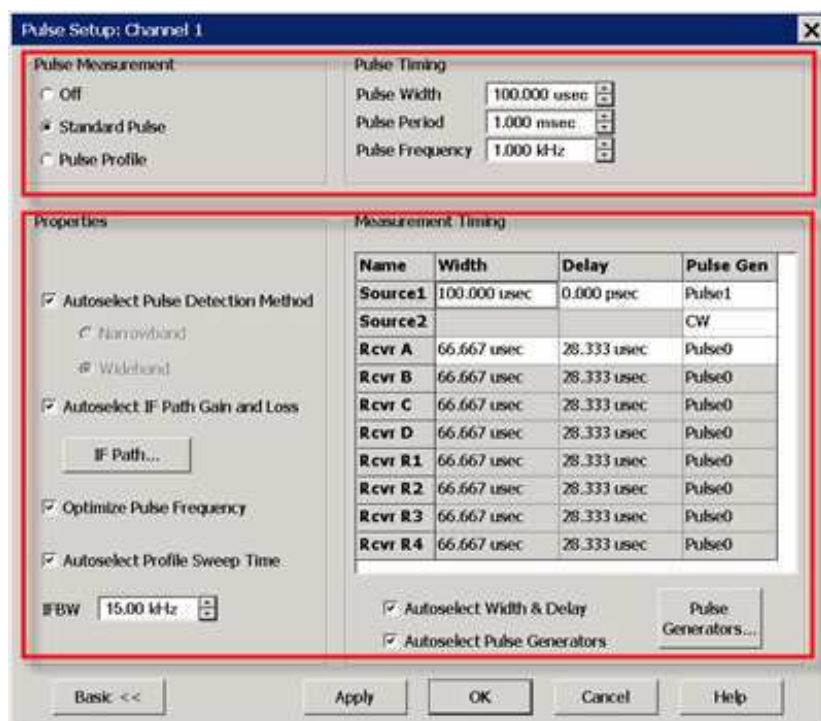
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SWEEP**
2. then **[Pulse Setup]**

#### Using a mouse with PNA-X Menus

1. Click **Stimulus**
2. then **Sweep**
3. then **Pulse Setup**

[Programming Commands](#)



## Pulse Setup dialog box help

The Basic (upper box) controls allow simple pulse measurements using the default (Autoselect) settings in the Advanced section (lower box) of the dialog.

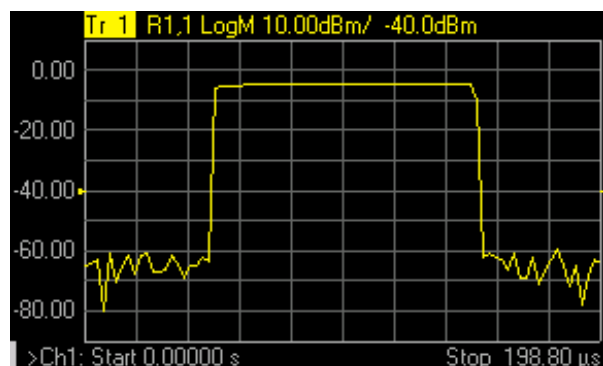
Pulsed measurements are performed in a Standard channel. [See Measurement Class](#). However, several PNA measurement settings are controlled by the Pulse setup, such as sweep type, number of points, and so forth.

## Pulse Measurement

**Off** - Source and Receivers are NOT pulsed

**Standard Pulse** - With pulsed RF, the PNA can be configured to sweep in frequency, power sweep, and CW time. Narrow the receiver pulse width and enter delay to make 'point-in-pulse' measurements.

**Pulse Profile** - Pulse profile measurements provides a time domain (CW frequency) view of the pulse envelope. Profiling is performed using a measurement technique that "walks" a narrow receiver "snapshot" across the width of the pulse. This is analogous to using a camera to take many small snapshots of a wide image, then piecing them together to form a single, panoramic view.



*Pulse Profile measurement using default settings and R1 receiver.*

- Pulse Profiling can be performed using ratioed or unratioed measurements. You can preview the pulse on port 1 by using an R1 receiver measurement.
- Pulse Profiling is performed at a single CW frequency in either Narrowband or Wideband mode.
- To select the CW Frequency, click **Stimulus**, then **Sweep Type**.
- In Narrowband mode, the delay increment value, which is responsible for "walking" the receiver snapshot across the pulse, is selected by the PNA and is accessible only with a [programming command](#).
- In Wideband mode, the receiver is walked across the pulse by making a sequence of closely-spaced measurements in real-time.

### Pulse Timing

**Pulse Width** - Sets the width of the source pulse. See [measurement timing](#) to learn how to control the receiver width and delay.

**Pulse Period** The time to make one complete pulse.

**Pulse Frequency (PRF)** The reciprocal of Period (1/ Period). See [Internal Pulse Generators](#) to learn more.

By default, these settings configure Pulse Gen 1 to drive Source Modulators 1 and 2. This can be changed from the Advanced Settings [Pulse Generator Setup](#) dialog.

### ----- Advanced Settings -----

The following settings allow maximum control of a Pulse measurement.

**Note:** When the "Auto" check boxes are cleared, it is possible to configure settings to make an invalid measurement.

### Properties

**Autoselect pulse detection method** - check to automatically switch between Narrowband and Wideband based on the Pulse Width.

In Standard Pulse:

- **Wideband** - used when the (source) Pulse Width is WIDER than the fastest receiver acquisition time. This allows the receiver to measure all pulse ON time - no pulse OFF time. The PNA will select Wideband whenever possible.
- **Narrowband** - used when the (source) Pulse Width is NARROWER than the fastest receiver acquisition time (267 ns). This measurement requires a spectral nulling technique to measure the pulse response through the DUT.

In Pulse Profile:

- **Wideband** - used when the (source) Pulse Width is greater than 1.600 us. This allows the receiver make several sequential measurements to measure the entire pulse.
- **Narrowband** - used when the (source) Pulse Width is less than or equal to 1.600 us.

**Autoselect IF Path Gain and Loss** - For future use.

**IF Path** - Click to launch the [IF Path dialog](#).

**Optimize Pulse Frequency** - Automatically selects the Pulse Frequency and Pulse Period.

- In Narrowband, the pulse frequency is adjusted slightly to get the best spectral-nulling filtering possible.
- In Wideband, this checkbox is ignored.

**Autoselect Profile Sweep Time** - In Pulse Profile mode, adjusts the default X-axis start time to zero and the stop time double the Pulse Width. This allows you to see one complete pulse. If unchecked, the Sweep Time will not be changed.

To adjust the X-axis manually, click **OK** to close the dialog. Then click **Stimulus**, then **Sweep**, then **Sweep Time**, then change the **Start Time** and **Stop Time**.

**IFBW** - Select the IFBW for the measurement.

- In Narrowband, an IFBW as close as possible to the entered value will be used.
- In Wideband, this setting determines the receiver acquisition time - approximately 1/ IFBW.

## Measurement Timing

**Source1** and **Source2** - Used as RF Source Modulation Drive.

- **Width** - source pulse width.
- **Delay** - source pulse delay relative to the pulse generator clock.
- **Pulse Gen** - Pulse generator used to modulate the source. Select **CW** to have NO source modulation.

The receiver settings in this table change depending on whether the PNA is in Narrowband or Wideband mode.

- In Narrowband, for each IF receiver path, configure the Pulse Width, Delay, and Pulse Generator to be used to drive the receiver gate.
- In Wideband, all receiver paths are the same.

**Autoselect Width and Delay** - When checked, for Wideband mode and Pulse0, the receiver is adjusted to approximately 75% of the source pulse width, with 20% delay. This leaves approximately 5% of the source pulse ON after acquisition is complete.

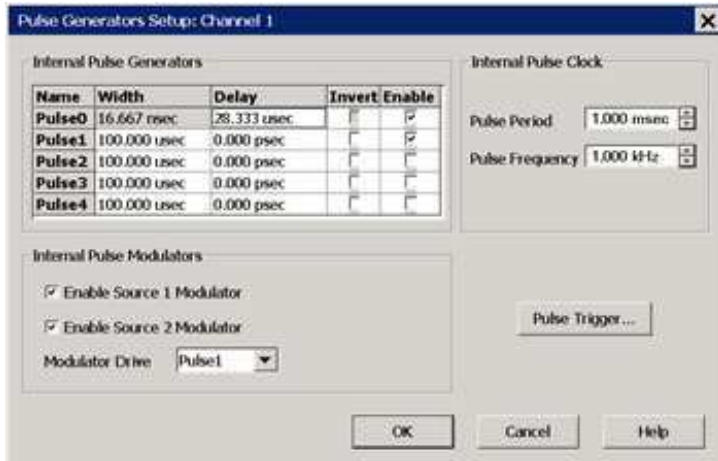
When checked for Narrowband mode and Pulse Gen = CW, then the delay and width have no values. When a Pulse Gen is selected, then Delay and Width matches the RF Source.

**Autoselect Pulse Generators** - When checked:

- Pulse1 is selected for Modulator Drive.
- Pulse2 is CW (OFF).

- For Wideband, Pulse0 is selected to gate the ADC.
- For Narrowband, Pulse2 is selected.

**Pulse Generators** Click to launch the [Pulse Generators Setup](#) dialog.



### Pulse Generators Setup dialog box help

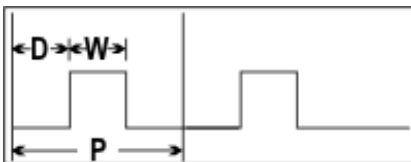
This dialog is available with Option 025 (pulse generators).

To see this dialog, press **Pulse Gen Setup** on the [Pulse Setup dialog](#).

#### Internal Pulse Generators

For each Pulse Generator configure the width, delay, and polarity.

Pulse0 triggers the PNA receivers when Enabled. The Width can NOT be configured. Delay defines the delay between the pulse clock and when the receiver begins to make a measurement. [Learn more.](#)



- **D** = Delay; the time before each pulse begins
- **W** = Width; the time the pulse is ON
- Duty Cycle =  $W/P$

#### Internal Pulse Clock (P)

- **P** = Period; one complete pulse cycle
- Pulse Frequency (PRF) =  $1 / \text{Period}$

**Important:** If **D + W** is greater than **P**, then undefined PNA behavior results. There is NO error message or warning.

**Invert** Check to cause the pulse ON time to be active low and OFF be active high.

[Learn more about the Pulse Generators.](#)

### Internal Pulse Modulators

Check to enable one or both internal source modulators.

These are switches 8 and 9 in the [Block Diagram](#)

**Important:** When internally modulating the sources, [source leveling is automatically set to Open-loop](#).

**Modulator Drive** Choose the pulse generator to modulate the specified source. Choose from **CW** (NO pulse), Pulse 1, 2, 3, 4, External. This is switch 7 [Block Diagram](#).

**Pulse Trigger..** click to start the [Pulse Trigger dialog](#).



### Pulse Trigger Tab - Trigger dialog box help

To see this dialog, press **Pulse Trigger** on the [Pulse Generator Setup dialog](#) or select **Stimulus**, then **Trigger** from the PNA Menu.

#### Pulse Sync Trigger - Source

Select **Internal** or **External** to provide sync capability for the internal pulse generators.

- **Internal** - The pulse generator is internally triggered and puts out a periodic pulse train with a period defined by the [Pulse Generator Setup dialog](#).
- **External** - The period is ignored and the internal pulse generator puts out one set of pulses (P0-P4) per external trigger. All five pulse outputs have unique delay and pulse width settings. External trigger input is on the [Pulse I/O connector](#) pin 7 (PulseSyncIn). The PulseSyncIn line provides a configurable trigger signal into the Pulse Generators. If a level trigger is still valid when the first pulse set is finished, another set will be generated. Only one set of pulses is emitted when edge triggering is used. The length of time that it takes to emit one set of pulses is the end time of the last enabled pulse (largest of width + delay of



all the pulses P0-P4).

### Level/Edge

Sets the edge or level of the trigger signal to which the internal pulse generators will respond when being externally triggered at the PulseSynIn pin.

Positive = rising edge; Negative = falling edge.

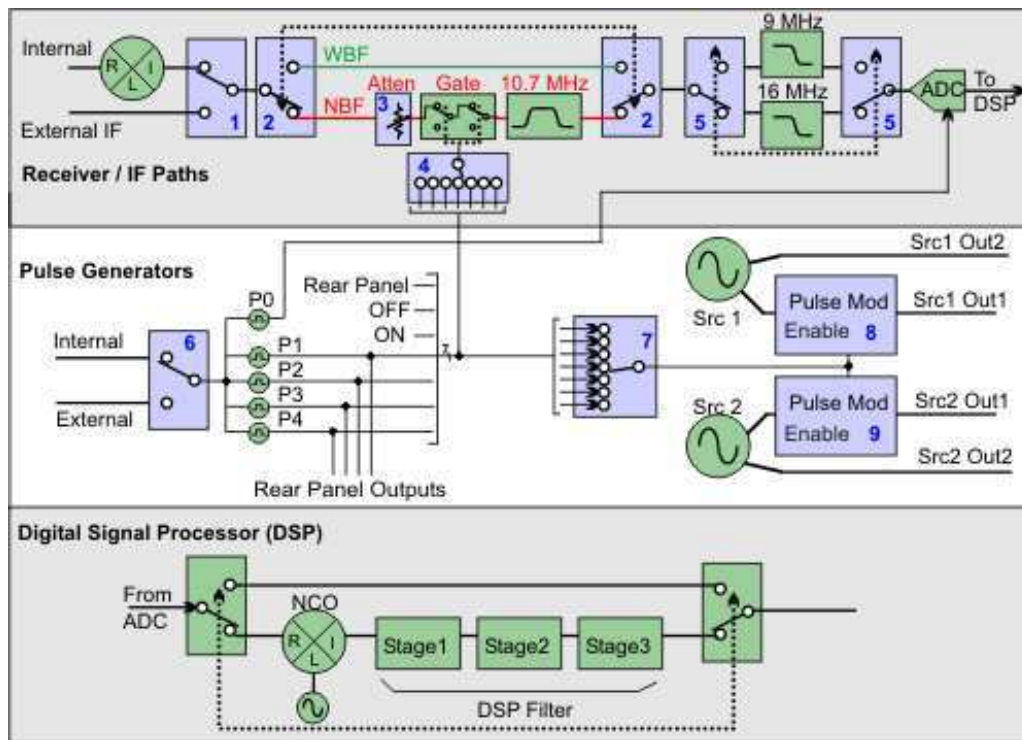
These selections are available ONLY with DSP version: 4.0 FPGA: 34 or higher. [Learn more](#). Otherwise, the pulse generators respond only to positive, level input trigger signals.

### Receiver

**Synchronize receiver to pulse generator Pulse0** - Check to enable Pulse Gen 0 (P0) which is used to gate the ADC for wideband receiver measurements. [Learn more about Pulse0](#).

**Receiver delay** - Set the amount of time to wait before triggering the ADC to begin acquisition.

### Pulse Trigger Block Diagram



[See complete description at IF Path Configuration](#)

### Calibration in Pulse Mode

To perform a calibration in pulse mode (option H08 / 008), first configure and apply the pulse parameters (PRF, Pulse Width, Delays, IF gating, and so forth) **before** calibrating the system. This will ensure the PNA is configured properly during the calibration and measurement.

---

Last Modified:

11-Apr-2010    MX New topic

## Narrowband Pulsed Application

---

**Note:** The PNA-X with Option 025 has an integrated [Pulse Measurement Setup](#) (Opt 008) that makes this application no longer necessary on the PNA-X.

The Narrowband Pulsed Application is a Visual Basic program that provides a user interface for making pulsed measurements.

In this topic:

- [Required Options](#)
- [Connecting External Pulse Generators](#)
- [Using the Narrowband Pulsed Application](#)
- [How to Configure Pulse Generators and Receivers](#)
- [Calibration in Pulse Mode](#)
- [Pulse Profiling](#)
- [Signal Reduction versus Gate Width](#)
- [Pulsed Frequency Converter Measurements](#)
- [Writing your own Narrowband Pulsed Application](#)

The following enhancements were made In PNA Rev. 7.2:

- [Enhanced Pulse Measurement Capabilities](#)
- [Support for Internal Pulse Generators / Modulators \(PNA-X only\)](#)

### See Also

- Learn about the [Wideband Pulsed Application](#).
- For more conceptual information see our [Pulsed Measurement App Notes](#).
- [See PNA-X Block Diagram of IF Path / Pulse Generators / Source Modulation](#)
- [Programming commands](#)
- [Programming examples](#)

### Other IF Access Topics

## Required Options and Equipment

The PNA H08 option provides the Narrowband Pulsed Application. The following options are also required. If your PNA does not have the required options, a message is displayed on the screen. For more information, see [Pulsed-RF Measurements Configuration Guide](#)

### PNA-X Models

- PNA-X models: None; however [Opts 021, 022, and 025](#) greatly enhance speed, performance, and convenience.
- PNA-X Opt 224 (2-port Dual Source) select the **2-port Dual Source** Configuration in the [Path Configuration](#) dialog to provide Pulse modulation on both port 1 and port 2.
- Use the PNA-X [Pulse I/O connector](#) to access the internal pulse generators
- See the PNA-X [IF Path Configuration block diagram](#) which includes the Pulse Modulators and Generators.

### Other PNA Models

- E836x models: [Opt 014 \(front panel access\)](#) and [Opt 080 \(frequency offset\)](#). To use the internal receiver gating feature of the Narrowband Pulsed Application, your PNA must have the [H11 hardware option](#).
- PNA-L models: H08 NOT available
- Agilent 81104A or 81110A Pulse Generator with ONLY the 81105A or 81111A output modules. The 81112A module does NOT have selectable 50 ohm/1K ohm output impedance/load compensation to drive the 1K ohm PNA IF gates. For more information, see the 81100 Family of Pulse Pattern Generators Technical Specifications at: <http://cp.literature.agilent.com/litweb/pdf/5980-1215E.pdf>

### Connecting External Pulse Generators

Each 81110A Pulse Generator has two output modules. Each output can drive a PNA IF Receiver or Source Modulation (Z5623A H81). Connect the Pulse Generators as follows:

#### 81110A front panel connectors



- Connect GPIB cables to the 81110A and PNA.
- Connect the PNA 10 MHz Ref Out to the 81110A 10 MHz IN.
- If using two 81110As for a total of 4 outputs, then connect the TRIGGER OUT of one to the EXT INPUT of the other 81110A.
- Connect the 81110A OUTPUTs to the PNA rear panel IF inputs to be gated. The outputs are mapped in the [Pulsed Generator Configuration](#) dialog box.

**Connect the Z5623A H81Pulse Test Set (optional) to the PNA front-panel port 1 loops as follows:**

PNA	H81
Src Out	Source IN
CPLR THRU	CPLR THRU
RCVR R1 IN	RCVR R1 Out

### See Also

- [PNA Front-panel loops](#)
- [PNA-X rear-panel](#)
- [PNA IF connectors](#)
- [81110A Documentation](#)
- [Z5623A H81 Documentation](#)

## Using the Narrowband Pulsed Application

### How to start the Narrowband Pulsed Application

#### Using front-panel HARDKEY [softkey] buttons

1. Press **MACRO**
2. then **[Pulse]**

#### Using a mouse with PNA Menus

1. Click **Utility**
2. then **Macro**
3. then **Pulse**

◀ **Programming Commands** ▶

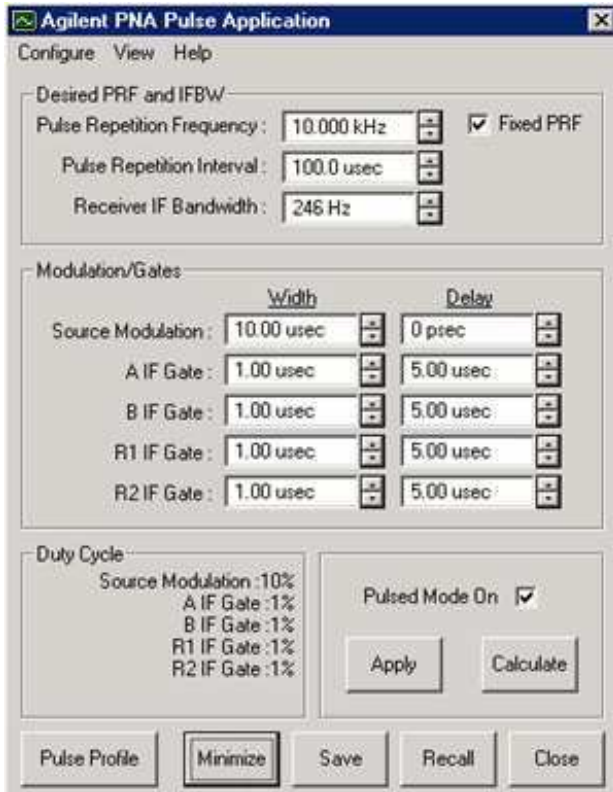
### See Also

- [See programming commands to launch the Macro remotely.](#)
- [See how to write your own custom Narrowband Pulsed Application.](#)

### Keypad Data Entry

The PNA front-panel Numeric Entry and Navigation keys can be used for dialog box input. Also, a keyboard can be used to enter values, including alpha characters for prefixes (for example, **u** for usec.) . After typing values, first press **Enter**, then press **Tab** to go to the next field.

The following is an image of the main dialog box:



### Pulsed Application Main dialog box help

**Note:** An **error message** may appear on the PNA stating that the response frequency has exceeded the maximum allowed frequency.

The Narrowband Pulsed Application may set the offset frequency ([option 080](#)) of the PNA to some value other than zero (the default value). If the stop frequency is set to the maximum of the PNA model, then the error message will appear.

To fix this, set the stop frequency to a value that is at least 2 KHz less than the maximum allowed. For example, if you have a 20 GHz PNA, and the stop frequency is set to 20 GHz, and the error message appears, then set the stop frequency to 19.999998 GHz

[See Block Diagram of IF Path / Pulse Generators / Source Modulation](#)

#### Configure

You can configure more than one channel to make pulsed measurements, but the channels must use the same [pulse generator settings](#).

Only the Agilent 81110A Pulse Generator is supported with the Narrowband Pulsed Application. Refer to the 81110A documentation for pulse repetition frequency and duty cycle capabilities.

#### See Also

- [Configure Receivers](#)

- [Converter Measurements](#)

**Edit / Undo** Pulse Application settings revert to those when Apply was last pressed.

**Desired PRF and IFBW** Enter the DESIRED values. When **Calculate** is pressed, one or both of these values may change.

**Pulse Repetition Frequency (PRF):** Frequency of the pulses from the Pulse Generator.

**Pulse Repetition Interval:**  $1/\text{PRF}$  Changes to either PRF or this setting changes both.

**Receiver IF Bandwidth:** IF Bandwidth of the PNA. Choose a setting from 1 Hz to 10 KHz.

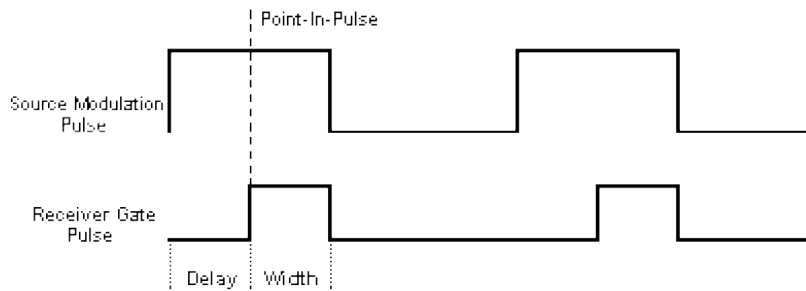
**Fixed PRF** When checked, (default setting) the **Calculate** algorithm will NOT adjust the PRF, but only change the IF Bandwidth.

**Modulation/Gates** The Source Modulation and four PNA receiver gates can each have their own, or share, Pulse Generator outputs. Shared outputs have identical Width and Delay values. To configure and enable outputs, click **Configure**, then **Pulse Generators** to launch the [Pulsed Generator Configuration](#) dialog box.

**Note:** Option 036 and 037 limits the Source Modulation width to 117 ns.

**Width** Pulse Width.

**Delay** The delay that occurs before the pulse.



**Duty Cycle** Calculated Duty Cycle of the source and each of the selected receivers. Updated when **Calculate** is pressed.

**Pulse Mode On** When this box is checked, the PNA is enabled for Pulsed measurements. The PNA [Status Bar](#) annotation indicates the following:

- **G** Internal IF gates enabled.
- **F** Filtering for Pulsed Measurements enabled.

**Apply** All selections are sent to the pulse generator and the active channel of the PNA.

**Calculate** All selections are calculated and valid PRF and IFBW values are entered in their fields. If these settings are not acceptable, try changing the values you previously entered and click Calculate again. When acceptable values are attained, click **Apply** to send these values to the pulse generator and PNA.

**Pulse Profile** Launches the Pulse Profile dialog box. Same as clicking **View / Pulse Profile**. If not available, check **Pulse Mode ON**, click **Calculate**, then **Apply**.

**Minimize** Click to minimize the dialog box to make changes in the PNA application. To see the dialog again, select **Macro, Pulse**, or turn the [Status Bar](#) ON.

**Save** All settings from the Narrowband Pulsed Application are saved in a \*.ppf file. These settings are NOT

saved with PNA instrument state.

**Recall** Restore settings from the specified \*.ppf file that were previously saved.

**Close** Closes the dialog box without saving changes.

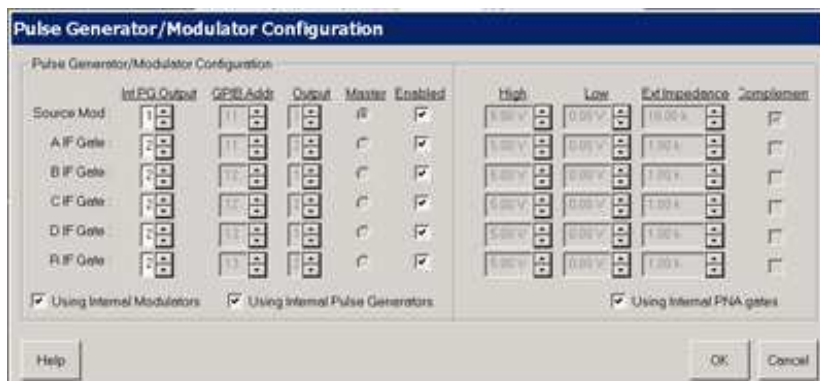
## How to configure Pulse Generators / Modulators and Receivers

From the [Pulse App main dialog box](#)



Learn about...

- [Configure Receiver Gain](#)
- [Converter Measurements](#)
- **No Pulse Generators** When checked, the Narrowband Pulsed Application does NOT attempt to communicate with internal or external pulse generators. This setting is used for troubleshooting purposes.
- **No SW Gating** When checked, the improved SW gating sensitivity is turned OFF. This setting is used for troubleshooting purposes.





## Pulsed Generator Configuration dialog box help

### Notes:

- [See Block Diagram of PNA-X IF Path / Pulse Generators / Source Modulation](#)
- For PNA-X Opt 224 (2-port Dual Source) select the **2-port Dual Source** Configuration in the [Path Configuration](#) dialog to provide Pulse Modulation on both port 1 and port 2.
- This dialog may look different depending on the PNA model and number of receivers available.

Configures either the internal pulse generators (PNA-X models with relevant options), or Agilent 81110A Pulse Generator outputs. You can configure each 81110A Pulse Generator with either one or two 81111A output modules.

The Source Mod and four PNA receiver gates can each have their own, or shared, pulsed generators allowing identical **Width** and **Delay** values which are selected on the Main dialog.

To share an external generator output between one or more PNA inputs, use the same GPIB address and output module for each PNA input.

**Internal Pulse Gen Output** (available ONLY on the [PNA-X opt 025](#))

Specify the Pulse Gen (1 through 4) to use to modulate each of the PNA receiver IF gates or Sources.

### External Pulse Generator settings

**GPIB Addr:** The GPIB address of the 81110A.

**Output:** The output module of the 81110A.

**Master:** The 81110A that uses the 10 MHz reference signal from the PNA.

**Enabled:** Turns the pulse output ON.

### External Gate/Modulator settings

**High:** Specify a 'TTL-High' voltage level

**Low:** Specify a 'TTL-Low' voltage level

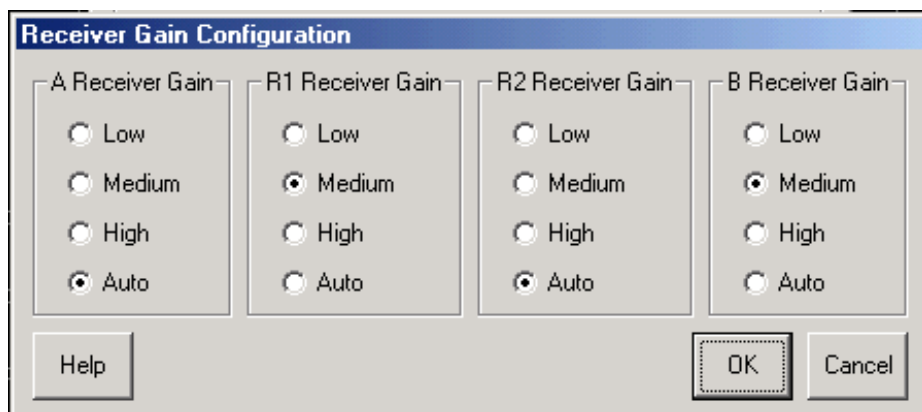
**Ext Impedance:** Impedance of the modulator used to create the pulse.

**Complement:** When this box is cleared, TTL HIGH is the pulse. When checked, TTL LOW is the pulse.

**Using Internal Modulators** When this box is checked, the voltage, impedance, and complement values are forced to settings that prevent damage to the internal modulator.

**Using Internal Pulse Generators** Makes the appropriate settings on this dialog available.

**Using Internal PNA gates** When this box is checked, the voltage, impedance, and complement values are forced to settings that prevent damage to the internal gates.



The image shows a 'Receiver Gain Configuration' dialog box with a blue title bar. It contains four panels, each for a different receiver: 'A Receiver Gain', 'R1 Receiver Gain', 'R2 Receiver Gain', and 'B Receiver Gain'. Each panel has four radio button options: 'Low', 'Medium', 'High', and 'Auto'. In the 'R1' and 'R2' panels, the 'Auto' option is selected. In the 'A' and 'B' panels, the 'Medium' option is selected. At the bottom of the dialog, there are three buttons: 'Help', 'OK', and 'Cancel'.

### Receiver Gain Configuration dialog box help

[See Block Diagram of PNA-X IF Path / Pulse Generators / Source Modulation](#)

This dialog may look different depending on the PNA model and number of receivers available.

Sets the gain of each PNA receiver manually or automatically.

**Auto** - The PNA selects the best gain level to make pulsed measurements.

Use the following to manually set the gain for each receiver.

**Low** - about 0 dB of gain

**Medium** - about 17 dB of gain

**High** - about 24 dB of gain

The **PNA-X** has the following attenuation settings:

**Low** - 30 dB of attenuation

**Medium** - 15 dB of attenuation

**Hi** - 0 dB of attenuation

## Calibration in Pulse Mode

To perform a calibration in pulse mode (option H08), first configure and apply the pulse parameters (PRF, Pulse Width, Delays, IF gating, and so forth) **before** calibrating the system. This will ensure the PNA is configured properly during the calibration and measurement.

When performing [Unknown Thru](#) or [TRL calibrations](#), ALL receivers must be gated. Otherwise, the error terms will not be correct after the calibration has completed. This can be accomplished by either having a separate pulse generator output for each of the IF gates, or by connecting pairs of the IF gates together with BNC-T's. For example, if the pulse generator does not have enough outputs, then connect the R1 and R2 IF gates to the same pulse generator output. Also, connect the A and B IF gates to either separate outputs (recommended) or one output (reduces flexibility). The error terms will then be valid after the calibration is complete.

## Pulse Profiling

Pulse profiling provides a time domain view of the pulse envelope. Profiling is performed using a measurement technique that "walks" a narrow receiver "snapshot" across the width of the pulse. This is analogous to using a camera to take many small snapshots of a wide image, then piecing them together to form a single, panoramic

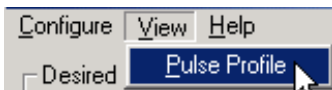
view.

- Pulse Profiling can be performed using ratioed or unratioed measurements.
- Pulse Profiling is performed at a single CW frequency.

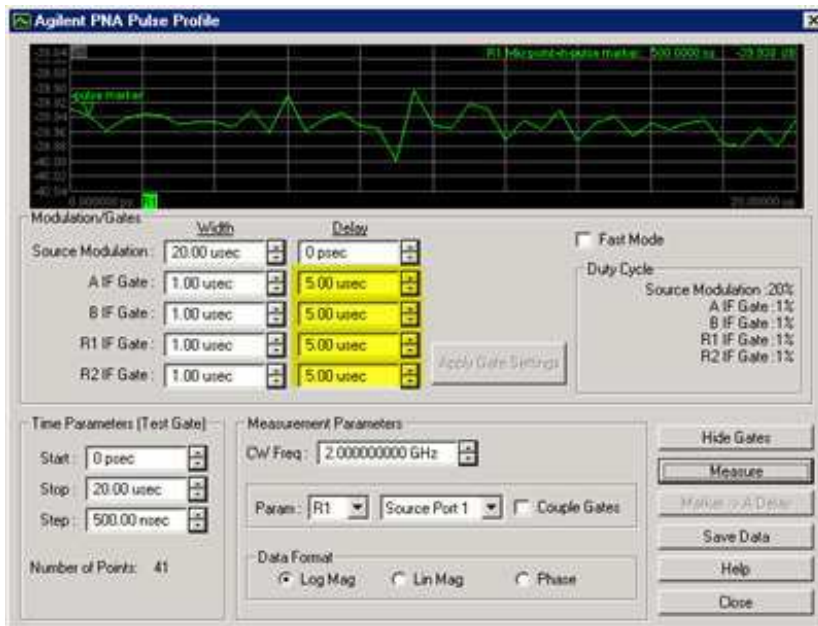
### How to perform Pulse Profiling

From the [Pulse App main dialog box](#),

Click the **Pulse Profile** button. or:



If this setting is unavailable, check **Pulse Mode ON**, click **Calculate**, then **Apply**.



### Pulse Profile dialog box help

[Learn about Pulse Profiling \(scroll up\)](#)

[See Block Diagram of PNA-X IF Path / Pulse Generators / Source Modulation](#)

#### Modulation / Gates

These settings duplicate those found on the main [Pulse App dialog box](#).

In Pulse Profile, the Gate Delay settings (highlighted in yellow) are significant only with certain **Measurement Parameter** and **Couple Gates** settings.

#### Time Parameters

**Start, Stop** These two combine to make the window of the assembled pulse profile. To view the entire pulse, the start and stop values must be at least as wide as the Source Modulation **Width** plus **Delay** value.

**Step** Each consecutive snapshot is incremented by this value until the stop value is reached. Therefore, the number of points for the pulse profile measurement can be calculated as:  $(\text{Stop} - \text{Start}) / \text{Step}$ . The higher the number of points, the longer it takes to make the measurement.

### Measurement Parameter

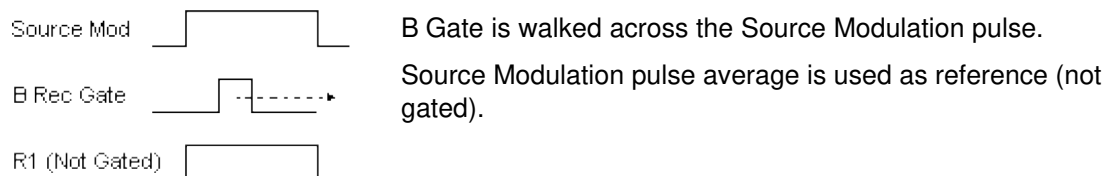
**CW Freq.** Frequency of the PNA source.

**Source Port** The PNA port supplying the source power. Only required for single receiver (unratioed) measurements.

**Param(eter)** Only those receiver gates (and relevant measurements) that are configured in [Pulsed Generator Configuration](#) are available.

**Note:** When a single receiver (unratioed) is selected, **Gate Delay** Settings (highlighted in yellow on above dialog image) are ignored.

If the reference receiver gate is NOT configured, the average of the Source Modulation pulse is used as the reference. **For example:** With **S21** Selected, but ONLY **B** receiver gate is configured, then...

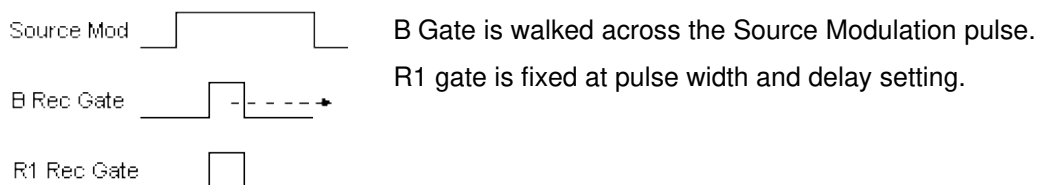


**Coupled Gates** Used when the appropriate receiver gates are [configured](#) for your **S-parameter** measurement ONLY. This setting is ignored when a single receiver (**Param**) is selected.

- **Uncoupled** (box cleared) The reference gate is FIXED at the delay setting as the test gate is walked across the Source Modulation pulse as dictated by the **Time Parameter** settings.

**For example:**

**S21** Selected, **B** and **R1** receiver gates configured, Gates **Uncoupled**



- **Coupled** (box checked) The reference gate is walked synchronously with the test gate as dictated by the **Time Parameter** settings. Only the **difference** between the test and reference gate delay values is significant; NOT the absolute values.

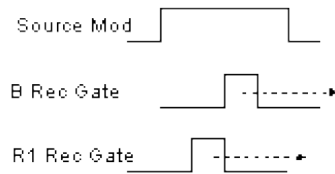
**For example:**

**S21** Selected, **B** and **R1** receiver gates configured, Gates **Coupled**

**B** gate delay = 3 microseconds,

**R1** gate delay = 2 microseconds

Difference = 1 microsecond



B Gate is walked across the Source Modulation pulse.

R1 gate is fixed at pulse width and delay setting.

B gate leads R1 gate by 1 microsecond.

**Data Format** Log Magnitude, Linear Magnitude, or Phase (only available if S-parameter selected).

### Buttons

**Show Gates** Allows you to change the receiver gating width and delay while looking at the results.

**Apply Gate Settings** Click after making changes to gate settings.

**Continuous Sweep** Check, then click **Measure**, to continuously measure pulse profiling.

**Measure** Click to start the pulse profile measurement. Becomes **Stop** when continuously sweeping.

**Marker to Delay** After making a measurement, you can drag the display maker to any point along the trace. Click this button and the marker time is entered into the Receiver **Delay** field on the [main dialog box](#).

**Save Data** Saves time domain data to the PNA hard drive in any of the following formats:

- Touchstone (\*.s1p)
- Comma delimited (\*.prn)
- Citifile (\*.cti)

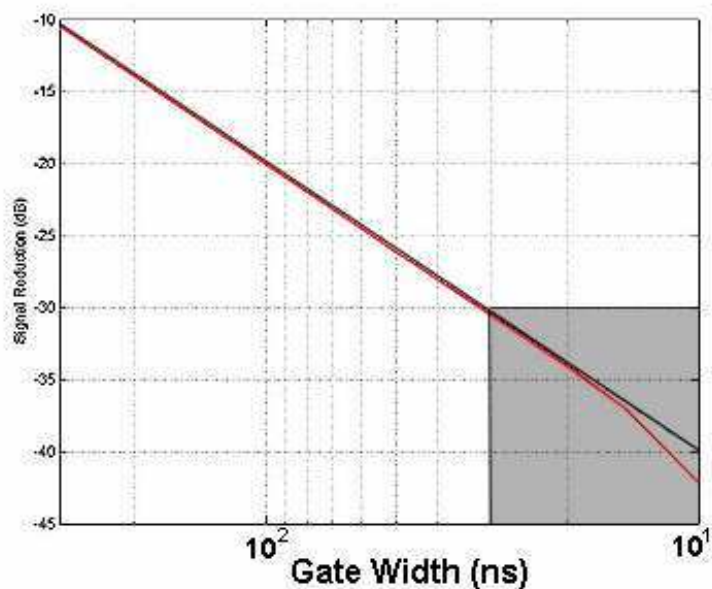
[Learn more about these data formats](#)

## Signal Reduction versus Gate Width

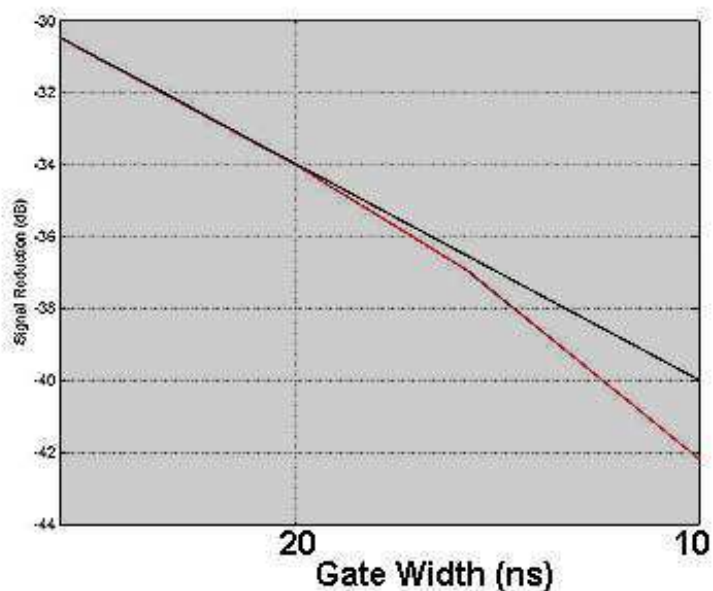
### Signal Reduction versus Gate Width

PRF = 1 MHz

The following two figures show the performance of the internal IF gates as the width is narrowed.



The following is a zoomed image of the shaded area (above).



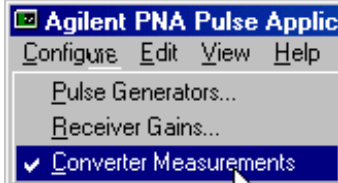
- The straight line shows the theoretical loss in dynamic range due to duty cycle effects when using narrowband detection.
- The curved (red) line shows the actual measured performance of the gates.
- The minimum gate width for <1dB deviation from theoretical is approximately 20ns.

See the specifications for the [option H11](#) and [option H08](#).

## Pulsed Frequency Converter Measurements

The Narrowband Pulsed Application works with both [FCA \(option 083\)](#) and standard [Frequency Offset \(opt 080\)](#) measurements. On the **Configure** menu, check **Converter Measurements**. When checked, this setting prevents the Narrowband Pulsed Application from [overwriting frequency offset values](#). This may limit the number of **PRF** and **IFBW** solutions that are returned when **Calculate** is pressed on the main Pulsed Application dialog box.

**Note:** Pulse Profiling can NOT be performed with frequency converter measurements.



## Writing your own Narrowband Pulsed Application

You can use the Narrowband Pulsed Application or use an example program as a template for making your own Narrowband Pulsed Application.

The Narrowband Pulsed Application uses a custom .dll to perform the calculations that are necessary to make pulsed measurements. Use the COM Method below to send and return values to **agilentpnapulsed.dll**. Then use SCPI or COM commands to control the PNA.

	E836xC	PNA-X
COM Example Program	<a href="#">E836x Create</a>	<a href="#">PNA-X Create</a>
SCPI Example Programs	None	<a href="#">Point-in-Pulse</a> <a href="#">Pulse Profile</a>
COM Methods	<a href="#">ConfigNarrowBand3</a>	<a href="#">ConfigEnhancedNB2</a> <a href="#">ConfigEnhancedNBIFAtten</a>
SCPI commands	<a href="#">SCPI</a>	<a href="#">SCPI</a>
COM commands	<a href="#">COM</a>	<a href="#">COM</a>

## Install and Register the Pulsed .dll on your PC

To create your own Narrowband Pulsed Application, or run the Narrowband Pulsed Application from a remote PC, you must do the following:

1. Copy the following files from the PNA C:/program files/agilent/network analyzer/ to a directory on your PC.
  - **agilentpnapulsed.dll**
  - **OffsetList.txt**
  - **prfbw.txt**

- **prfbwmixer.txt**

2. To register the ActiveX DLL in Microsoft Windows Operating System:

- From a command prompt on your PC, navigate to the directory where you copied the DLL.
- Type: **regsvr32 agilentpnapulsed.dll** and press **Enter**

For Operating Systems other than Windows, see their associated help files to learn how to register DLL files.

---

Last Modified:

29-Apr-2009    Added Opt 036 and 037 note (A.08.60)

4-Sep-2008    Removed legacy content

2-Jul-2008    Added links to SCPI examples



## Wideband Pulsed Application

---

**Note:** The PNA-X with Opt 025 and Opt 008 has an integrated [Pulse Measurement Setup](#) that makes this application no longer necessary on the PNA-X.

The Wideband Pulsed Application configures the PNA-X internal pulse generators and modulators for measuring pulsed S-parameters using the wideband mode detection technique.

The Wideband Pulse Application is designed to be used with the PNA-X with Opts 021, 022, and 025.

**Note:** Wideband Pulse application is NOT supported on the E836x and PNA-L models.

### See Also

- To learn more about wideband detection, see [Application Note 1408-12](#).
- See a Visual Basic example: [Create a Wideband Pulsed Measurement using the PNA-X](#)
- Learn about the [Narrowband Pulsed Application](#).

## Download and Install the Wideband Pulsed Application

This application is installed and run as a macro on the PNA-X. [Learn more about macros.](#)

1. Go to <http://na.tm.agilent.com/pna/apps/applications.htm>
2. Click the download link
3. Save the downloaded file to the PNA hard drive
4. Double-click the downloaded file to install the Wideband Pulsed Application on the PNA.
5. The application is saved on the PNA at C:/Program Files/Agilent/Network Analyzer/Applications/WB Pulse/Wideband\_pulse.exe
6. Configure the macro. [Learn how.](#)

To learn more about Wideband pulsed application, click Help in the application.

---

### Last Modified:

17-Mar-2010    Added PNA-X note

22-Jun-2007    MX New topic

## Connecting the PNA to a PC

---

This document describes how to temporarily connect a PC to a PNA using a LAN cable. This is not necessary if your PNA is already connected to a network. This type of connection is for conveniently transferring large files, such as firmware, that may have already been downloaded and stored on the PC.

The PC can have any version of Windows (Windows 95 or newer.)

You will need the following:

- RJ-45 LAN crossover cable (or two normal cables with a suitable hub).
- Must be logged on the PNA with an [Administrator User name](#) and password.

**Note:** If your PC is on a domain, do not leave that domain by changing to a workgroup. This may prevent you from later rejoining your domain unless you involve your IT systems administrator. The following procedure will work regardless of whether or not you are on a domain and will not change any domain settings on the PC.

### Procedure for All Operating Systems

1. Disconnect the PC from any existing LAN and connect it to the PNA using a crossover cable or hub. There is no need to turn it off to do this.
2. Find the current IP address of your PC. Open a DOS prompt (command window) and type **ipconfig**. This should then show your current IP address. Note this number. For this example, we will assume a PC IP address of 10.0.0.100. If it shows 0.0.0.0, you will have to assign an IP address. [See procedure below.](#)
3. On the PNA, click **File**, then **Exit** to close the PNA application.
4. Right-click on **My Network Places** and select **Properties**.
5. Right-click on **Local Area Connection** and select **Properties**.
6. Select **Internet Protocol (TCP/IP)** then click **Properties**.
7. Click **Use the following IP address** then enter an IP address that is ONE more or ONE less than that noted for the PC. Do not use a number that ends in 000. For this example, one could use either 10.0.0.101 or 10.0.0.99. We'll assume the use of 101.
8. For **Subnet mask**, enter 255.255.255.0. Click **OK**, then **OK** again

For Windows 95/98 systems, go to [Win95/98 step 9](#) below.

9. On the PC, open Windows Explorer. Click **Tools**, then **Map Network Drive**. Note the drive letter shown (for this example we'll assume it is "X".)
10. In **Folder or Path** type the IP address that was just entered for the PNA followed by C\$ in the following format: //10.0.0.101/C\$
11. In **Connect As** dialog, enter the PNA User name and Password. Unless it has been changed, this is either Administrator with NO password or PNA-Admin, Password Agilent. Click **OK**.

12. In Windows Explorer on the PC, you should now see a new drive letter entry (X) with the description of C\$ on 10.0.0.101. The entire contents of the PNA C drive will now be available for reading or writing. Files can now be transferred by simply dragging them from one location to the other.
13. When you are done transferring files, disconnect the LAN cable connection between the two and reconnect your PC to the network if needed. If the PNA will never be connected to any network, the current network settings could remain. However it may be safer to reset the TCP/IP settings (changed in step 7) back to **Obtain an IP address automatically**.

#### **If your PC currently has no IP address assigned**

Follow steps 4 through 8 to add an IP address for the PC. The actual steps may vary slightly depending upon your operating system (NT4 uses Network Neighborhood and you must click on Protocols.) Assign the PC an IP address of 10.0.0.100. The last digits can be anything between 1 and 255, but do not alter the first 3 numbers (10.0.0.) When complete, reset the PC network configuration back to its previous settings.

#### **Win95/98 Procedure for Accessing/Transferring Files**

Because Windows 95/98 does not have the security features of NT-based operating systems (NT, Win2k, XP), the PC cannot access the drive on the PNA. To get around this limitation, any files that need to be transferred to/from the PC must be placed in a shared folder. The PNA can then read or write within this folder. This procedure will work for all versions of Windows.

Steps 1 through 8 are identical to the above and must be performed first.

9. On the PC, open Windows Explorer. Create a directory under the C drive named **Shared**. Right click the Shared folder name and select **Sharing**. Share the directory with full read/write permissions. If Sharing does not appear as a choice, then file sharing is not enabled. To enable file sharing,
  1. Right click **Network Neighborhood**, then click **Properties**.
  2. Click **File and Print Sharing** and enable **give others access to my files**
  3. Click **OK**, then **OK** again.
  4. Repeat the beginning of this step.
10. Copy the files to be transferred to the PNA to this shared folder.
11. On the PNA, open Windows Explorer. Click on Tools, Map Network Drive. Note the drive letter shown (for this example we'll assume it is "X".) Uncheck "Reconnect at Logon" if it is currently checked.
9. Under the Folder entry, enter the IP address of the PC and the shared folder name in the following format:  
//10.0.0.100/Shared
10. The PNA should immediately connect to this folder and display its contents as drive "X". Files can now be read from, or written to, this shared directory (shown as Drive X.) Files can be transferred by simply dragging them from one location to the other.
11. When you are done transferring files, disconnect the LAN cable connection between the two and reconnect your PC to the network if needed. If the PNA will never be connected to any network, its current network settings could remain, however it is probably safer to reset the TCP/IP settings (changed in step 7) back to

**Obtain an IP address automatically.**

## Easy versus Secure Configuration

---

When upgrading Firmware on the PNA, you encounter a **Choose Configuration** dialog box. This is used to determine the level of security set for the DCOM interface on the PNA. For more detailed information on the security settings for the DCOM interface, including a procedure for making these settings manually, see [Configure for COM-DCOM Programming](#).

Comparison of the "Easy and More Secure" settings are as follows:

### Easy Connection:

- No configuration of the PNA required for remote access to connect.
- Anyone on the local subnet can access the PNA remotely.
- People from other NT domains can connect to the PNA.

### More Secure:

- Requires creating users on the PNA or adding the PNA to a domain
- An administrator of the PNA can specify users or groups that are allowed remote access to the PNA application

## Changing Network Client

---

If your PC network uses Novell NetWare servers, a change must be made to the PNA setup before it can operate on your network. If you are unsure, ask your local IT department.

**Note:** Do NOT **Uninstall** "Client for Microsoft Networks". This will prevent proper operation of the PNA..

To remove "Client for Microsoft Networks" (Remove is different from Uninstall):

1. From the PNA Desktop, right-click **My Network Places**
2. Click **Properties**
3. Right-click **Local Area Connection**
4. Click **Properties**
5. Click (remove the check from) **Client for Microsoft Networks**

To install "Client Service for NetWare".

1. Click **Install**
2. In **Select Network Component Type**, make sure **Client** is selected
3. Click **Add**
4. In **Select Network Client**, make sure **Client Service for NetWare** is selected
5. Click **OK**.

## Troubleshooting the PNA

---

By running a few checks, you can identify if the analyzer is at fault. Before calling Agilent Technologies or returning the instrument for service, please make the following checks.

- [Check the Basics](#)
- [PNA Application Terminates Unexpectedly](#)
- [Check Error Terms](#)
- [Check the Service Guide](#)

### Other Support Topics

## Check the Basics

A problem can often be solved by repeating the procedure you were following when the problem occurred. Before calling Agilent Technologies or returning the instrument for service, please make the following checks:

**Note:** Problems with the PNA application (slow or terminates unexpectedly) can be caused by a faulty Hard Disk Drive (HDD). For more information, see [Preventing PNA Hard Drive Problems](#) and [The PNA HDD Recovery Process](#).

1. Is there power at the power socket? Is the instrument plugged in?
2. Is the instrument turned on? Check to see if the front panel line switch and at least one of the LED rings around the test ports glows green. This indicates the power supply is on.
3. If you are experiencing difficulty with the front-panel keypad or peripherals, the USB bus may be overloaded. Remove the USB devices, restart the PNA, and reconnect the USB devices. See [Power-up](#).
4. If other equipment, cables, and connectors are being used with the instrument, make sure they are connected properly and operating correctly.
5. Review the procedure for the measurement being performed when the problem appeared. Are all the settings correct?
6. If the instrument is not functioning as expected, return the unit to a known state by pressing the **Preset** key.
7. Is the measurement being performed, and the results that are expected, within the [specifications](#) and capabilities of the instrument?
8. If the problem is thought to be due to firmware, check to see if the instrument has the [latest firmware](#) before starting the troubleshooting procedure.
9. Check that the measurement calibration is valid. See [Accurate Measurement Calibrations](#) for more information.
10. If the necessary test equipment is available, perform the operator's check and system verification in Chapter

2 of the PNA Service Guide, "System Tests, Verifications, and Adjustments," . You can download a copy of the Service Guide from our Web site: <http://www.agilent.com>.

11. **Phase lock lost message** - This usually occurs when there is not enough source power to phase lock the PNA. It can occur during an errant [FCA setup](#) or [Source Power Calibration](#). It can also occur if one of the front panel reference channel loops is not connected. Otherwise, this indicates a hardware problem.

## PNA Application Terminates Unexpectedly

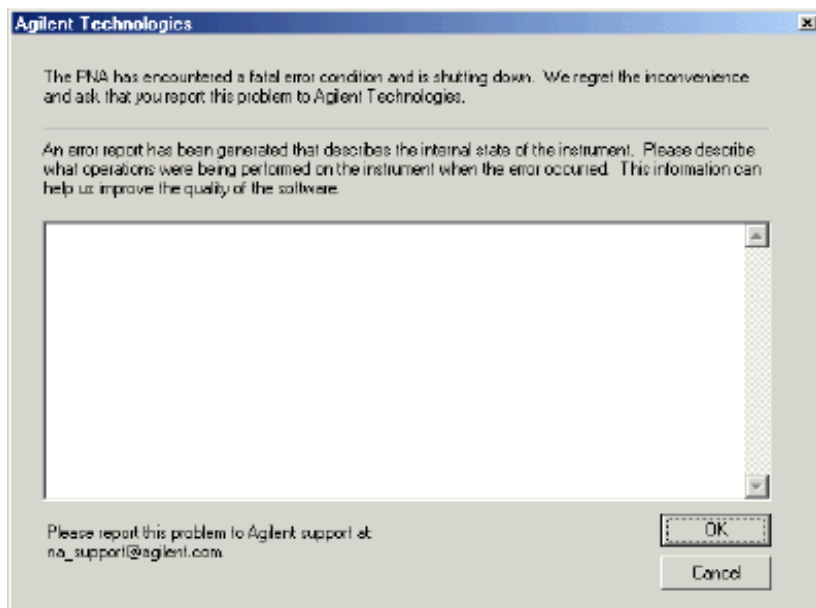
If an unexpected and irrecoverable error occurs, Agilent would like to know about it. The PNA attempts to save pertinent information about the state of the system. **The PNA does NOT send this information to Agilent.**

We respect the privacy of our customers. However, access to information that helps us improve the PNA is a benefit to both Agilent and you. Please take the time to contact us or email the saved information to **na\_support@agilent.com**.

The following procedure shows how to do this:

1. A message box immediately appears on the screen containing the location of a directory. Please record this message. If you miss the message, you can find the directory location using the Windows Event Log: On the PNA, click Start, Settings, Control Panel, Administrative Tools, Event Viewer. Double-click the top line (most recent event). The location of the directory is seen in the Description.
2. A dialog box may appear on the screen (shown below) allowing you to add comments to help us replicate the crash.
3. Find the directory (described in Step 1) which contains the following files:
  - 835x.dmp which is the 835x.exe capturing the context in which the program crashed.
  - 835x.xml which reports some very basic information ( exception code, OS version, and the list of modules loaded at the time of the crash and their respective version numbers).
  - 835xCrashLog.txt: The text file with your comments (described in Step 2), if submitted.
4. If your PNA is not connected to LAN or is not configured to send email, copy the files to a PC. Then, please email the files to [na\\_support@agilent.com](mailto:na_support@agilent.com)





## Check Error Terms

If you print the error terms at set intervals (weekly, monthly, and so forth), you can compare current error terms to these records. A stable, repeatable system should generate repeatable error terms over long time intervals, for example, six months. If a subtle failure or mild performance problem is suspected, the magnitude of the error terms should be compared against values generated previously with the same instrument and calibration kit. See the [procedure for monitoring error terms](#).

- A long-term trend often reflects drift, connector and cable wear, or gradual degradation, indicating the need for further investigation and preventative maintenance. Yet, the system may still conform to specifications. The cure is often as simple as cleaning and gaging connectors or inspecting cables.
- A sudden shift in error terms reflects a sudden shift in systematic errors, and may indicate the need for further troubleshooting.

Consider the following while troubleshooting:

- All parts of the system, including cables and calibration devices, can contribute to systematic errors and impact the error terms.
- Connectors must be clean and gauged, and within specification for error term analysis to be meaningful. See the Chapter 2 in the PNA Service Guide for information on cleaning and gaging connectors.
  - Avoid unnecessary bending and flexing of the cables following measurement calibration, thus minimizing cable instability errors.
  - Use good connection techniques during the measurement calibration. The connector interface must be repeatable. See the PNA Service Guide for information on connection techniques.
- It is often worthwhile to perform the procedure twice (using two distinct measurement calibrations) to establish the degree of repeatability. If the results do not seem repeatable, check all connectors and cables.

- Use error-term analysis to troubleshoot minor, subtle performance problems. See Chapter 3, "Troubleshooting," in the PNA Service Guide if a blatant failure or gross measurement error is evident.

## Check the Service Guide

Check the PNA Service Guide for specific troubleshooting procedures to help identify problems. You can download a copy of the Service Guide from our Web site: <http://www.agilent.com>.

---

Last modified:

10/16/06    Added phase lock lost

## PNA Error Messages

---

- [500 - 750 Calibrate](#)
- [770 - 1000 Hardware](#)
- [1000 - 1200 Measure](#)
- [1281 - 1535 Parser](#)
- [1536 - 1650 Display](#)
- [1700 - 2000 Channel](#)
- [2048 - 2200 General](#)
- [Standard SCPI Errors](#)

**Note:** The **EventID**'s listed below are provided for COM programming. For more information, see [Working with PNA Events](#)

For more information on PNA error messages [\(see Error Messages\)](#).

### Memory Overflow Error

Memory overflow. Trigger state set to Hold. Lower the IF bandwidth, or increase dwell or sweep time.

**Severity:** Informational

**Further explanation:** The measurement that you are currently making requires that data be stored faster than it can be processed. Very few customers will experience this situation.

**Suggestions:** To limit the amount of data to be stored, try lowering the IF Bandwidth, slow the sweep time, increase the dwell time, or limit the number of data points. There are many other settings that can be adjusted to solve this problem.

**EventID:**

---

### Cal Errors

---

#### Message: 512

"A secondary parameter (power, IFBW, sweep time, step mode) of the calibrated state has changed."

**Severity:** Informational

**Further explanation:** The calibration is questionable when any of these secondary parameters change after the calibration is performed.

**Suggestions:** If you require an accurate measurement with the new settings, repeat the calibration.

**EventID:** 68020200 (hex)

#### Message: 513

"Calibration cannot be completed until you have measured all the necessary standards for your selected Cal Type."

**Severity:** Informational

**Further explanation:** You probably received this message because you attempted to turn correction on without first measuring all of the calibration standards

**Suggestions:** Finish measuring the cal standards

**EventID:** 68020201 (hex)

**Message: 514**

"Calibration set has been recalled using a file previously saved on an analyzer that had a different hardware configuration."

**Severity:** Informational

**Further explanation:**

**Suggestions:**

**EventID:** 68020202 (hex)

**Message: 515**

"Calibration is required before correction can be turned on. Channel number is <x>, Measurement is <x>."

**Severity:** Informational

**Further explanation:** There are no error correction terms to apply for the specified channel and measurement.

**Suggestions:** Perform or recall a calibration

**EventID:** 68020203 (hex)

**Message: 516**

"Critical parameters in your current instrument state do not match the parameters for the calibration set, therefore correction has been turned off. The critical instrument state parameters are sweep type, start frequency, frequency span, and number of points."

**Severity:** Informational

**Further explanation:** None

**Suggestions:** You can either recalibrate using the new settings or change back to the original setting that was used when the calibration was performed.

**EventID:** 68020204 (hex)

**Message: 517**

"Interpolation is turned off and you have changed the stimulus settings of the original calibration, so correction has been turned off."

**Severity:** Informational

**Further explanation:** The most accurate calibration is maintained only when the original stimulus settings are used.

**Suggestions:** If reduced accuracy is OK, set interpolation ON to allow stimulus setting changes.

**EventID:** 68020205 (hex)

**Message: 518**

"Interpolation is turned off and you have selected correction ON. Correction has been restored with the previous stimulus settings."

**Severity:** Informational

**Further explanation:** None

**Suggestions:** None

**EventID:** 68020206 (hex)

#### Message: 519

"Stimulus settings for your current instrument state exceeded the parameters of the original calibration, so correction has been turned off."

**Severity:** Informational

**Further explanation:** Correction data outside the stimulus settings does not exist.

**Suggestions:** Perform a broadband calibration, with increased numbers of points with interpolation ON, to maintain calibration over the widest possible stimulus frequency settings.

**EventID:** 68020207(hex)

#### Message: 520

"Cal Type is set to NONE for Channel <x>, Measurement <x>; please select Calibration menu or press Cal hard key."

**Severity:** Informational

**Further explanation:** A cal operation can not proceed until a calibration exists or the cal type is selected. This error can occur if the calibration can not be found. Also this error can happen if a calibration type is not specified before attempting to programmatically execute cal acquisitions.

**Suggestions** To find a calibration, select a Cal Set that contains the calibration needed for the current measurements. OR specify the cal type before beginning a calibration procedure.

**EventID:** 68020208 (hex)

#### Message: 521

"The measurement you set up does not have a corresponding calibration type, so correction has been turned off or is not permitted."

**Severity:** Informational

**Further explanation:** The calibration for the channel may apply only to certain S-Parameters. For example, a 1-Port calibration for S11 can not be applied to a 1-Port calibration applied to S22.

**Suggestions:** Select a calibration type, such as full 2-Port cal, that can be applied to all the measurements to be selected.

**EventID:** 68020209 (hex)

#### Message: 522

"The calibration type you selected cannot be set up."

**Severity:** Informational

**Further explanation:** "Please use the SCPI command ROUTe:PATH:DEFine:PORT <num>,<num> for full 2 port type port assignment."

**Suggestions:**

**EventID:** 6802020A (hex)

**Message: 523**

"The calibration path you selected cannot be set up because it is not valid for the current measurement."

**Severity:** Informational

**Further explanation:** "Please use the SCPI command ROUTe:PATH:DEFine:PORT <num>,<num> for full 2 port type port assignment related to your current measurement."

**Suggestions:**

**EventID:** 6802020B (hex)

**Message: 524**

"The source power calibration is complete."

**Severity:** Informational

**Further explanation:**

**Suggestions:**

**EventID:** 6802020C (hex)

**Message: 525**

"You have specified more than 7 standards for one or more calibration classes."

**Severity:** Informational

**Further explanation:** These have been truncated to 7 selections.

**EventID:** 6802020D (hex)

**Message: 526**

"No user calibration found for this channel."

**Severity:** Informational

**Further explanation:** A cal operation can not proceed until a calibration exists.

**Suggestions:** To find a calibration, you can select a Cal Set that contains the calibration needed for the current measurement.

**EventID:** 6802020E (hex)

**Message: 527**

"You do not need to acquire this standard for this calibration type."

**Severity:** Informational

**Further explanation:** This error can happen as a result of PROGRAMMATICALLY requesting the measurement of an un-needed calibration standard during a calibration procedure.

**Suggestions:** Check the specified cal type or eliminate the request for the measurement of the standard.

**EventID:** 6802020F (hex)

**Message: 528**

"Could not configure the Electronic Calibration system. Check to see if the module is plugged into the proper connector."

**Severity:** Informational

**Further explanation:** During an ECal operation, communication could not be established with the ECal module. The calibration will not be initiated until the presence of the ECal module is verified.

**Suggestions:** Verify the USB cable is connected properly. Disconnect and re-connect the cable to ensure the analyzer recognizes the module.

**EventID:** 68020210 (hex)

**Message: 529**

"DATA OUT OF RANGE: Design Limits Exceeded"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020211(hex)

**Message: 530**

"EXECUTION ERROR: Could not open ECal module memory backup file"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020212 (hex)

**Message: 531**

"EXECUTION ERROR: Access to ECal module memory backup file was denied"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020213 (hex)

**Message: 532**

"EXECUTION ERROR: Failure in writing to ECal module memory backup file"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020214 (hex)

**Message: 533**

"EXECUTION ERROR: Failure in reading from ECal module memory backup file"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020215 (hex)

**Message: 534**

"EXECUTION ERROR: Array index out of range"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020216 (hex)

**Message: 535**

"EXECUTION ERROR: Arrays wrong rank"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020217 (hex)

**Message: 536**

"EXECUTION ERROR: CPU"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020218 (hex)

**Message: 537**

"EXECUTION ERROR: Cannot ERASE module"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020219 (hex)

**Message: 538**

"EXECUTION ERROR: Cannot WRITE module"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E802021A (hex)



**Message: 539**

"EXECUTION ERROR: Entry Not Found"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E802021B (hex)

**Message: 540**

"EXECUTION ERROR: Invalid command while system is busy"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E802021C (hex)

**Message: 541**

"Electronic Cal: Unable to orient ECal module. Please ensure the module is connected to the necessary measurement ports."

**Severity:** Error

**Further explanation:** There is no RF connection to the ECal module during a calibration step. An ECal orientation measurement has been attempted but the signal was not found.

**Suggestions:** Connect the ECal module RF connections to ports specified for the calibration step. The ECal module typically requires at least -18dBm for measurements. If your measurement requires the power level to be less than that, clear the **Do orientation** checkbox to bypass the automatic detection step.

**EventID:** E802021D (hex)

**Message: 542**

"EXECUTION ERROR: NO SPACE for NEW CAL, DELETE A CAL"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E802021E (hex)

**Message: 543**

"EXECUTION ERROR: No More Room"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E802021F (hex)

**Message: 544**

"EXECUTION ERROR: Other array error"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020220 (hex)

**Message: 545**

"EXECUTION ERROR: Ranks not equal"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020221 (hex)

**Message: 546**

"EXECUTION ERROR: Too few CONSTANT ranks"

**Severity:** Error

**EventID:** E8020222 (hex)

**Message: 547**

"EXECUTION ERROR: Too few VARYing ranks"

**Severity:** Error

**EventID:** E8020223 (hex)

**Message: 548**

"EXECUTION ERROR: Unknown error"

**Severity:** Error

**EventID:** E8020224 (hex)

**Message: 549**

"EXECUTION ERROR: ecaldrvr.dll bug or invalid module #"

**Severity:** Error

**EventID:** E8020225 (hex)

**Message: 550**

"EXECUTION ERROR: unexpected error code from ecal driver"

**Severity:** Error

**EventID:** E8020226 (hex)

**Message: 551**

"EXECUTION ERROR: unexpected internal driver error"

**Severity:** Error

**EventID:** E8020227 (hex)

**Message: 552**

"HARDWARE ERROR: Can't access ECal Interface Module"

**Severity:** Error

**EventID:** E8020228 (hex)

**Message: 553**

"HARDWARE ERROR: Can't release LPT port, reboot"

**Severity:** Error

**EventID:** E8020229 (hex)

**Message: 554**

"HARDWARE ERROR: VNA Error"

**Severity:** Error

**EventID:** E802022A (hex)

**Message: 555**

"HARDWARE ERROR: not enough data read from ECal module"

**Severity:** Error

**EventID:** E802022B (hex)

**Message: 556**

"OPERATION ABORTED BY HOST COMPUTER"

**Severity:** Error

**EventID:** E802022C (hex)

**Message: 557**

"OPERATION ABORTED BY USER"

**Severity:** Error

**EventID:** E802022D (hex)

**Message: 558**

"OUT OF MEMORY"

**Severity:** Error

**EventID:** E802022E (hex)

**Message: 559**

"QUERY INTERRUPTED:Message(s Abandoned"

**Severity:** Error

**EventID:** E802022F (hex)

**Message: 560**

"QUERY UNTERMINATED: INCOMPLETE PROGRAM Message"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020230 (hex)

**Message: 561**

"QUERY UNTERMINATED: NOTHING TO SAY"

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020231 (hex)

**Message: 562**

"QUEUE OVERFLOW"

**Severity:** Error

**EventID:** E8020232 (hex)

**Message: 563**

"SETTINGS CONFLICT: ADDITIONAL STANDARDS ARE NEEDED"

**Severity:** Error

**EventID:** E8020233 (hex)

**Message: 564**

"SETTINGS CONFLICT: Adapter Cal is NOT possible"

**Severity:** Error

**EventID:** E8020234 (hex)

**Message: 565**

"SETTINGS CONFLICT: COMMAND OUT OF SEQUENCE"

**Severity:** Error

**EventID:** E8020235 (hex)

**Message: 566**

"SETTINGS CONFLICT: Cal STOPPED - VNA SETUP CHANGED"

**Severity:** Error

**EventID:** E8020236 (hex)

**Message: 567**

"SETTINGS CONFLICT: Calibration is NOT in progress"

**Severity:** Error

**EventID:** E8020237 (hex)

**Message: 568**

"SETTINGS CONFLICT: Can't find specified GPIB board"

**Severity:** Error

**EventID:** E8020238 (hex)

**Message: 569**

"SETTINGS CONFLICT: Can't find/load gpib32.dll"

**Severity:** Error

**EventID:** E8020239 (hex)

**Message: 570**

"SETTINGS CONFLICT: Can't find/load sicl32.dll"

**Severity:** Error

**EventID:** E802023A (hex)

**Message: 571**

"SETTINGS CONFLICT: Can't initialize VNA (bad address?)"

**Severity:** Error

**EventID:** E802023B (hex)

**Message: 572**

"SETTINGS CONFLICT: Can't load LPT port driver or USB driver DLL"

**Severity:** Error

**EventID:** E802023C (hex)

**Message: 573**

"SETTINGS CONFLICT: Invalid Calibration Sweep Mode."

**Severity:** Error

**EventID:** E802023D (hex)

**Message: 574**

"SETTINGS CONFLICT: Invalid Calibration Type"

**Severity:** Error

**EventID:** E802023E (hex)

**Message: 575**

"SETTINGS CONFLICT: Invalid Calibration"

**Severity:** Error

**EventID:** E802023F (hex)

**Message: 576**

"SETTINGS CONFLICT: Invalid GPIB board number specified"

**Severity:** Error

**EventID:** E8020240 (hex)

**Message: 577**

"SETTINGS CONFLICT: Invalid GPIB board type specified"

**Severity:** Error

**EventID:** E8020241 (hex)

**Message: 578**

"SETTINGS CONFLICT: Invalid Module Status"

**Severity:** Error

**EventID:** E8020242 (hex)

**Message: 579**

"SETTINGS CONFLICT: Invalid States"

**Severity:** Error

**EventID:** E8020243 (hex)

**Message: 580**

"SETTINGS CONFLICT: LPT port must be between 1 and 4"

**Severity:** Error

**EventID:** E8020244 (hex)

**Message: 581**

"Could not configure the Electronic Calibration system. Check to see if the module is properly connected."

**Severity:** Error

**EventID:** E8020245 (hex)

**Message: 582**

"SETTINGS CONFLICT: Specified LPT port does not exist"

**Severity:** Error

**EventID:** E8020246 (hex)

**Message: 583**

"SETTINGS CONFLICT: Use frequency domain for cal"

**Severity:** Error

**EventID:** E8020247 (hex)

**Message: 584**

"SETTINGS CONFLICT: Use step sweep type for cal."

**Severity:** Error

**EventID:** E8020248 (hex)

**Message: 585**

"SETTINGS CONFLICT: VNA address must be between 0 and 30"

**Severity:** Error

**EventID:** E8020249 (hex)

**Message: 586**

"SETTINGS CONFLICT: Wrong LPT port driver or USB driver DLL"

**Severity:** Error

**EventID:** E802024A (hex)

**Message: 587**

"SYNTAX ERROR: ECAL:DELAY command must have 2 numbers"

**Severity:** Error

**EventID:** E802024B (hex)

**Message: 588**

"SYNTAX ERROR: INCORRECT SYNTAX"

**Severity:** Error

**EventID:** E802024C (hex)

**Message: 589**

"SYNTAX ERROR: UNKNOWN COMMAND"

**Severity:** Error

**EventID:** E802024D (hex)

**Message: 590**

"Wrong port of module in RF path"

**Severity:** Error

**EventID:** E802024E (hex)

**Message: 591**

"User characterization not found in module"

**Severity:** Error

**EventID:** E802024F (hex)

**Message: 592**

**Severity:** Informational

"No source power calibration found for the channel and source port of the current measurement."

**Further explanation:** You tried to turn on source power cal but there is no source power cal data.

**Suggestions:** Perform a source power calibration

**EventID:** 68020250 (hex)

**Message: 593**

**Severity:** Informational

"A source power calibration sweep was not performed, so there is no correction for the channel and source port of the current measurement."

**Further explanation:** You tried to turn on source power cal but there is incomplete source cal data.

**Suggestions:** Perform a complete source power calibration

**EventID:** 68020251 (hex)

**Message: 594**

**Severity:** Informational

"A new trace could not be added to the active window for viewing the source power cal sweep, because it would have exceeded the limit on number of traces/window. Please remove a trace from the window before proceeding with source power cal."

**Further explanation:** The source power cal attempts to add a data trace to the active window. The active window already contains four traces.

**Suggestions:** Make the active window contain less than four traces.

**EventID:** 68020252 (hex)

**Message: 595**

**Severity:** Informational

"A new measurement could not be added for performing the source power cal sweep, because the limit on number of measurements has been reached. Please remove a measurement before proceeding with source power cal."

**Further explanation:** The source power cal attempts to add a measurement. The PNA already has the maximum number of measurements.

**Suggestions:** Delete a measurement.

**EventID:** 68020253 (hex)



#### Message: 596

**Severity:** Informational

"The calibration power value associated with the source power calibration of Port %1 on Channel %2 was changed with the calibration on. The calibration was not turned off, but the power value might no longer represent the calibration."

**Further explanation:** The source power cal accuracy is questionable.

**Suggestions:** If high accuracy is required, perform another source power calibration.

**EventID:** 68020254 (hex)

#### Message: 597

**Severity:** Informational

- Message that is passed from the power meter driver for a source power calibration. -

**Further explanation:** This error is generated by the power meter driver and passed through the PNA.

**EventID:** 68020255 (hex)

#### Message: 598

"During the acquisition of the sliding load standard, the slide was not properly moved to perform a circle fit. The standard's raw impedance was used to determine the directivity for one or more points."

**Severity:** Informational

**Further Explanation:** To accurately characterize the standard, the sliding load must be move sufficiently to ensure enough samples around the complex circle or Smith Chart. Under-sampling will cause an inaccurate result.

**Suggestions:** For best results when using a sliding load, be sure to use multiple slide positions that cover the full range of movement from front to back of the slot.

**EventID:** 68020256 (hex)

#### Message: 599

"This feature requires an unused channel, but could not find one. Please free up a channel and try again."

**Severity:** Informational

**Further Explanation:** You attempted to view an item within a calset. However, the calset viewer requires that the result be displayed in a channel that is not currently in use. All the channels are currently used. The view can not display the requested item.

**Suggestions:** You must delete at least one channel that is currently in use.

**EventID:** 68020257 (hex)

#### Message: 600

"Interpolation of the original calibration is not allowed since it was performed using Segment Sweep. Correction has been turned off."

**Severity:** Informational

**EventID:** 68020258 (hex)

#### Message: 601

"Cal preferences saved. Cal preference settings can be changed from the 'Cal Preferences' drop down Cal menu."

**Severity:** Informational

**Further explanation:** [See Save Preference](#)

**EventID:** 68020259 (hex)

#### Message: 608

"CalType not set."

**Severity:** Error

**Further explanation:** A cal operation can not proceed until a calibration exists or the proper cal type is selected.

**Suggestions:** This error can happen if the calibration can't be found. To find a calibration, you can select a Cal Set that contains the calibration needed for the current measurements. This error can also happen if a calibration type is not specified before attempting to programmatically execute cal acquisitions. Specify the cal type before beginning a calibration procedure.

**EventID:** E8020260 (hex)

#### Message: 609

"The Calibration feature requested is not implemented."

**Further explanation:** The specified cal type can be one of many choices. For example, response calibrations require single standards, 1-Port calibrations require 3 standards, and 2-Port calibrations require up to 12 standards.

**Suggestions:** Be sure to measure only the standards needed for the specified cal type.

**EventID:** E8020261 (hex)

#### Message: 610

"The Calibration Class Acquisition requested is not valid for the selected Calibration Type. Please select a different acquisition or a different Calibration Type."

**EventID:** E8020262 (hex)

#### Message: 611

"The Calibration Standard data required for the selected caltype was not found."

**Severity:** Error

**Further explanation:** An unsuccessful attempt was made to retrieve a specified standard from the raw measurement buffer. The buffer should contain the raw measurements of cal standards stored during a calibration procedure.

**Suggestions:** Be sure the requested standard is required for the current cal type. Not all standards are needed for all cal types.

**EventID:** E8020263 (hex)

#### Message: 612

"The Error Term data required for the selected caltype was not found."

**Severity:** Error

**Further explanation:** An unsuccessful attempt was made to retrieve a specified error term from the error

correction buffer. The buffer should contain the error correction arrays for the current calibration.

**Suggestions:** Be sure the requested error term is required for the current cal type. Not all error terms are needed for all cal types.

EventID: E8020264 (hex)

#### Message: 613

The Calibration data set was not found.

**Severity:** Error

**Further explanation:** An unsuccessful attempt to access a cal set has been made. This may indicate a calset has been deleted or has been corrupted.

**Suggestions:** Try again or select another cal set. If the cal set appears in the cal set list, it may need to be deleted.

EventID: E8020265 (hex)

#### Message: 614

"The specified measurement does not have a calibration valid for Confidence Check. Please select a different measurement, or recall or perform a different Calibration Type."

**Severity:** Error

**Further explanation:** The measurement choice is prevented so that calibration will not be turned off. Not all cal types support all measurements. For example, an 1-Port cal on S11 can not be used to calibrate an S12 measurement. When a measurement is selected that does not have a calibration which can be applied, an informational message is displayed and calibration is turned off.

**Suggestions:** Use a full 2-Port calibration to be compatible with any S-Parameter.

EventID: E8020266 (hex)

#### Message: 615

" New calset created."

**Severity:** Informational message.

**Further explanation:** The newly created cal set will be automatically named and time stamped. If this is the beginning of a calibration procedure, the cal set will not be stored to memory until the calibration has completed successfully. The new cal set will be deleted if the calibration is canceled or does not otherwise complete successfully.

**Suggestions:** Informational

EventID: 68020267

#### Message: 617

The calset file: <x> appears to be corrupted and cannot be removed. Exit the application, remove the file, and restart.

**Severity:** Error

**Suggestions:** The cal set file is stored in the application home directory C:/Program Files/Agilent/Network Analyzer/PNACalSets.dat. Remove this file, then restart the application.

EventID: E8020269 (hex)

**Message: 634**

"The calset file: <x> load failed."

**Severity:** Error

**Further explanation:** The calset file contains a collection of calsets. The file resides on the hard drive.

**Suggestions:** Try restarting the application. If the failure persists, you may have to delete the cal set data file and restart the application. The cal set file is stored in the application home directory. C:/Program Files/Agilent/Network Analyzer/PNACalSets.dat. Remove this file, then restart the application.

**EventID:** E802027A (hex)

**Message: 635**

"The calset file: <x> save failed."

**Severity:** Error

**Further explanation:** The file operation detected an error. The save operation was aborted.

**Suggestions:** Retry.

**EventID:** E802027B (hex)

**Message: 636**

"A calset was deleted."

**Severity:** Informational

**Further explanation:** One of the calsets has been successfully deleted from the collection of calsets available. This can happen as the result of a user request or intentional operation.

**Suggestions:** None

**EventID:** 6802027C (hex)

**Message: 637**

"The version of the calset file: <x> is not compatible with the current instrument."

**Severity:** Error

**Further explanation:** A versioning error can prevent a calset from being used. This can happen as a result of instrument firmware upgrades.

**Suggestions:** If the versioning error is the result of firmware upgrade, you will have to re-install the old version of firmware to re-use the calset file. Or you can re-create the calsets with the current version of firmware.

The cal set file is stored in the application home directory C:/Program Files/Agilent/Network Analyzer/PNACalSets.dat. Remove this file, then restart the application.

**EventID:** E802027D (hex)

**Message: 638**

"Incompatible CalSets found: <x> of <y> stored calsets have been loaded."

**Severity:** Error

**Further explanation:** Errors were found on some of the calsets stored in the calset file. The errors may have been caused by versioning issues that may have corrupted the various calset keys.

**Suggestions:** Use the calset viewer to look at the contents of calset files. Delete the files that are corrupted.

**EventID:** 6802027E (hex)

**Message: 639**

"The Calset file: <x> was not found. A new file has been created."

**Severity:** Informational

**Further explanation:** The calset file should be stored on the hard drive. When the application is started, a search is done and the file is loaded if it can be found. If the file is not found, the analyzer will create a new file and display this message.

**Suggestions:** None

**EventID:** 6802027F (hex)

**Message: 640**

"The Calset specified is currently in use."

**Severity:** Error

**Further explanation:** This may indicate a conflict between multiple calset users attempting calibration tasks.

**Suggestions:** Save the instrument state. Preset the analyzer and recall the instrument state. This may abort any processes that may be in progress.

**EventID:** E8020280 (hex)

**Message: 641**

"The calset specified has not been opened."

**Severity:** Error

**Further explanation:** Multiple users may be attempting to access the calset.

**Suggestions:** Close multiple calset users so that only one user will access the calset.

**EventID:** E8020281 (hex)

**Message: 642**

"The maximum number of cal sets has been reached. Delete old or unused cal sets before attempting to create new ones."

**Severity:** Error

**Suggestions:** You may also delete the calsets data file.

The cal set file is stored in the application home directory. C:/Program Files/Agilent/Network\_Analyzer/PNACalSets.dat. Remove this file, then restart the application.

**EventID:** E8020282 (hex)

**Message: 643**

The requested power loss table segment was not found.

**Severity:** Error

**EventID:** E8020283 (hex)

**Message: 644**

"A valid calibration is required before correction can be turned on."

**Severity:** Error

**Further explanation:** This usually indicates a calibration procedure has not run to completion or that the selected measurement does not have a valid calibration available from within the currently selected cal set.

**Suggestions:** To find a calibration, you can select a Cal Set that contains the calibration needed for the current measurements. This error can happen if a calibration type is not specified before attempting to programmatically execute cal acquisitions. Specify the cal type before beginning a calibration procedure.

**EventID:** E8020284 (hex)

**Message: 645**

The cal data for <x> is incompatible and was not restored. Please recalibrate."

**Severity:** Warning

**Further explanation:** None

**Suggestions:** None

**EventID:** A8020285 (hex)

**Message: 646**

"CalSet not loaded, version is too new."

**Severity:** Error

**Further explanation:** An old version of firmware is attempting to run with a new calset version. The version is incompatible.

**Suggestions:** The calset can be removed. You may also delete the calsets data file if you are migrating between various firmware revisions often and you would like to avoid this error. The cal set file is stored in the application home directory. C:/Program Files/Agilent/Network Analyzer/PNACalSets.dat. Remove this file, then restart the application.

**EventID:** E8020286 (hex)

**Message: 647**

"Custom cal type not found."

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E8020287 (hex)

**Message: 648**

"Custom correction algorithm defers to the client for interpolation."

**Severity:** Informational

**EventID:** 68020288 (hex)

**Message: 649**

"Custom cal dll threw an exception."

**Severity:** Error

**EventID:** E8020289 (hex)

**Message: 650**

"Could not load the ecal.dll library"

**Severity:** Error

**EventID:** E802028A (hex)

**Message: 656**

"The argument specified is not a valid cal type."

**Severity:** Error

**EventID:** E8020290 (hex)

**Message: 657**

"The function found existing interpolated data"

**Severity:** Informational

**EventID:** 68020291 (hex)

**Message: 658**

"The function computed new interpolation values."

**Severity:** Informational

**EventID:** 68020292 (hex)

**Message: 659**

"The source power measurement failed."

**Severity:** Error

**Suggestions:** Please check GPIB, power meter settings and sensor connections.

**EventID:** E8020293 (hex)

**Message: 660**

"Duplicate session found. Close session and retry."

**Severity:** Error

**EventID:** E8020294 (hex)

**Message: 661**

"The session does not exist. Open the session and try again."

**Severity:** Error

**Further explanation:**

**EventID:** E8020295 (hex)

**Message: 662**

"Attempt to launch a custom calibration failed."

**Severity:** Error

**Further explanation:**

**EventID:** E8020296 (hex)

**Message: 663**

"Request to measure a cal standard failed."

**Severity:** Error

**Further explanation:** Please ensure you are requesting to measure standards which are defined for this calibration.

**EventID:** E8020297 (hex)

**Message: 664**

"Since Electronic Calibration Kit is selected, Mechanical Cal Kit parameter cannot be changed."

**Severity:** Error

**Further explanation:**

**EventID:** E8020298 (hex)

**Message: 665**

"Frequencies of the active channel are below minimum or above maximum frequencies of the ECal module factory characterization."

**Suggestions:** Change the channel frequencies, or select another ECal module.

**Severity:** Error

**EventID:** E8020299 (hex)

**Message: 666**

"Calset chosen for characterizing the ECal Module Ports %1 does not contain a calibration for PNA Ports %2."

**Severity:** Error

**Suggestions:** Go back to select another calset or to perform another cal.

**EventID:** E802029A (hex)

**Message: 667**

"ECal module only has sufficient memory remaining to store a maximum of %1 points in User Characterization %2."

**Severity:** Error

**Suggestions:** Decrease your number of points, or choose to overwrite another user characterization.

**EventID:** E802029B (hex)

**Message: 668**

Input values are non-monotonic. Cannot interpolate.



**Severity:** Error

**EventID:** E802029C (hex)

**Message: 669**

Interpolation target is out of range. Cannot interpolate.

**Severity:** Error

**EventID:** E802029D (hex)

**Message: 670**

Guided Calibration Error: <>

**Severity:** Error

**EventID:** E802029E (hex)

**Message: 671**

The first call to the guided calibration interface must be Initialize.

**Severity:** Error

**EventID:** E802029F (hex)

**Message: 672**

The selected thru cal method was not recognized.

**Severity:** Error

**EventID:** E80202A0 (hex)

**Message: 673**

Could not generate the error terms.

**Severity:** Error

**EventID:** E80202A1 (hex)

**Message: 674**

Guided calibration must be performed on the active channel

**Severity:** Error

**EventID:** E80202A2 (hex)

**Message: 675**

You can not start using calibration steps until you have successfully called [generate steps](#).

**Severity:** Error

**EventID:** E80202A3 (hex)

**Message: 676**

The step number given is out of range. Step numbers should be between 1 and the number of steps. 0 is not a

valid step number.

**Severity:** Error

**EventID:** E80202A4 (hex)

**Message: 677**

A calset was selected for channel: <n> without restoring stimulus.

**Severity:** Informational

**EventID:** 680202A5 (hex)

**Message: 678**

A calset was selected for channel: <n> restoring stimulus.

**Severity:** Informational

**EventID:** 680202A6 (hex)

**Message: 679**

The selected calset stimulus could not be applied to the channel.

**Severity:** Informational

**EventID:** 680202A7 (hex)

**Message: 680**

You attempted to measure power at a frequency outside the frequency range defined for the specified power sensor. Select another sensor or adjust the range for this sensor.

**Severity:** Error

**EventID:** E80202A8 (hex)

**Message: 681**

Specified frequency is outside the frequency ranges currently defined for the power meter's sensors.

**Severity:** Error

**EventID:** E80202A9 (hex)

**Message: 682**

Additional Calibration Standards need to be acquired in order to calibrate over the entire frequency range currently being measured.

**Severity:** Informational

**EventID:** 680202AA (hex)

**Message: 683**

The PNA failed to convert cal kits for use by unguided calibrations. The recommended action is to restore Cal Kit defaults.

**Severity:** Error

**EventID:** E80202AB (hex)

**Message: 684**

The PNA failed to convert cal kits for use by unguided calibrations. CalKit defaults have been restored.

**Severity:** Error

**EventID:** E80202AC (hex)

**Message: 685**

Power meter is reserved by a source power cal acquisition already in progress.

**Severity:** Error

**EventID:** E80202AD (hex)

**Message: 686**

Source power calibration has not been performed or uploaded for the specified channel and source port.

**Severity:** Error

**EventID:** E80202AE (hex)

**Message: 687**

Source power calibration data array size for the specified channel and source port does not match it's associated stimulus number of points.

**Severity:** Error

**EventID:** E80202AF (hex)

**Message: 688**

Source power calibration of Port <n> on Channel <n> was turned off because the correction array no longer exists.

**Severity:** Error

**EventID:** E80202B0 (hex)

**Message: 689**

This command can only be used on a measurement created with a specified calibration loadport.

**Severity:** Error

**EventID:** E80202B1 (hex)

**Message: 690**

Interpolation is turned off and you have changed the stimulus settings of the original calibration, so correction has been turned off.

**Severity:** Error

**EventID:** E80202B2 (hex)

**Message: 691**

Stimulus settings for your current instrument state exceeded the parameters of the original calibration, so correction has been turned off.

**Severity:** Error

**EventID:** E80202B3 (hex)

**Message: 692**

Fixturing: the requested S2P file cannot be read. Possible formatting problem.

**Severity:** Error

**EventID:** E80202B4 (hex)

**Message: 693**

Fixturing: the requested S2P file cannot be opened.

**Severity:** Error

**EventID:** E80202B5 (hex)

**Message: 694**

Fixturing: the requested S2P file cannot be interpolated. This is usually because the frequency range in the file is a subset of the current channel frequency range.

**Severity:** Error

**EventID:** E80202B6 (hex)

**Message: 695**

Cal Registers can only be used by one channel: the channel conveyed in the name of the cal register. The name cannot be changed.

**Severity:** Error

Further explanation: See [Cal Registers](#)

**EventID:** E80202B7 (hex)

**Message: 696**

Fixturing: cannot be enabled with Response Calibrations and has been turned off.

**Severity:** Error

**EventID:** E80202B8 (hex)

**Message: 697**

The selected calibration cannot be performed for this measurement.

**Severity:** Error

**EventID:** E80202B9 (hex)

**Message: 698**

Fitting: RemoveAllConnectors() should be called prior to calling AddConnector after a fit has been attempted.

**Severity:** Error

**EventID:** E80202BA (hex)

**Message: 699**

An attempt was made to acquire calibration data before the system was properly initialized.

**Severity:** Error

**EventID:** E80202BB (hex)

**Message: 700**

Use IGuidedCalibration for multiport calibration types.

**Severity:** Error

**EventID:** E80202BC (hex)

**Message: 701**

Guided calibration requires number of thru measurement paths be at least equal to the number of calibration ports minus 1.

**Severity:** Error

**EventID:** E80202BD (hex)

**Message: 702**

A thru path was specified that includes a port which the calibration was not specified to include.

**Severity:** Error

**EventID:** E80202BE (hex)

**Message: 703**

One or more of the ports to be calibrated was not found in the set of specified thru paths.

**Severity:** Error

**EventID:** E80202BF (hex)

---

**Hardware Errors**

---

**Message: 770**

Input power too high. Source power is off.

**Severity:** Warning

**EventID:** A8030302 (hex)

**Message: 771**

Source power restored.

**Severity:** Informational

**EventID:** 68030303 (hex)

**Message: 772**

"The spamnp.sys driver is not working. Check system hardware. ! Data will be simulated. !"

**Severity:** Error

**Further explanation:** The Network Analyzer application cannot locate the DSP board. Hardware or a driver may be malfunctioning. This is also common when attempting to run the Network Analyzer on a workstation.

**EventID:** E8030304 (hex)

**Message: 773**

"Instrument Serial Bus Not Working."

**Severity:** Error

**Further explanation:** The instrument EEPROM appears to contain either all ones or all zeros. A serial bus hardware failure prevents reading the EEPROM.

**EventID:** E8030305 (hex)

**Message: 848**

"Phase lock lost"

**Severity:** Error

**Further explanation:** The instrument source was not able to lock properly. This can be the result of broken hardware, poor calibration, or bad EEPROM values.

**Suggestions:** Perform source calibration. Click System / Service / Adjustments / Source Calibration

**EventID:** E8030350 (hex)

**Message: 849**

Phaselock restored.

**Severity:** Success

**EventID:** 0x28030351 (hex)

**Message: 850**

"Unknown hardware error."

**Severity:** Error

**Further explanation:** Hardware malfunctioned prevents communication with the DSP.

**EventID:** E8030352 (hex)

**Message: 851**

DSP communication lost.

**Severity:** Error

**EventID:** E8030353 (hex)

**Message: 852**

RF power off.

**Severity:** Error

**EventID:** E8030354 (hex)

**Message: 853**

RF power on.

**Severity:** Success

**EventID:** 28030355 (hex)

**Message: 854**

Hardware OK.

**Severity:** Success

**EventID:** 28030356 (hex)

**Message: 855**

"Source unleveled."

**Severity:** Error

**Further explanation:** The source was unable to properly level at the requested power. The indicated power may not be accurate.

**Suggestions:** Try a different power level. Recalibrate source, if problem persists.

**EventID:** E8030357 (hex)

**Message: 856**

Source leveled.

**Severity:** Success

**EventID:** 28030358 (hex)

**Message: 857**

Input overloaded.

**Severity:** Error

**EventID:** E8030359 (hex)

**Message: 858**

Input no longer overloaded.

**Severity:** Success

**EventID:** 2803035A (hex)

**Message: 859**

"Yig calibration failed."

**Severity:** Error

**Further explanation:** Internal self-calibration of YIG oscillator tuning failed.

**EventID:** E803035B (hex)

**Message: 860**

Yig calibrated.

**Severity:** Success

**EventID:** 2803035C (hex)

**Message: 861**

"Analog ramp calibration failed."

**Severity:** Error

**Further explanation:** Internal analog sweep ramp calibration has failed.

**EventID:** E803035D (hex)

**Message: 862**

Analog ramp calibrated.

**Severity:** Success

**EventID:** 2803035E (hex)

**Message: 863**

Source temperature high.

**Severity:** Error

**EventID:** E803035F (hex)

**Message: 864**

Source temperature OK.

**Severity:** Success

**EventID:** 28030360 (hex)

**Message: 865**

"EEPROM write failed."

**Severity:** Error

**Further explanation:** Attempt to store calibration data to EEPROM has failed. There is a possible hardware failure.

**EventID:** E8030361 (hex)

**Message: 866**

EEPROM write succeeded.

**Severity:** Success

**EventID:** 28030362 (hex)

**Message: 867**

Attempted I/O write while port set to read only.

**Severity:** Error

**Further explanation:** Attempt to write to an I/O data port while the port set to input/read only.



**Suggestions:** Set data port to write/output before attempting to write to port.

**EventID:** E8030363 (hex)

**Message: 868**

" Attempted I/O read from write only port.

**Severity:** Error

**Further explanation:** Attempt to read from an I/O data port while the port set to output/write only.

**Suggestions:** Set data port to read/input before attempting to read from port.

**EventID:** E8030364 (hex)

**Message: 869**

Invalid hardware element identifier.

**Severity:** Error

**EventID:** E8030365 (hex)

**Message: 870**

Invalid gain level setting.

**Severity:** Error

**EventID:** E8030366 (hex)

**Message: 871**

Device driver was unable to allocate enough memory. Please try rebooting.

**Severity:** Error

**EventID:** E8030367 (hex)

**Message: 872**

DSP Error. Please Contact Agilent Support. Technical Information: DSP Type 1

**Severity:** Error

**EventID:** E8030368 (hex)

**Message: 873**

DSP Error. Please Contact Agilent Support. Technical Information: DSP Type 2

**Severity:** Error

**EventID:** E8030369 (hex)

**Message: 874**

DSP Error. Please Contact Agilent Support. Technical Information: DSP Type 3

**Severity:** Error

**EventID:** E803036A (hex)

**Message: 875**

DSP Error. Please Contact Agilent Support. Technical Information: DSP Type 4

**Severity:** Error

**EventID:** E803036B (hex)

**Message: 876**

DSP Error. Please Contact Agilent Support. Technical Information: DSP Type 5

**Severity:** Error

**EventID:** E803036C (hex)

**Message: 910**

The trigger connection argument was not recognized as valid by the firmware.

**Severity:** Error

**EventID:** 0xE803038E (hex)

**Message: 911**

The trigger connection specified does not support this trigger behavior

**Severity:** Error

**EventID:** E803038F (hex)

**Message: 912**

The trigger behavior specified was not recognized as valid by the firmware.

**Severity:** Error

**EventID:** E8030390 (hex)

**Message: 913**

The trigger connection specified does not physically exist on this network analyzer

**Severity:** Error

**EventID:** E8030391 (hex)

**Message: 914**

Cannot set "Accept Trigger Before Armed", since this hardware configuration does not support edge triggering.

**Severity:** Error

**EventID:** E8030392 (hex)

**Message: 915**

Cannot set "Trigger Output Enabled", since this hardware configuration does not support BNC2.

**Severity:** Error

**EventID:** E8030393 (hex)

**Message: 916**

Exceeded maximum trigger delay.

**Severity:** Error

**EventID:** E8030394 (hex)

**Message: 917**

Exceeded minimum trigger delay.

**Severity:** Error

**EventID:** E8030395 (hex)

---

**Measure Errors**

---

**Message: 1024**

If you are going to display or otherwise use a memory trace, you must first store a data trace to memory.

**Severity:** Warning

**EventID:** A8040400 (hex)

**Message: 1025**

"The measurement failed to shut down properly. The application is in a corrupt state and should be shut down and restarted."

**Severity:** Error

**Further explanation:** This message is displayed if the PNA application becomes corrupt. If you continue to get this error, please call customer service

**EventID:** E8040401 (hex)

**Message: 1026**

The measurement failed to shut down properly. The update thread failed to exit properly.

**Severity:** Warning

**EventID:** A8040402 (hex)

**Message: 1027**

"Group Delay format with CW Time or Power sweeps produces invalid data."

**Severity:** Warning

**Further explanation:** Group Delay format is incompatible with single-frequency sweeps. Invalid data is produced.

**Suggestions:** Ignore the data or choose a different format or sweep type.

**EventID:** A8040403 (hex)

**Message: 1028**

**Severity:** Informational

"MSG\_LIMIT\_FAILED"

**Further explanation:** Limit line test failed.

**EventID:** 68040404 (hex)

**Message: 1029**

**Severity:** Informational

"MSG\_LIMIT\_PASSED"

**Further explanation:** Limit line test passed.

**EventID:** 68040405 (hex)

**Message: 1030**

"Exceeded the maximum number of measurements allowed."

**Severity:** Warning

**Further explanation:** See [Traces, Channels, and Windows on the PNA](#) for learn about maximum measurements.

**EventID:** A8040406 (hex)

**Message: 1031**

"Network Analyzer Internal Error. Unexpected error in AddNewMeasurement."

**Severity:** Warning

**Further explanation:** If you continue to get this message, contact product support.

**EventID:** A8040407 (hex)

**Message: 1032**

"No measurement was found to perform the selected operation. Operation not completed."

**Severity:** Warning

**Further explanation:** None

**Suggestions:** Create a measurement before performing this operation.

**EventID:** A8040408 (hex)

**Message: 1033**

The Markers All Off command failed.

**Severity:** Warning

**EventID:** A8040409 (hex)

**Message: 1034**

"A memory trace has not been saved for the selected trace. Save a memory trace before attempting trace math operations."

**Severity:** Warning

**Further explanation:** Must have a memory trace when trying to do Trace Math,

**EventID:** A804040A (hex)

**Message: 1035**

"MSG\_SET\_AVERAGE\_COMPLETE"

**Severity:** Informational

**Further explanation:** Informational for COM programming. Averaging factor has been reached.

**EventID:** 6804040B (hex)

**Message: 1036**

"MSG\_CLEAR\_AVERAGE\_COMPLETE"

**Further explanation:** Informational for COM programming. Averaging factor has NOT been reached.

**EventID:** 6804040C (hex)

**Message: 1037**

"Time Domain transform requires at least 3 input points. The transform has been deactivated."

**Severity:** Informational

**Further explanation:** None

**Suggestions:** Increase the number of points.

**EventID:** 6804040D (hex)

**Message: 1038**

Smoothing requires a scalar format, and has been deactivated.

**Severity:** Informational

**EventID:** 6804040E (hex)

**Message: 1039**

A receiver power calibration in this instrument state file cannot be recalled into this firmware version.

**Severity:** Warning

**EventID:** A804040F (hex)

**Message: 1104**

"Exceeded limit on number of measurements."

**Severity:** Error

**Further explanation:** See [Traces, Channels, and Windows on the PNA](#) for measurement limits.

**EventID:** E8040450 (hex)

**Message: 1105**

"Parameter not valid."

**Severity:** Error

**Further explanation:** A measurement parameter that was entered programmatically is not valid.

**EventID:** E8040451 (hex)

**Message: 1106**

"Measurement not found."

**Severity:** Error

**Further explanation:** Any of these could be the cause:

Trying to calibrate but already have maximum measurements.

Trying to do a confidence check but there is not a measurement.

Trying to create, activate, or alter a measurement through COM that has been deleted through the front panel.

Trying to use a trace name through programming that is not unique.

**EventID:** E8040452 (hex)

**Message: 1107**

"No valid memory trace."

**Severity:** Error

**Further explanation:** Must have a memory trace when trying to do Trace Math,

**Suggestions:** Store a memory trace.

**EventID:** E8040453 (hex)

**Message: 1108**

"The reference marker was not found."

**Severity:** Error

**Further explanation:** Attempted to create a delta marker without first creating a reference marker (COM only).

**EventID:** E8040454 (hex)

**Message: 1109**

"Data and Memory traces are no longer compatible. Trace Math has been turned off."

**Severity:** Error

**Further explanation:** Warning - channel setting has changed while doing trace math.

**Suggestions:** Store another memory trace and turn trace math back on.

**EventID:** A8040455 (hex)

**Message: 1110**

"Data and Memory traces are not compatible. For valid trace math operations, memory and data traces must have similar measurement conditions."

**Severity:** Error

**Further explanation:** Tried to do trace math without compatible data and memory traces.

**Suggestions:** Store another memory trace.

**EventID:** E8040456 (hex)

**Message: 1111**

"Marker Bandwidth not found."

**Severity:** Error

**Further explanation:** Could not find a portion of trace that meets the specified bandwidth criteria.

**EventID:** E8040457 (hex)

**Message: 1112**

"The peak was not found."

**Severity:** Error

**Further explanation:** Could not find portion of trace that meets peak criteria.

**Suggestions:** See Marker Peak criteria.

**EventID:** E8040458 (hex)

**Message: 1113**

"The target search value was not found."

**Severity:** Error

**Further explanation:** Could not find interpolated data point that meets search value.

**EventID:** E8040459 (hex)

**Message: 1114**

"Reflection measurement, such as S11, must supply an auxiliary port to disambiguate 2-port measurements on multiport instruments."

**Severity:** Error

**Further explanation:**

**EventID:** E804045A (hex)

**Message: 1115**

"The receiver power calibration has been turned off because the type of measurement or source port has changed, so the calibration is no longer valid."

**Severity:** Warning

**Further explanation:**

**EventID:** A804045B (hex)

**Message: 1116**

"Receiver power cal requires the active measurement to be of unratiod power."

**Severity:** Warning

**Further explanation:**

**EventID:** A804045C (hex)

**Message: 1117**

"There is currently no source power calibration associated with the channel and source port of the active

measurement. A source power cal should be performed or recalled before performing a receiver power calibration."

**Severity:** Warning

**Further explanation:**

**EventID:** A804045D (hex)

**Message: 1118**

"The attempted operation can only be performed on a standard measurement type."

**Severity:** Error

**Further explanation:**

**EventID:** E804045E (hex)

**Message: 1119**

"The custom measurement cannot be loaded because it is not compatible with the Network Analyzer hardware."

**Severity:** Error

**Further explanation:**

**Suggestions:**

**EventID:** E804045F (hex)

**Message: 1120**

"The custom measurement cannot be loaded because it is not compatible with the Network Analyzer software."

**Severity:** Error

**Further explanation:**

**EventID:** E8040460 (hex)

**Message: 1121**

"The custom measurement load operation failed for an unspecified reason."

**Severity:** Error

**Further explanation:**

**EventID:** E8040461 (hex)

**Message: 1122**

"The custom measurement data processing has generated an unhandled exception, and will be terminated. The PNA software may be in an unstable state and it is recommended that the PNA software be shutdown and restarted."

**Severity:** Error

**Further explanation:**

**EventID:** E8040462 (hex)

**Message: 1123**



"The attempted operation can only be performed on a custom measurement type."

**Severity:** Error

**Further explanation:**

**EventID:** E8040463 (hex)

**Message: 1124**

"The requested custom measurement is not available."

**Severity:** Error

**Further explanation:**

**EventID:** E8040464 (hex)

**Message: 1125**

"The requested custom algorithm was not found."

**Severity:** Error

**Further explanation:**

**EventID:** E8040465 (hex)

**Message: 1126**

"Normalization cannot be turned on because the measurement does not have a valid divisor buffer."

**Severity:** Error

**Further explanation:**

**EventID:** E8040466 (hex)

**Message: 1127**

"The Raw Data requested by the measurement could not be provided."

**Severity:** Warning

**Further explanation:**

**EventID:** A8040467 (hex)

**Message: 1128**

"The selected Sweep Type does not allow Transform and Gating. Transform and Gating disabled. "

**Severity:** Error

**Further explanation:**

**EventID:** E8040468 (hex)

**Message: 1129**

Memory trace can not be applied to this measurement

**Severity:** Error

**EventID:** E8040469 (hex)

**Message: 1130**

Normalization can not be applied to this measurement

**Severity:** Error

**EventID:** E804046A (hex)

**Message: 1131**

The data provided has an invalid number of points. It could not be stored

**Severity:** Error

**EventID:** E804046B (hex)

**Message: 1132**

The measurement stored in the save/recall state has an invalid version. It could not be loaded

**Severity:** Error

**EventID:** E804046C (hex)

**Message: 1133**

This data format argument for this operation must be "naDataFormat\_Polar"

**Severity:** Error

**EventID:** E804046D (hex)

**Message: 1134**

This data format argument for this operation must be a scalar data format

**Severity:** Error

**EventID:** E804046E (hex)

**Message: 1135**

The memory trace is not valid for the current measurement setup.

**Severity:** Error

**EventID:** E804046F (hex)

**Message: 1136**

This measurement is incompatible with existing measurements in this channel. Choose another channel.

**Severity:** Error

**EventID:** E8040470 (hex)

**Message: 1137**

Port extension correction is not available for offset frequency measurements. Port extension correction has been disabled.

**Severity:** Error

**EventID:** E8040471 (hex)

**Message: 1138**

Physical port number assignments for logical port mappings must be unique.

**Severity:** Error

**EventID:** E8040472 (hex)

---

**Parser Errors**

---

**Message: 1281**

"You have sent a read command to the analyzer without first requesting data with an appropriate output command. The analyzer has no data in the output queue to satisfy the request."

**Severity:** Error

**EventID:** 68050501 (hex)

**Message: 1282**

"You must remove the active controller from the bus or the controller must relinquish the bus before the analyzer can assume the system controller mode."

**Severity:** Error

**EventID:** E8050502(hex)

**Message: 1283**

"The analyzer did not receive a complete data transmission. This is usually caused by an interruption of the bus transaction."

**Severity:** Error

**EventID:** E8050503 (hex)

**Message: 1284**

"The instrument status byte has changed."

**Severity:** Informational

**EventID:** 68050504 (hex)

**Message: 1285**

"The SCPI command received has caused error number %1: "%2"."

**Severity:** Informational

**EventID:** 68050505 (hex)

**Message: 1286**

"The INET LAN server has been started as process number %1."

**Severity:** Informational

**EventID:** 68050506 (hex)

**Message: 1360**

"Execution of the SCPI command has failed"

**Severity:** Error

**EventID:** E8050550 (hex)

**Message: 1361**

" The INET/LAN device is not accessible."

**Severity:** Error

**EventID:** E8050551 (hex)

**Message: 1362**

"The INET/LAN driver was not found. "

**Severity:** Error

**EventID:** E8050552 (hex)

**Message: 1363**

"The INET/LAN driver was not found."

**Severity:** Error

**EventID:** E8050553 (hex)

**Message: 1364**

"The INET/LAN device is unable to acquire the necessary resources. "

**Severity:** Error

**EventID:** E8050554 (hex)

**Message: 1365**

"The INET/LAN device generated a generic system error. "

**Severity:** Error

**EventID:** E8050555 (hex)

**Message: 1366**

"Invalid address for the INET/LAN device."

**Severity:** Error

**EventID:** E8050556 (hex)

**Message: 1367**

"The INET I/O library was not found. "

**Severity:** Error

**EventID:** E8050557 (hex)

**Message: 1368**

"An error occurred in the INET system. "

**Severity:** Error

**EventID:** E8050558 (hex)

**Message: 1369**

"Access to the INET/LAN driver was denied. "

**Severity:** Error

**EventID:** E8050559 (hex)

**Message: 1370**

"Could not load error system message dll."

**Severity:** Error

**EventID:** E805055A (hex)

**Message: 1371**

"ErrorMessage.dll does not export the right function."

**Severity:** Error

**EventID:** E805055B (hex)

**Message: 1372**

"Custom scpi library was not able to be knitted"

**Severity:** Error

**EventID:** E805055C (hex)

**Message: 1373**

"Could not knit the scpi error messages from the ErrorMessage lib"

**Severity:** Error

**EventID:** E805055D (hex)

**Message: 1374**

Command is obsolete with this software version.

**Severity:** Error

**EventID:** E808055E (hex)

**Message: 1375**

CALC measurement selection set to none. Use [Calc:Par:Sel](#)

**Severity:** Error

**EventID:** E808055F (hex)

**Message: 1535**

"Parser got command: %1."

**Severity:** Informational

**EventID:** 680505FF (hex)

---

**Display Errors 1536 - 1621**

---

**Message: 1536**

"Exceeded the maximum of 4 traces in each window. The trace for <x> will not be added to window <x>."

**Severity:** Warning

**Further explanation:** None

**Suggestions:** Create the trace in another window. See the [PNA window limits](#).

**EventID:** A8060600 (hex)

**Message: 1537**

"Exceeded the maximum of 16 data windows. New window will not be created."

**Severity:** Warning

**Further explanation:** None

**Suggestions:** Create the trace in an existing window. See the [PNA window limits](#).

**EventID:** A8060601 (hex)

**Message: 1538**

"No Data Windows are present. Unable to complete operation."

**Severity:** Warning

**Further explanation:** Your remote SCPI operation tried to create a new measurement while there were no windows present

**Suggestions:** Create a new window before creating the measurement. See example [Create a measurement using SCPI](#)

**EventID:** A8060602 (hex)

**Message: 1539**

"No data traces are present in the selected window. Operation not completed."

**Severity:** Warning

**Further explanation:** None

**EventID:** A8060603 (hex)

**Message: 1540**

"Cannot complete request to arrange existing measurements in <x> windows due to the limit of <x> traces per window."

**Severity:** Informational

**Further explanation:** The arrange window feature cannot put the existing traces into the number of windows you requested because only 4 traces per window are allowed. [See Arranging Existing Measurements](#)

**Suggestions:** Either create more windows or delete some traces.

**EventID:** 68060604 (hex)

**Message: 1541**

"Unable to establish a connection with the specified printer."

**Severity:** Warning

**Further explanation:** None

**Suggestions:** Refer to Printer Help

**EventID:** A8060605 (hex)

**Message: 1542**

"Printout canceled."

**Severity:** Informational

**EventID:** 68060606 (hex)

**Message: 1616**

"Window not found."

**Severity:** Error

**Further explanation:** A window was specified in your program which does not exist.

**Suggestions:** Query the name of your window before specifying.

**EventID:** E8060650 (hex)

**Message: 1617**

"Duplicate window ID specified."

**Severity:** Error

**Further explanation:** None

**EventID:** E8060651 (hex)

**Message: 1618**

"Exceeded limit on number of windows."

**Severity:** Error

**Further explanation:** There is a limit of 4 windows per screen.

**EventID:** E8060652 (hex)

**Message: 1619**

"Exceeded limit on number of traces/window."

**Severity:** Error

**Further explanation:** There is a limit of 4 traces per window. See the [Traces, Channels, and Windows on the PNA](#).

**Suggestions:** Create the trace in another window

**EventID:** E8060653 (hex)

**Message: 1620**

"Trace not found."

**Severity:** Error

**Further explanation:** Your program tried to communicate with a non-existing trace.

**Suggestions:** Query the trace ID before writing to it.

**EventID:** E8060654 (hex)

**Message: 1621**

"The operating system does not recognize this printer."

**Severity:** Warning

**EventID:** A8060655 (hex)

**Message: 1622**

Duplicate trace ID specified.

**Severity:** Error

**EventID:** E8060656 (hex)

---

**Channel Errors 1792 -1878**

---

**Message: 1792**

"Sweep Complete."

**Severity:** Informational

**Further explanation:** None

**Suggestions:** None

**EventID:** 68070700 (hex)

**Message: 1793**

"All triggerable acquisitions have completed."

**Severity:** Informational

**Further explanation:**

**EventID:** 68070701 (hex)

**Message: 1794**

"The last trigger produced an aborted sweep."



**Severity:** Informational

**Further explanation:**

**EventID:** 68070702 (hex)

**Message: 1795**

"The segment list must be adjusted to have at least one active segment with more than 0 points to use segment sweep."

**Severity:** Informational

**Further explanation:** You attempted to change **Sweep type** to Segment sweep, but there is either no segments defined or no sweep points in the defined segments

**Suggestions:** Define at least one segment with at least one measurement point. See Segment sweep for more information

**EventID:** 68070703 (hex)

**Message: 1796**

"MSG\_SET\_CHANNEL\_DIRTY"

**Severity:** Informational

**Further explanation:** This informational message occurs when a channel setting has changed but the channel still has data that was taken with the previous setting. The following CLEAR message occurs when new channel data is taken.

**EventID:** 68070704 (hex)

**Message: 1797**

"MSG\_CLEAR\_CHANNEL\_DIRTY"

**Severity:** Informational

**Further explanation:** The previous SET message occurs when a channel setting has changed but the channel still has data that was taken with the previous setting. This CLEAR message occurs when new channel data is taken.

**EventID:** 68070705 (hex)

**Message: 1798**

Sweep time has changed from Auto to Manual mode. If desired to return to Auto mode, enter sweep time value of 0.

**Severity:** Informational

**EventID:** 68070706 (hex)

**Message: 1799**

"Set Sweep Completed"

**Severity:** Informational

**Further explanation:** This event occurs when a sweep and it's associated sweep calculations finish. This is typically when all sweeps on a channel complete.

**EventID:** 68070707 (hex)

**Message: 1800**

"Clear Sweep Completed"

**Severity:** Informational

**Further explanation:** This event occurs immediately after the SET SWEEP COMPLETED event. These two events set and clear the "Sweep Completed" bit (bit 4) on the SCPI Device Status register.

**EventID:** 68070708 (hex)

**Message: 1801**

"All Sweeps Completed and Processed"

**Severity:** Informational

**Further explanation:** This event occurs when all of the sweeps and sweep calculations are complete for a channel.

**EventID:** 68070709 (hex)

**Message: 1802**

Low Pass : Frequency limits have been changed.

**Severity:** Informational

**EventID:** 6807070A (hex)

**Message: 1803**

Low Pass : Number of points have been changed.

**Severity:** Informational

**EventID:** 6807070B (hex)

**Message: 1804**

Low Pass : Frequency limits and number of points have been changed.

**Severity:** Informational

**EventID:** 6807070C (hex)

**Message: 1805**

"Channel created"

**Severity:** Informational

**EventID:** 6807070D (hex)

**Message: 1806**

"Channel deleted"

**Severity:** Informational

**EventID:** 6807070E (hex)

**Message: 1872**

"Channel not found."

**Severity:** Error

**Further explanation:** A non-existent channel is being referenced under program control.

**Suggestions:** Query the channel number, then refer to it by number.

**EventID:** E8070750 (hex)

**Message: 1873**

"The requested sweep segment was not found."

**Severity:** Error

**Further explanation:** A non-existent sweep segment is being referenced under program control.

**EventID:** E8070751 (hex)

**Message: 1874**

"The sweep segment list is empty."

**Severity:** Error

**Further explanation:** Segment Sweep cannot be specified unless there is at least one defined segment. This error will only occur under remote control.

**EventID:** E8070752 (hex)

**Message: 1875**

"The number of points in active sweep segment list segments is 0."

**Severity:** Error

**Further explanation:** Segment Sweep cannot be specified unless there is at least data point specified in a segment. This error will only occur under remote control.

**EventID:** E8070753 (hex)

**Message: 1876**

"The specified source attenuator is not valid."

**Severity:** Error

**Further explanation:** You tried to set the Attenuator property on the Channel object on a PNA that doesn't have a source attenuator.

**EventID:** E8070754 (hex)

**Message: 1877**

"Log Frequency sweep cannot be selected with the current Number of Points. Please reduce Number of Points."

**Severity:** Error

**Further explanation:** The maximum number of points that can be used for Log sweep is 401.

**EventID:** E8070755 (hex)

**Message: 1878**

"The requested Number of Points is greater than can be selected for Log Frequency sweep."

**Severity:** Error

**Further explanation:** The maximum number of points that can be used for Log sweep is 401.

**EventID:** E8070756 (hex)

#### **Message: 1879**

"Response frequencies exceeded instrument range so Frequency Offset has been turned off."

**Severity:** Error

**Further explanation:** This error is returned whenever the instrument detects that the stimulus sweep setup and Frequency Offset settings result in computed response frequencies that exceed instrument limits. When this occurs, the instrument automatically turns off Frequency Offset to avoid the out-of-range conditions.

**Suggestions:** When this condition has occurred, change settings for either the stimulus frequencies or Frequency Offset so that the Response frequencies are within instrument bounds. Once this is done, Frequency Offset can once again be turned on.

**EventID:** E8070757 (hex)

#### **Message: 1880**

The total number of points for all the given segments exceeds the maximum number of points supported. The segments were not changed.

**Severity:** Error

**EventID:** E8070758 (hex)

#### **Message: 1881**

This instance of the Channels object was not used to place the channels in Hold, so no channels were resumed.

**Severity:** Error

**EventID:** E8070759 (hex)

#### **Message: 1882**

The port number was outside the range of allowed port numbers.

**Severity:** Error

**EventID:** E807075A (hex)

#### **Message: 1883**

More ports than are present are required for this operation.

**Severity:** Error

**EventID:** E807075B (hex)

---

### **General Errors**

---

#### **Message: 2048**

"The function you requested requires a capability provided by an option to the standard analyzer. That option is not currently installed."

**Severity:** Error

**Further explanation:** None

**Suggestions:** To view the options on your analyzer, click **Help / About Network Analyzer**. For more information see [PNA Options](#)

**EventID:** 68080800 (hex)

#### **Message: 2049**

"The feature you requested is not available on the current instrument."

**Severity:** Error

**Further explanation:** None

**EventID:** 68080801 (hex)

#### **Message: 2050**

"The feature you requested is incompatible with the current instrument state."

**Severity:** Error

**Further explanation:** None

**Suggestions:** None

**EventID:** 68080802 (hex)

#### **Message: 2051**

"File<x> has been saved."

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080803 (hex)

#### **Message: 2052**

"Attempt to save <x> failed."

**Severity:** Error

**Further explanation:** None

**Suggestions:** If using a floppy disk, ensure it is inside the drive and the disk is not full. Check the filename for special characters.

**EventID:** E8080804 (hex)

#### **Message: 2053**

"Attempt to recall file failed because <x> was not found."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080805 (hex)

**Message: 2054**

"<x> has a bad header."

**Severity:** Error

**Further explanation:** None

**Suggestions:** Recopy the file and / or delete the file.

**EventID:** E8080806 (hex)

**Message: 2056**

"Request to enter hibernate state."

**Further explanation:** None

**EventID:** 68080808 (hex)

**Message: 2057**

"Power up from automatic hibernate state. Program received PBT\_APMRESUMEAUTOMATIC Message."

**Further explanation:** None

**EventID:** 68080809 (hex)

**Message: 2058**

"Power up from suspend hibernate state. Program received PBT\_APMRESUMESUSPEND Message."

**Further explanation:** None

**EventID:** 6808080A (hex)

**Message: 2059**

"Power up from suspend hibernate state. Program received PBT\_APMRESUMECRITICAL Message."

**Severity:** Warning

**Further explanation:** None

**EventID:** A808080B (hex)

**Message: 2060**

"Power up from unknown hibernate state UI recovery called. Program received no PBT\_Message within the time allotted and is attempting recovery."

**Severity:** Warning

**Further explanation:** None

**EventID:** A808080C (hex)

**Message: 2061**

"<x> already exists. File is being overwritten."

**Further explanation:** Used only for remote applications

**EventID:** 6808080D (hex)

**Message: 2062**

"File has not been saved."

**Severity:** Error

**Further explanation:** Used only for remote applications

**EventID:** E808080E (hex)

**Message: 2063**

"File <x> has been recalled."

**Further explanation:** Used only for remote applications

**EventID:** 6808080F (hex)

**Message: 2064**

"State version in <x> is considered obsolete by this version of this code."

**Severity:** Error

**Further explanation:** You attempted to recall a file that is no longer valid.

**Suggestions:** You must recreate the file manually.

**EventID:** E8080810 (hex)

**Message: 2065**

"State version in <x> is newer than the latest version supported by this code."

**Severity:** Error

**Further explanation:** You attempted to recall a file that was created by a later version of the PNA application.

**Suggestions:** You must recreate the file manually.

**EventID:** E8080811 (hex)

**Message: 2066**

"Error occurred while reading file <x>"

**Severity:** Error

**Further explanation:** The file may be corrupt.

**Suggestions:** Try to recreate the file.

**EventID:** E8080812 (hex)

**Message: 2067**

"Windows shell error: <x>"

**Severity:** Error

**Further explanation:** None

**EventID:** E8080813 (hex)

**Message: 2068**

Send message timed out returning: <x>.

**Severity:** Error

**Further explanation:** None

**EventID:** E8080814 (hex)

**Message: 2069**

"Changing GPIB mode to System Controller."

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080815 (hex)

**Message: 2070**

"Changing GPIB mode to Talker Listener."

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080816 (hex)

**Message: 2071**

"The Network Analyzer can not be put in GPIB System Controller mode until the GPIB status is Local. Stop any remote GPIB programs which may be using the Network analyzer, press the Macro/Local key and try again. "

**Severity:** Informational

**Further explanation:** See [LCL and RMT Operation](#)

**Suggestions:** Press the Macro/Local key and try again.

**EventID:** 68080817 (hex)

**Message: 2120**

"This method can not be invoked through a late-bound COM call."

**Severity:** Error

**Further explanation:** None

**Suggestions:** Use the alternate method described in the [COM programming documentation](#)

**EventID:** E8080878 (hex)

**Message: 2128**

"The specified format is invalid."

**Severity:**Error

**Further explanation:** None

**EventID:** E8080850 (hex)

**Message: 2129**

"WINNT exception caught by Automation layer."



**Severity:** Error

**Further explanation:** None

**EventID:** E8080851 (hex)

**Message: 2130**

"Bad port specification."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080852 (hex)

**Message: 2131**

"Failed to find a printer."

**Severity:** Error

**Further explanation:** None

**Suggestions:** See [Connecting to a Printer](#)

**EventID:** E8080853 (hex)

**Message: 2132**

"Manual trigger ignored."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080854 (hex)

**Message: 2133**

"Attempt to set trigger failed."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080855 (hex)

**Message: 2134**

"Macro execution failed."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080856 (hex)

**Message: 2135**

"Specified macro definition is incomplete."

**Severity:** Error

**Further explanation:**

**EventID:** E8080857 (hex)

**Message: 2137**

"Block data length error."

**Severity:** Error

**Further explanation:** See [Getting Data from the Analyzer](#)

**EventID:** E8080859 (hex)

**Message: 2139**

"Requested data not found."

**Severity:** Error

**Further explanation:** None

**EventID:** E808085B (hex)

**Message: 2142**

"The parameter supplied was out of range, so was limited to a value in range before being applied to the instrument."

**Severity:** Success

**Further explanation:** None

**Suggestions:** View range limits before sending programming commands.

**EventID:** 2808085E (hex)

**Message: 2143**

The parameter supplied was out of range, so was limited to a value in range before being applied to the instrument.

**Severity:** Error

**EventID:** E808085F (hex)

**Message: 2144**

"Request failed. The required license was not found."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080860 (hex)

**Message: 2145**

"A remote call to the front panel has returned hresult <x>"

**Severity:** Error

**Further explanation:** This may indicate a problem with the front panel

**Suggestions:** Contact Technical support

**EventID:** E8080861 (hex)

**Message: 2146**

The recall operation failed.

**Severity:** Error

**Further explanation:**

**EventID:** E8080862 (hex)

**Message: 2147**

Attempt to save file failed.

**Severity:** Error

**Further explanation:**

**EventID:** E8080863 (hex)

**Message: 2148**

Recall attempt failed because file was not found.

**Severity:** Error

**Further explanation:**

**EventID:** E8080864 (hex)

**Message: 2149**

Recall file has a bad header.

**Severity:** Error

**Further explanation:**

**EventID:** E8080865 (hex)

**Message: 2150**

Recall file version is obsolete and no longer compatible with this instrument.

**Severity:** Error

**Further explanation:**

**EventID:** E8080866 (hex)

**Message: 2151**

The recall file contains an istate version newer than this instrument. A remote call to the front panel has returned hresult %1

**Severity:** Error

**Further explanation:**

**EventID:** E8080867 (hex)

**Message 2152**

"Front Panel <x>

**Severity:** Error

**Further explanation:** None

**EventID:** E8080868 (hex)

#### **Message 2153**

"Front Panel message"

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080869 (hex)

#### **Message 2154**

"Power Service <x>

**Severity:** Error

**Further explanation:** There is more than 1 instance of powerservice running. There should only be one running. This might happen after running install shield - especially when upgrading the CPU board.

**Suggestions:** Try rebooting. If this persists, please call [Customer Support](#).

**EventID:** E808086A (hex)

#### **Message 2155**

"Power Service <x>

**Severity:** Informational

**Further explanation:** None

**EventID:** 6808086B (hex)

#### **Message 2156**

"The Agilent Technologies GPIB driver can not be loaded or unloaded."

**Severity:** Error

**Further explanation:** None

**Suggestions:** If the problem persists, from the PNA desktop, right-click on My Computer. Click Properties, Click Hardware Tab. Click Device Manager Button. Expand GPIB Devices. Right-click and click Uninstall all GPIB interfaces devices. Reboot the PNA.

**EventID:** E808086C (hex)

#### **Message 2157**

"The National Instruments GPIB driver can not be loaded or unloaded."

**Severity:** Error

**Further explanation:** None

**Suggestions:** If the problem persists, from the PNA desktop, right-click on My Computer. Click Properties, Click Hardware Tab. Click Device Manager Button. Expand GPIB Devices. Right-click and click Uninstall all GPIB interfaces devices. Reboot the PNA.

**EventID:** E808086D (hex)

**Message 2158**

"The Agilent GPIB driver is loaded but it can not start its parser."

**Severity:** Error

**Further explanation:** None

**EventID:** E808086E (hex)

**Message: 2159**

The front panel is in remote mode.

**Severity:** Warning

**EventID:** A808086F (hex)

**Message: 2160**

The Registry Key specified could not be found.

**Severity:** Error

**EventID:** E8080870 (hex)

**Message: 2161**

An overcurrent condition has been detected on a probe plugged into the front panel.

**Severity:** Warning

**EventID:** A8080871 (hex)

**Message: 2162**

The operation timed out.

**Severity:** Error

**EventID:** E8080872 (hex)

**Message 2163**

"The Network Analyzer executed a preset."

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080873 (hex)

**Message 2164**

"Access to file denied."

**Severity:** Error

**Further explanation:** This means that the system can not open an output file for writing. Most likely because the file is write protected.

**Suggestions:** Pick another file name or file directory, check floppy disk hard disk write access.

**EventID:** E8080874 (hex)

**Message 2165**

"File type is structured storage."

**Severity:** Informational

**Further explanation:** None

**EventID:** 68080875 (hex)

**Message 2166**

"The trigger operation failed."

**Severity:** Error

**Further explanation:** None

**EventID:** E8080876 (hex)

**Message 2167**

"Argument out of range error."

**Severity:** Error

**Further explanation:** None

**Suggestions:** None

**EventID:** E8080877 (hex)

**Message: 2169**

The given COM object is not a custom application

**Severity:** Error

**EventID:** E8080879 (hex)

**Message: 2170**

The eventID supplied was not recognized as a valid PNA eventID

**Severity:** Error

**EventID:** E808087A (hex)

**Message: 2171**

The operation was canceled.

**Severity:** Error

**EventID:** E808087B (hex)

**Message: 2172**

High security level cannot be disabled directly. Only an instrument preset or recall of lower security instrument state will reset this security level.

**Severity:** Error

**EventID:** E808087C (hex)

**Message: 2173**

Local lockout mode is on. The PNA application will not accept input from front panel, keyboard or mouse until this mode is turned off from a remote interface.

**Severity:** Error

**EventID:** E808087D (hex)

**Message: 2174**

The SnP request is not valid for the selected measurement.

**Severity:** Error

**EventID:** E808087E (hex)

**Message: 2175**

Preset is not supported while this dialog or wizard is open. Close the dialog or wizard and then try again.

**Severity:** Error

**EventID:** E808087F (hex)

**Message: 2176**

The function you requested requires a capability provided by an option to the standard analyzer. That option is not currently installed.

**Severity:** Error

**EventID:** E8080880 (hex)

**Message: 2177**

Catastrophic error. Crash dump recorded at <n>

**Severity:** Error

**EventID:** E8080881 (hex)

**Message: 2178**

<n>

**Severity:** Error

**EventID:** E8080882 (hex)

**Message: 2179**

Failed to open gen.lic.

**Severity:** Error

**EventID:** E8080883 (hex)

---

Last modified:

Nov. 29, Updated bookmarks  
2006





## About Error Messages

---

PNA errors and Operating System errors are displayed and logged in an error file. You can choose how to display PNA errors, or choose to not display PNA errors at all.

- [Error Display](#)
- [View Error Log](#)
- [List of PNA Errors](#)
- [SCPI Errors](#)

### Other System topics

## Error Display

By default, error messages appear on the PNA screen for a brief period. You can choose to have them stay on the screen until you click an OK button, or have them not appear at all. When they stay on the screen, a Help button is available to provide further assistance.

### How to select the display of Error Messages

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Help]**
3. then **[Error Messages]**
4. then **[Error Display]**

#### Using a mouse with PNA Menus

1. Click **Help**
2. then **Error Messages**
3. then **Error Display**

◀ **Programming Commands** ▶



## Error Display dialog box help

On Preset, these settings revert to their defaults (enabled, timed popups).

**Enable Messages** Check to display all PNA error messages as they occur. Clear to suppress the display of PNA error messages. You can still view them in the [error log](#).

### Calibration Error Message Windows

**Timed Popups** Displays error messages on the screen for a duration of time proportional to the length of the message. You can then view the message in the [error log](#) and get further assistance.

**Confirmation Dialog boxes** Displays error messages in a standard dialog box. You then choose **OK** or **Cancel** to close the dialog box, or press Help to get further information on the error message.

## View Error Log

The PNA Error Log is a list of all events that have occurred. (Events are used in programming the PNA using COM.) PNA errors is a subset of PNA events. Only events with severity codes of ERROR are displayed on the PNA screen as they occur. From the error log, you can access further help with an error by selecting the error and clicking Help.

### How to view the Error Log

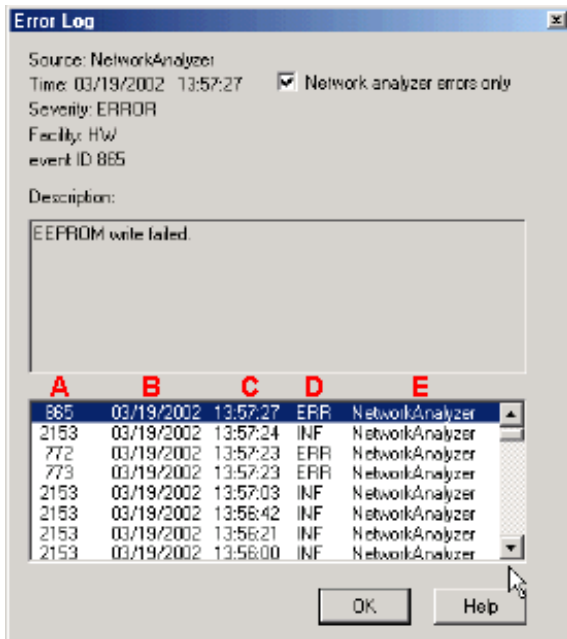
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Help]**
3. then **[Error Messages]**
4. then **[View Error Log]**

#### Using a mouse with PNA Menus

1. Click **Help**
2. then **Error Messages**
3. then **View Error Log**

No programming commands



### Error Log dialog box help

**Network analyzer errors only** Select to view only PNA errors. Clear to view all errors that occur on all applications of the computer.

**Description** Error message that appears on the PNA screen.

**A** - Event ID Error message number

**B** - Date the Error occurred

**C** - Time the Error occurred

**D** - Severity Code - All events have one of the following severity codes:

- SUCcess - the operation completed successfully
- INFormational - events that occur without impact on the measurement integrity
- WARning - events that occur with potential impact on measurement integrity
- ERRor - events that occur with serious impact on measurement integrity

**E** - Application in which the error occurred.

**OK** Closes the Dialog box

**Help** Provides further information on the selected Error message

To clear the Error Log:

1. From the **View** menu click **Minimize Application**
2. On the desktop, select **Start, Settings, Control Panel**
3. On the Control Panel, click **Administrative Tools**

4. On the Administrative Tools window, click **Event Viewer**
5. On the Event Viewer window, right-click **Application Log**
6. Select **Clear all Events**
7. If you want to save a file with the contents of the Event Log, click **Yes**. Otherwise, click **No**

To restore the PNA application, click on the PNA Analyzer taskbar button at the bottom of the screen

---

Last Modified:

24-Mar-2010    Updated to Error Display (A.09.20)

1-Jan-2007    MX Added UI

## Analyzer Accessories

---

- [Coax Mechanical Calibration Kits](#)
- [Waveguide Mechanical Calibration Kits](#)
- [Electronic Calibration \(ECal\)](#)
- [Mechanical Verification Kits](#)
- [Adapter and Accessory Kits](#)
- [Test Port Cables](#)
- [USB Peripherals](#)
- [Connector Care and Cleaning Supplies](#)

### Other Support topics

For product and order information:

- Visit [www.agilent.com/find/accessories](http://www.agilent.com/find/accessories)
- Use the search function to locate information about a particular accessory or view the entire RF and Microwave Test Accessories Catalog.

Accessories are available in these connector types:

- 50 ohm Type-N
- 75 ohm Type-N
- 3.5 mm
- 7 mm (APC-7)
- 7-16
- 2.92 mm
- 2.4 mm
- 1.85 mm
- 1 mm

Test port cables and a calibration kit are necessary for a complete measurement system.

A verification kit is used to verify corrected system performance.

[See the connector type for each PNA model](#)

### Coax Mechanical Calibration Kits

Model	Connector Type	Frequency Upper Limit
85032B	Type-N (50 Ohm)	6 GHz
85032F	Type-N (50 Ohm)	9 GHz
85054B	Type-N (50 Ohm)	18 GHz
85036E	Type-N (75 Ohm)	3 GHz
85050B	7 mm	18 GHz
85033D	3.5 mm	6 GHz
85038A	7-16	7.5 GHz
85033E	3.5 mm	9 GHz
85052B	3.5 mm	26.5 GHz
85052C	3.5 mm TRL	26.5 GHz
85056K	2.92 mm	50 GHz
85056A	2.4 mm	50 GHz
85058B/E <a href="#">(data-based)</a>	1.85 mm	67 GHz
85059A <a href="#">(data-based)</a>	1.00 mm	110 GHz

### Waveguide Mechanical Calibration Kits

Model	Connector Type	Frequency Range
X11644A	WR-90	8.2-12.4 GHz
P11644A	WR-62	12.4-18 GHz
K11644A	WR-42	18-26.5 GHz
R11644A	WR-28	26.5-40 GHz
Q11644A	WR-22	33-50 GHz
U11644A	WR-19	40-60 GHz
V11644A	WR-15	50-75 GHz

### Electronic Calibration (ECal)

Model	Connector Type	Frequency Range
<b>RF Two-Port</b>		
85091C	7 mm (APC-7)	300 kHz-9 GHz
85092C	Type-N (50 ohm) Port B available with 3.5 mm or 7-16 <sup>a</sup>	300 kHz-9 GHz
85093C	3.5 mm Port B available with Type-N (50 ohm) or 7-16 <sup>a</sup>	300 kHz-9 GHz
85096C	Type-N (75 ohm)	300 kHz-3 GHz
85098C	7-16 <sup>a</sup> Port B available with Type-N (50 ohm) or 3.5 mm	300 kHz-7.5 GHz
85099C	Type-F	300 kHz-3 GHz
<b>RF Four-Port</b>		
N4431B	3.5mm (f) (four-port), Type-N (f) (four-port), Mixed connector types	9 kHz <sup>b</sup> -13.5 GHz

N4432A Option 020	Type-N (f) (four-port)	300 kHz-18 GHz (available Feb. 2006)
N4432A Option 030	APC 7 (four-port)	300 kHz-18 GHz (available Feb. 2006)
N4433A Option 010	3.5mm (f) (four-port)	300 kHz-20 GHz (available Feb. 2006)
<b>Microwave Two-Port</b>		
N4690B	Type-N (50 ohm)	300 kHz-18 GHz
N4691B	3.5 mm	300 kHz-26.5 GHz
N4692A	2.92 mm	10 MHz-40 GHz
N4693A	2.4 mm	10 MHz-50 GHz
N4694A	1.85 mm	10 MHz-67 GHz
N4696BA	7 mm	300 kHz-18 GHz

a Limits ECal module high frequency to 7.5 GHz.

b Performance from 9 kHz to 300 kHz is valid only for the E5071C ENA Network analyzer with firmware version A.09.10 or higher.

## Verification Kits

Model	Connector Type	Frequency Range
85055A	Type-N (50 Ohm)	300 kHz-9 GHz
85053B	3.5 mm	300 kHz-26.5 GHz
85057B	2.4 mm	.045-50 GHz
R11645A	WR-28	26.5-40 GHz
Q11645A	WR-22	33-50 GHz

## Adapters and Accessory Kits



Model	Description
11878A	Type-N to 3.5 mm Adapter Kit
11525A	Type-N (m) to 7 mm (APC-7)
11853A	Type-N Accessory Kit
11900B	2.4 mm (f) to 2.4 mm (f)
11900C	2.4 mm (f) to 2.4 mm (m)
85130G	Test Port Adapter Set, 2.4 mm (f) to 2.4 mm (m,f)
11901B	2.4 mm (f) to 3.5 mm (f)
11901D	2.4 mm (f) to 3.5 mm (m)
85130F	Test Port Adapter Set, 2.4 mm (f) to 3.5 mm (m,f)
11902B	2.4 mm (f) to 7 mm (APC-7)
11920A	1 mm (m) to 1 mm (m)
11920B	1 mm (f) to 1 mm (f)
11920C	1 mm (m) to 1 mm (f)
11921A	1 mm (m) to 1.85 mm (m)
11921B	1 mm (f) to 1.85 mm (f)
11921C	1 mm (m) to 1.85 mm (f)
11921D	1 mm (f) to 1.85 mm (m)
11922A	1 mm (m) to 2.4 mm (m)
11922B	1 mm (f) to 2.4 mm (f)
11922C	1 mm (m) to 2.4 mm (f)
11922D	1 mm (f) to 2.4 mm (m)

## Test Port Cables

Model	Description
N4697E	1.85 mm (f) to 1.85 mm (rugged f) flexible (single)
N4697F	1.85 mm (rugged f, f) to 1.85 mm (rugged m, rugged f) flexible (set)
N6315A	Type-N (m) to Type-N (f), 16 in. (single)
N6314A	Type-N (m) to Type-N (m), 24 in. (single)
85133D	2.4 mm (f) to 2.4 mm (m,f) semi-rigid (set)
85133F	2.4 mm (f) to 2.4 mm (m,f) flexible (set)
85134D	2.4 mm (f) to 3.5 mm (m,f) semi-rigid (set)
85134F	2.4 mm (f) to 3.5 mm (m,f) flexible (set)

## USB Peripherals

Model	Description
N4688A	<b>CD RW drive</b> - with USB cable.
N4689A	<b>USB 4-port hub</b> - for connecting additional USB peripherals.
82357A	<b>USB/GPIB Interface</b> - for controlling GPIB devices through USB. Learn more about <a href="#">using the 82357A with the PNA</a>

## Connector Care and Cleaning Supplies

[See Connector Care](#)

Part Number	Description
1401-0225	Standard End-Cap, Type-N (m)
1401-0248	ESD Safe End-Cap, Type-N (m)
1401-0225	Standard End-Cap, Type-N (f)
1401-0247	ESD Safe End-Cap, Type-N (f)

## ESD Supplies

[See ESD topic.](#)

Part Number	Description
9300-1367	Adjustable antistatic wrist strap
9300-0980	Antistatic wrist strap grounding cord (5 foot)
9300-0797	Static control table mat (2 foot x 4 foot) with earth ground wire
9300-1126	ESD heel strap

Last Modified:

9-Jan-2009    Added footnote for N4431B

1-Dec-2008    Added link to connector care

## 82357A USB / GPIB Interface

---

The Agilent 82357A is an adapter that creates a GPIB Interface from one of your unused PNA USB ports.

- [Applications](#)
  - [Installing](#)
  - [Configuring](#)
  - [Connecting](#)
  - [Communicating with other Equipment](#)
- 

### Applications

The 82357A can be used for the following PNA applications:

- **Frequency Converter Application** - The 82357A is included with the Frequency Converter Application ([option 083](#)). External sources MUST be connected to this Interface if controlling the PNA using an external PC. See [connecting diagram](#) below. To learn more, see [Configure an external LO source](#).

**Note:** If the PNA is [hibernated](#) during an FCA measurement involving an external source under FCA control, and then the PNA is restarted, a VISA error message will appear stating "VI\_ERROR\_INV\_OBJECT." To correct this problem, the 82357 USB/GPIB interface must be reinitialized after hibernation. This is done by clicking on the Accept button in the interface initialization dialog box. The green READY light on the interface will illuminate.

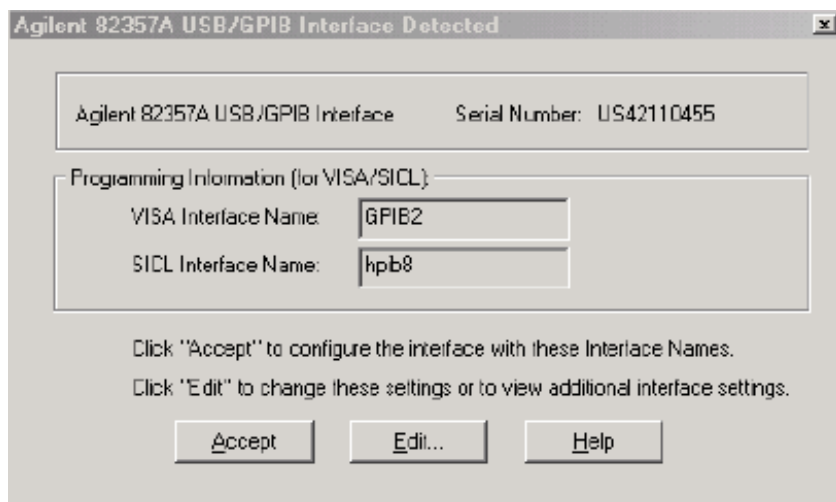
- **PNA Controller** - The 82357A can be used by the PNA to control other GPIB devices. This frees the default GPIB interface to perform other GPIB operations, such as control the PNA from an external PC.
- **Source Power Cal** - The 82357A can be used to run a source power calibration.

### Installing the 82357A USB/GPIB Interface

1. [Download and install firmware](#) PNA revision 3.0 or greater. To check the revision of your PNA firmware, click **Help** then **About Network Analyzer**.
2. Upgrade to the latest Agilent IO libraries from the CDROM that was shipped with the 82357A. If not available, download them from [www.Agilent.com](http://www.Agilent.com) (search for **82357A**)

### Configure the 82357A USB/GPIB Interface

When the 82357A is connected to the PNA USB, the following dialog box appears:



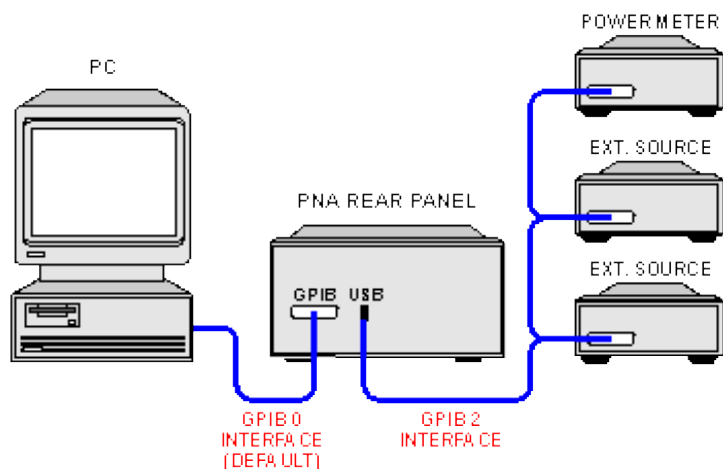
Normally, you do NOT need to edit these settings. The 82357A USB/GPIB Interface is configured automatically as the next unused VISA interface. This is usually **GPIB2** unless you have already configured it for another purpose.

If the VISA Interface Name appears as GPIB0 or GPIB1, these Interfaces must be returned to their default settings for the 82357A to work properly with the PNA. [See Configure for VISA / SICL to learn how.](#)

### Connecting the 82357A USB/GPIB Interface

The following diagram illustrates how to connect GPIB test equipment using the USB/GPIB Interface.

- Plug the USB/GPIB Interface into any unused PNA USB port.
- The default GPIB Interface and USB/GPIB Interface should never be connected together.



### Communicating with Equipment Connected to the USB/GPIB Interface

- The Frequency Converter Application will automatically find and communicate with test equipment that is connected to the USB/GPIB Interface.
- Source power calibration will automatically find and communicate with the power meter that is connected to the USB/GPIB Interface.

- To control other devices through your own program using the 82357A, you must include the new GPIB Interface number when addressing the devices.

## Firmware Upgrade

---

PNA firmware upgrades are available to you at no cost in a self-extracting Install Shield file. The upgrade includes the PNA application, Online help, and Service Utilities. Note: The file is **at least** 50 MB.

**Note:** The [CPU speed](#), [amount of RAM](#), and [Operating System](#) in your PNA may limit your ability to upgrade firmware. See [http://na.tm.agilent.com/pna/firmware/PNA\\_support\\_matrix.doc](http://na.tm.agilent.com/pna/firmware/PNA_support_matrix.doc).

The following options are available for you to upgrade your PNA application:

- [Auto-Check](#) and [AgileUpdate](#) If your PNA is connected to the Internet, these utilities will automatically check for, download, and install, the new firmware and associated files when the PNA application is started. You will be prompted before this occurs.
- [Website Access](#) If your PNA is NOT connected to the Internet, but you have a PC that is, you can download the PNA firmware and associated files to a storage medium.

To manually check the version of firmware on the PNA, click **Help**, then [About Network Analyzer](#).

### **Note: After a firmware upgrade...**

- Custom Cal Kits must be imported. [Learn more](#)
- If a different desktop icon named "Network Analyzer" exists, the shortcut to the PNA application will assume the same icon. Right-click on the desktop, then click **Refresh**.

## [Other Support Topics](#)

### Auto-Check

With Internet access to your PNA, Auto-Check automatically and regularly checks the Internet for new PNA firmware revisions. If a new revision is found, a notification message prompts you to run the [AgileUpdate](#) utility, which then performs the actual download.

Without Internet access to your PNA, Auto-Check provides a reminder prompt at the selected intervals.

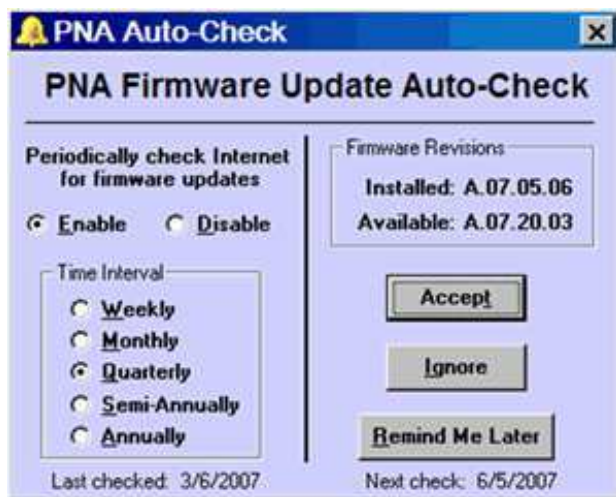
Auto-Check is run only when the PNA application is started. Once the PNA application is running, it will not check for updates again until it is restarted.

When Auto-Check runs, it checks the following conditions:

- Is there an active connection to the Internet?
- Is the Auto-Check utility enabled?
- Is it time to check for new firmware?
- Does new firmware exist?

If all of these conditions are true, Auto-Check shows the following dialog box.

If all of these conditions are NOT true, or to change these settings at any time, click **System, Service**, then **AgileUpdate**. From within AgileUpdate, click **AutoCheck**. These preferences are stored in the PNA registry. Future firmware upgrades will not change these settings.



#### PNA Auto-Check dialog box help

**Enable** When the PNA application is started, Auto-Check will search the PNA website for firmware updates at the selected time interval.

**Disable** When the PNA application is started, Auto-Check will NOT search the PNA website for firmware updates.

**Time Interval** Select the time interval Auto-Check is to search for firmware updates.

**Accept** Starts update process.

**Ignore** No further action is taken until the selected time interval has elapsed.

**Remind Me Later:** This window is displayed again after 1-20 days depending upon the time interval selected.

## AgileUpdate

**Note:** You must have administrative privileges on the analyzer to run this utility. See [Set Up Analyzer Users](#).



## How to start AgileUpdate

Connect the PNA to the Internet. A LAN connection is recommended because a firmware download can take many hours using a modem.

### Using front-panel HARDKEY [softkey] buttons

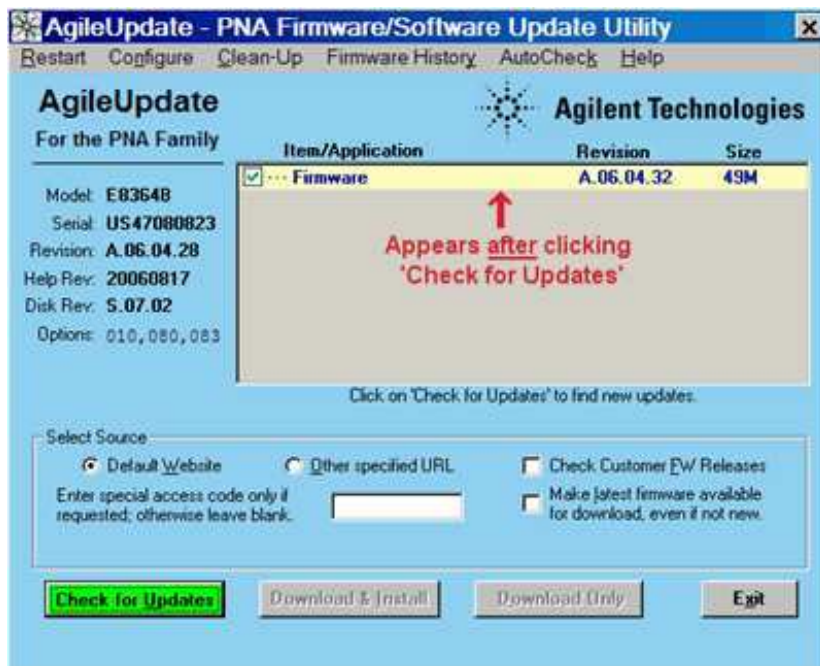
1. Press **SYSTEM**
2. then **[Service]**
3. then **[AgileUpdate]**

### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Service**
4. then **AgileUpdate**

1. Click **Check for Updates**.
2. If updates exist, click **Download & Install**.

No programming commands are available for this feature



## AgileUpdate dialog box help

**Note:** Your privacy is important to Agilent. AgileUpdate does NOT send ANY information from the PNA to the server. It only downloads from the server to the PNA.

**Restart** Click to restart from the beginning.

**Configure** Click to launch the [Configure dialog box](#).

**Clean-up** Click to delete all but the two most recent install shield packages from the PNA hard drive.

**Firmware History** Available after clicking **Check for Updates**.

**Auto-Check** Launches the [Auto-Check](#) dialog box.

**Item / Application** Lists the items available for download at the firmware website.

- Click items with **i** to read more information about the download.
- Items in **RED** should be downloaded and installed individually.
- Multi-language help includes all help files except English.

**Note:** The firmware includes the help file. Therefore, only the firmware checkbox will be selected if a new version for both the firmware and the help file are available.

### Select Source

**Default Website** The Agilent site that contains upgrade FW.

**Other Specified URL** Click if you were instructed to get firmware from a different website.

**Check Customer FW Releases** Check this box to also check Customer Releases in addition to Production Releases. This setting provides you with the very latest PNA firmware. Customer Releases are fully supported but have not yet been tested in all production models. Customer Releases take precedence over Production Releases. This setting is remembered and applied the next time AgileUpdate is run.

**Special Access Code...** Type in the code if you were given one from Agilent Technical Support. Otherwise, leave blank.

**Make Latest Firmware Available...** Select this checkbox if you want to download the latest firmware, even if it is not new.

**Check for Updates** Click to look for firmware updates at the Agilent website. If there are newer versions, the files will be listed.

**Download and Install** When updates are found, this selection becomes available. Some files may be pre-checked. Be sure the corresponding boxes are checked for the files you want to download. Then click to download and install the update.

**Download Only** Click to download the files to the analyzer hard disk and install the files at a later time. At that time, click **Install from File**.



### Configuration dialog box help

**Note:** If AgileUpdate will not connect, try to access ANY Internet website. Contact your local IT department if necessary.

#### Proxy Setting

**No Proxy or Default Proxy** Click if you use a LAN connection. AgileUpdate will automatically use the proxy specified in Internet Explorer.

**Use specified Proxy / Port** Click to enter the proxy name and port. The format is: proxyName:portNumber. (The proxy port number is typically 8088).

**Internet timeout** If you are using an automatic dial-up Internet connection you may need to increase the timeout.

**Current Connection Status** Shows the current status of the PNA connection to the Internet.

**Note:** These settings are NOT saved; they must be re-entered each time AgileUpdate is run.

### Agilent Website Access

If you cannot access the Internet directly with your PNA, you can use an external PC with Internet access to download the file from the Agilent website. You can then transfer the file from your PC to your analyzer over a LAN or other means.

1. Go to <http://na.tm.agilent.com/pna/firmware>
2. Click on the firmware to be downloaded.
3. Save the program to disk (hard drive of your PC).
4. Transfer the file from your PC to your PNA using LAN, CD, or USB Pen drive.
5. Double-click the file on the PNA.

**Warning:** You can save the upgrade file to your PC, but do not attempt to install the PNA application on your PC. It will alter system settings and can result in system crashes.

---

Last Modified:

12-Oct-2009	Added customer release (9.2)
13-Mar-2008	Added limitations note

## PNA Configurations and Options

---

Included with each PNA is a mouse and keyboard. This topic presents standard PNA models and the available options and upgrades.

- PNA Models
  - [PNA-X Series](#)
  - [PNA-L Series](#)
  - [Microwave Models](#)
  - [RF Models](#) (Discontinued)
  - [mmWave Model](#)
  - [Measurement Receiver](#) N5264A
- [Common Options and Upgrade Kits](#)
- [Warranty Period](#)

To view the options that are installed on your analyzer, click **Help** then **About Network Analyzer**

A documentation CD-ROM is no longer included in each PNA shipment (Feb.05).

### New PNA 'C' Models

In Spring 2008, new PNA 'C' models were introduced that will eventually replace the [PNA-L](#) and [Microwave](#) 'A' and 'B' models. The new 'C' models have the [PNA-X User Interface](#) capabilities, including touchscreen and softkeys. In addition:

- The 'C' models have a bigger 8.4" display screen.
- The 'C' models share the same [Specifications](#) as their 'A' and 'B' model counterparts.
- Upgrades from 'A' and 'B' to 'C' models are available.

**Note:** The term '**Legacy**' that is used in this help file refers to the PNA models that were introduced before the PNA-X (N5230A, E836xA/B, and all discontinued models).

### [Other Support Topics](#)

## PNA-X Series

- **N5241A: 10 MHz to 13.5 GHz**

- **N5242A:** 10 MHz to 26.5 GHz
- **N5244A:** 10 MHz to 43.5 GHz
- **N5245A:** 10 MHz to 50.0 GHz

The **N5244A** and **N5245A** do NOT have a Hibernate mode. [Learn more.](#)

## PNA-X Base Options

Block Diagrams are available at the end of each [specifications](#) document.

Option	Requires	Test Ports	Reference Receivers	Sources	Test Set
<b>200</b>	<a href="#">Opt 080</a>	<b>2</b>	<b>2</b>	<b>1</b>	2-port models include six front-panel access loops
<b>400</b>	<b>None</b>	<b>4</b>	<b>4</b>	<b>2</b>	4-port models include twelve front-panel access loops

## PNA-X Extended Options

[See more PNA-X options](#)

Option	Required Model/Options	Description
<b>224</b>	200, 219, and <a href="#">080</a>	To 2-port model, add internal 2nd source, combiner, and mechanical switches.
<b>423</b>	400, 419 and <a href="#">080</a>	To 4-port model, add combiner and mechanical switches.
<b>219</b>	200	To 2-port model, add: <ul style="list-style-type: none"> <li>• 65 dB source attenuator (5dB steps)</li> <li>• Bias-tees between each source and each test port</li> <li>• 35 dB receiver attenuator (5dB steps).</li> </ul>
<b>419</b>	400 ( <a href="#">080</a> recommended)	To 4-port model, add: <ul style="list-style-type: none"> <li>• 65 dB source attenuator (5dB steps).</li> <li>• Bias-tees between each source and each test port.</li> <li>• 35 dB receiver attenuator (5dB steps).</li> </ul>
<b>008</b>	All PNA-X	Pulsed App. <a href="#">Learn more.</a>
<b>020</b>	All PNA-X	Add <a href="#">IF inputs on the rear panel</a> for antenna and millimeter-wave.

021	All PNA-X	Add pulse modulator to internal Source1. <a href="#">Learn more.</a>
022	All PNA-X	Add pulse modulator to internal Source2. <a href="#">Learn more.</a>
025	All PNA-X	Add four internal pulse generators. <a href="#">Learn more.</a>
028	All PNA-X and <a href="#">080</a>	Noise Figure measurements on <a href="#">amplifiers</a> or <a href="#">converters</a> using a standard PNA receiver.
029	N5241A or N5242A with 080 and one of the following: 219, 224, 419 or 423	Noise Figure Application. Adds hardware and firmware for high-accuracy noise figure measurements on <a href="#">amplifiers</a> or <a href="#">converters</a> using internal low-noise receivers or a standard PNA receiver (Opt 028).
H29	N5244A or N5245A with 423 and 080.	Same as Opt 029, but on a <a href="#">N5244A or N5245A</a> . <a href="#">Learn more.</a>
036	N5244A or N5245A only.	<b>No longer available.</b> Add wide-pulse modulator to internal Source 1. Limits source modulation to 117 ns.
037	N5244A or N5245A only.	<b>No longer available.</b> Add wide-pulse modulator to internal Source 2. Limits source modulation to 117 ns.
H85	N5242A	Allows high-power measurements up to 20 Watts (+43 dBm). from 10 MHz to 26.5 GHz. Similar to the PNA-X -219 or -419 but deletes the bias tees from the test set. <a href="#">Learn more.</a>
086	All PNA-X	Gain Compression Application. Adds firmware for fast and accurate gain compression measurements. <a href="#">Learn more.</a>
087	Opt 224 or 423 <a href="#">Opt 080</a>	Swept IMD and IM Spectrum. Adds firmware for fast and accurate IMD measurements on amplifiers and converters. <a href="#">Learn more.</a>
118	None	Fast CW mode enables 500 million point data buffer. <a href="#">Learn more.</a>
460	Opt 400	iTMSA. Adds firmware for Integrated True Mode Balanced measurements. <a href="#">Learn more.</a>

### PNA-L Series Model N5230A and N5230C [See note](#)

**Note:** To see if your PNA has a **Reference Receiver for each Test Port**, scroll down to see the number of test ports and number of reference receivers. If they are equal, then there is a reference receiver for each test port. Block Diagrams are available at the end of each [specifications](#) document.

Option	Frequency Range	Test Ports	Reference Receivers	Connector	Test Set
020	300 KHz* to 6 GHz	2	2	3.5 mm Male	Standard
025	300 KHz* to 6 GHz	2	2	3.5 mm Male	Configurable
120	300 KHz* to 13 GHz	2	2	3.5 mm Male	Standard
125	300 KHz* to 13 GHz	2	2	3.5 mm Male	Configurable
140	300 KHz* to 13.5 GHz	4	1	3.5 mm Male	Standard
145	300 KHz* to 13.5 GHz	4	1	3.5 mm Male	Configurable <sup>2</sup>
146	300 KHz* to 13.5 GHz	4	1	3.5 mm Male	Configurable <sup>3</sup> + Second Source
220	10 MHz* to 20 GHz	2	2	3.5 mm Male	Standard
225	10 MHz* to 20 GHz	2	2	3.5 mm Male	Configurable
240	300 KHz* to 20 GHz	4	1	3.5 mm Male	Standard
245	300 KHz* to 20 GHz	4	1	3.5 mm Male	Configurable <sup>2</sup>
246	300 KHz* to 20 GHz	4	1	3.5 mm Male	Configurable <sup>3</sup> + Second Source
420	10 MHz* to 40 GHz	2	2	2.4 mm Male	Standard
425	10 MHz* to 40 GHz	2	2	2.4 mm Male	Configurable
520	10 MHz* to 50 GHz	2	2	2.4 mm Male	Standard
525	10 MHz* to 50 GHz	2	2	2.4 mm Male	Configurable

\* Typical specs apply

#### Test Set

**Standard** NO Configurable Test Set.

**Configurable (2-port)** - Adds six front panel access loops and two 60 dB step attenuators in 10 dB steps.

**Configurable<sup>2</sup> (4-port)** - Adds nine front panel access loops and one 60 dB step attenuator in 10 dB steps.

**Configurable<sup>3</sup> (4-port)** - Adds nine front panel access loops and two 60 dB step attenuators in 10 dB steps.



## Microwave Standard Models ([see options](#))

Block Diagrams are available at the end of each [specifications](#) document.

PNA Model	Frequency Range	Test Ports	Reference Receivers	Connector Type	Front Panel Jumpers
E8361A E8361C	10 MHz* to 67 GHz (tunable to 70 GHz)**	2	2	1.85 mm Male	0
<b>E8362A</b>	45 MHz to 20 GHz	2	2	3.5 mm Male	0
<b>E8363A</b>	45 MHz to 40 GHz	2	2	2.4 mm Male	0
<b>E8364A</b>	45 MHz to 50 GHz	2	2	2.4 mm Male	0
E8362B E8362C	10 MHz* to 20 GHz	2	2	3.5 mm Male	0
E8363B E8363C	10 MHz* to 40 GHz	2	2	2.4 mm Male	0
E8364B E8364C	10 MHz* to 50 GHz	2	2	2.4 mm Male	0
<b>Legend</b> <p><b>No longer produced</b></p> <p>* Typical specs apply from 10 to 45 MHz</p> <p>** Typical specs apply from 67 to 70 GHz</p> <p><b>New models</b> <a href="#">See note</a></p>					

## RF Standard Models ([see options](#))

Block Diagrams are available at the end of each [specifications](#) document.

PNA Model	Frequency Range	Ports	Connector Type	Reference Receivers	Front Panel Jumpers
<b>E8356A</b>	300 kHz to 3 GHz	<b>2</b>	Type-N Female	<b>2</b>	<b>4</b>
<b>E8357A</b>	300 kHz to 6 GHz	<b>2</b>	Type-N Female	<b>2</b>	<b>4</b>
<b>E8358A</b>	300 kHz to 9 GHz	<b>2</b>	Type-N Female	<b>2</b>	<b>4</b>
<b>E8801A</b>	300 kHz to 3 GHz	<b>2</b>	Type-N Female	<b>1</b>	<b>0</b>
<b>E8802A</b>	300 kHz to 6 GHz	<b>2</b>	Type-N Female	<b>1</b>	<b>0</b>
<b>E8803A</b>	300 kHz to 9 GHz	<b>2</b>	Type-N Female	<b>1</b>	<b>0</b>
<b>N3381A</b>	300 kHz to 3 GHz	<b>3</b>	Type-N Female	<b>1</b>	<b>0</b>
<b>N3382A</b>	300 kHz to 6 GHz	<b>3</b>	Type-N Female	<b>1</b>	<b>0</b>
<b>N3383A</b>	300 kHz to 9 GHz	<b>3</b>	Type-N Female	<b>1</b>	<b>0</b>
All RF models are no longer produced (June 1, 2005).					

## Millimeter Wave PNA

Block Diagrams are available at the end of each [specifications](#) document.

PNA Model	Frequency Range	Ports	Connector Type	Reference Receivers	Front Panel jumpers
<b>N5250A</b> <b>N5250C</b> <a href="#">See note</a>	10 MHz* to 110 GHz	<b>2</b>	1.0 mm	<b>2</b>	<b>N5260A</b>
<p>* Typical specs apply from 10 to 45 MHz  Test heads to 325 GHz are also available  Upgrade your existing E836xB with the following:</p> <ul style="list-style-type: none"> <li>• <a href="#">H11 option</a></li> <li>• N5260A Test Set</li> <li>• Millimeter-Wave VNA Frequency Extension Modules from Oleson Microwave Labs Extension Modules</li> </ul>					

## Measurement Receiver

Block Diagrams are available at the end of each [specifications](#) document.

PNA Model	Frequency Range	Ports	Connector Type	Reference Receivers	Front Panel jumpers
N5264A	IF Frequencies only	0	N/A	1	N/A
<a href="#">Learn more</a>					
<b>Options</b>					
108 - Built-in 26.5 GHz LO source with +10 dBm output power.					
118 - Fast-CW mode enables 500 million point data buffer.					

## Common Options and Upgrade Kits

The following options are installed at the time of purchase, and some are also available after the initial purchase of a PNA. To order an upgrade, contact your [Agilent representative](#).

Option	Supported Models	Description
006	E8356A E8801A N3381A	<b>Upgrade to 6 GHz frequency range. (No longer available)</b> Adds 6 GHz operation. Includes installation at an Agilent service center. Instrument calibration (required after frequency upgrade) is available for an additional fee.
009	E8356/7A E8801/2A N3381/2A	<b>Upgrade to 9 GHz frequency range.(No longer available)</b> Adds 9 GHz operation. Includes installation at an Agilent service center. Instrument calibration (required after frequency upgrade) is available for an additional fee.
010	All	<b>Time-domain</b> Adds time-domain capability to analyzer. The serial number of the analyzer must be specified when ordering this kit. Software upgrade. <a href="#">Learn more about Time Domain</a> <a href="#">Learn how this option is enabled.</a>
014	E8361A/C E8362B/C E8363B/C E8364B/C	<b>Configurable test set</b> Adds front panel access to the source output and coupler input on test ports 1 and 2. Adds front panel access directly to all receivers, including the reference receiver. Upgrade includes installation at an Agilent service center. All PNA-X models include front panel access.

015	E8356/ 57/ 58A	<b>Configurable test set (No longer available)</b> Adds front panel access to the source output and coupler input on test ports 1 and 2. Adds 35 dB step attenuators between the couplers and receivers. Upgrade includes installation at an Agilent service center.
016	E8361A/C E8362B/C E8363B/C E8364B/C	<b>Receiver Step Attenuators</b> Adds two step attenuators. (E8361A adds 50 dB step attenuators with 10 dB resolution; all other PNA models add 35 dB step attenuators with 5 dB resolution.) Each attenuator is inserted between a test port and its corresponding receiver. Requires option UNL.
022	E8361A/C E8362B/C E8363B/C E8364B/C	<b>Extended Memory</b> Adds more RAM for a total of 512MB. <a href="#">Learn more about PNA RAM</a>
080	All	<b>Frequency Offset</b> Enables you to set the PNA source independently from where the receivers are tuned. This capability is important for measuring mixers, converters, and amplifiers. To control the additional hardware, a basic user interface, SCPI, and COM commands are provided. Requires option 014 or Active test set (N5230A).
081	E8361A/C E8362B/C E8363B/C E8364B/C	<b>Reference Switch</b> Adds an internal solid state switch in the R1 reference path for controlling an external reference mixer. Requires option 014 and opt 080 See <a href="#">specifications</a> for a block diagram which includes the reference switch. All PNA-X models include this reference switch.
082	All	<b>Scalar Mixer Measurements</b> Allows Only the Scalar Mixer Converter (SMC) portion of the Frequency Converter Measurement Application. Provides the same intuitive user-interface, easy calibration, and external source control for making fixed and swept LO Scalar Mixer measurements. When used with a multiport PNA or <a href="#">external test set</a> , SMC is only available on PNA ports 1 and 2. Requires Opt 080. <a href="#">Learn more.</a>
083	E8361A/C E8362B/C E8363B/C E8364B/C  PNA-X	<b>Frequency Converter Measurement Application</b> Provides an intuitive user-interface for making extremely accurate conversion loss and absolute group delay measurements on mixers and converters. Exceptional amplitude and phase accuracy is achieved through two new calibration techniques: Scalar Mixer Calibration and Vector Mixer Calibration. The application also provides automatic control of all of Agilent's major signal sources. Requires option 014, 080, and 081.

		<a href="#">Learn more</a>
<b>084</b>	All PNA-X and C Models	<p><b>Embedded LO</b></p> <p>Using Opt 083, provides the ability to measure frequency converters that have an embedded LO.</p> <p><a href="#">Learn more</a></p>
<b>098</b>	E8356/ 57/ 58A	<p><b>CPU board upgrade (No longer available)</b></p> <p>Replaces the 266 MHz CPU board with a 500 MHz CPU board.</p> <p>Upgrade includes installation at an Agilent service center.</p> <p>Learn how to check your <a href="#">PNA CPU speed</a>.</p> <p>See also <a href="#">1.1 GHz CPU board (Dec. 2005)</a></p>
<b>1E1</b>	E8801/ 02/ 03A N3381/ 82/ 83A	<p><b>70 dB step attenuator (No longer available)</b></p> <p>Adds a single 70 dB step attenuator that is switched between the source and each output port to extend the output power to -85 dBm. <b>Note:</b> Two 70 dB step attenuators are standard equipment with PNA models E8356/ 57/ 58A.</p> <p>Upgrade includes installation at an Agilent service center.</p>
<b>1E5</b>	E8801/ 02/ 03A N3381/ 82/ 83A	<p><b>High stability 10 MHz time base (No longer available)</b></p> <p>Replaces 10 ppm time base with a 1 ppm time base. High stability time base is standard with PNA models E8356/ 57/ 58A and E8362/ 63/ 64B.</p>
<b>550</b>	E8361A/C E8362B/C E8363B/C E8364B/C N5230A/C	<p><b>Add full 4 port capability and differential measurements - <a href="#">Learn more</a>.</b></p> <p>Requires a PNA with configurable test set. <a href="#">E836x Opt 014</a> or <a href="#">N5230A Opt. x25</a></p>
<b>551</b>	All PNA-X and C Models	<p><b>Add fully integrated measurements at all of the available test ports.</b></p> <p>Solutions and PNA requirements depend on the supported test set. See <a href="http://www.agilent.com/find/multiport">http://www.agilent.com/find/multiport</a></p>
<b>897</b>	All PNA-X and C Models	<p>Perpetual license for built-in performance test software for <b>Agilent inclusive cal.</b></p> <p>Adds built-in performance testing and calibration software for self-maintainers. Requires additional equipment. See your PNA Service Guide for more information on equipment required.</p>
<b>898</b>	All PNA-X and C Models	<p>Perpetual license for built-in performance test software for <b>Standards compliant cal.</b></p> <p>Adds built-in performance testing and calibration software for self-maintainers. Requires additional equipment. See your PNA Service Guide for more information on equipment required.</p>

<b>H08</b>	All PNA-X and C Models	<p><b>Integrated Pulsed Measurement Setup</b></p> <p>Provides the Pulsed Application for average pulse and point-in-pulse measurements.</p> <p>PNA-X models use the <a href="#">Integrated Pulse Setup</a></p> <p>E836x models require options H11, UNL, 014, 080, and 081. <a href="#">Learn more</a></p> <p><a href="#">See PNA-X options for more information.</a></p>
<b>H11</b>	E8361A/C E8362B/C E8363B/C E8364B/C	<p><b>IF Access</b></p> <p>Provides rear panel access to the PNA IF paths for:</p> <ul style="list-style-type: none"> <li>• Extended frequency coverage to 325 GHz</li> <li>• Pulsed measurement capability</li> <li>• Advanced antenna measurements</li> </ul> <p>Requires options UNL, 014, 080, and 081.</p> <p><a href="#">Learn more</a></p>
<b>UNL</b>	E8361A/C E8362B/C E8363B/C E8364B/C	<p><b>Extended power range and bias tees</b></p> <p>Adds two step attenuators and two bias tees. (E8361A adds 50 dB step attenuators; E8362/ 63/ 64A/B adds 60 dB step attenuators. All attenuators have a 10 dB resolution.) A step attenuator and bias-tee set is inserted between the source and test port 1 and another set between the source and test port 2.</p> <p>Upgrade includes installation at an Agilent service center.</p>

## Certification Options

Option	Supported Models	Description
<b>UK6</b>	ALL except N5250A	Complete set of measurement data which was acquired from testing your PNA to published specifications. Includes calibration label, calibration certificate, and data report. Conforms to ISO 9001.
<b>1A7</b>	ALL except N5250A	Complete set of measurement data which was acquired from testing your PNA to published specifications. Includes calibration label, ISO 17025 calibration certificate, data report, measurement uncertainties and guard bands on all specifications. Conforms to ISO17025 and ISO 9001.

## Documentation and Localization Options

## Description

Printed versions of PNA Help in pdf format are available at [www.agilent.com/find/pna](http://www.agilent.com/find/pna). (Apr.2005)

A documentation CD-ROM is no longer included with each PNA shipment (Feb.2005).

To download a service guide for your PNA, or the latest version of PNA Help, visit [www.agilent.com/find/pna](http://www.agilent.com/find/pna), search for your PNA model, then click Library.

## PNA Warranty Period

The actual warranty on your instrument depends on the date it was ordered as well as whether or not any warranty options were purchased at that time. To determine the exact warranty on your instrument, contact [Agilent Technologies](#) with the model and serial number of your instrument.

For online information about Agilent's service and support products visit: [www.agilent.com/find/tm\\_services](http://www.agilent.com/find/tm_services).

---

Last modified:

3-Mar-2010	Added 128 and H29
17-Dec-2009	Corrected Certification Options
26-Oct-2009	Added 029 requirements
30-Apr-2009	Added new PNA-X models and PNA-X w/ Opts 081 and 014
8-Sep-2008	Added IMD
9-Jun-2008	Edits for C models and N5242A
25-Oct-2007	Added C models
5-Sep-2007	Added Embedded LO
9/26/06	MQ Added PNA-L 4port models
9/27/06	MX Added PNA-X

## Option Enable Utility

---

The Option Enable utility allows you to perform the following activities on your PNA.

- Enable or remove software options and some hardware options.
- Recover option data if the hard drive or other data-containing assembly is replaced.
- Input or change a serial number.

The following items are discussed in this topic:

### [Keywords](#)

### [Running the Program](#)

### [Removing an Option](#)

### [Installing an Option](#)

### [Repairing and Recovering Option Data](#)

### [Installing or Changing the Serial Number](#)

### See Also

[See PNA Configurations and Options](#)

## Keywords

To add certain options, you need a keyword that is provided by Agilent. There are two types of keywords:

- **Option Keywords** add a software option.
- **Model Keywords** may be required if you replace multiple assemblies.

Keywords are linked to the PNA **Host ID**, which is displayed on the Option Enable dialog box (below).

## Temporary and Permanent Options

Any software option can also be installed on a temporary basis for a specified amount of time. This allows you to evaluate a specific feature or capability at no cost.

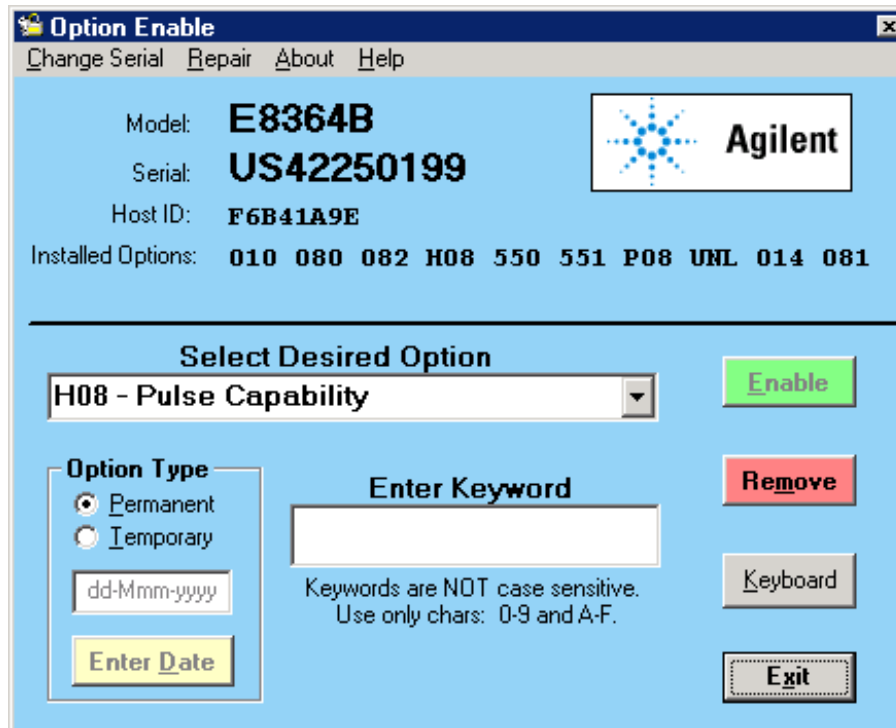
If the license key provided by Agilent has an expiration date, you must select the "temporary" option and enter the expiration date exactly as stated in the license statement. If you decide to make this option permanent, Agilent will provide a new keyword that converts the option to permanent status.

For either permanent or temporary software options, a provided keyword must be entered.

## Running the Program



On the PNA, click **System**, point to **Service**, then click **Option Enable**.



The image shows a software window titled "Option Enable" with a menu bar containing "Change Serial", "Repair", "About", and "Help". The window displays the following information:

- Model: **E8364B**
- Serial: **US42250199**
- Host ID: **F6B41A9E**
- Installed Options: **010 080 082 H08 550 551 P08 UNL 014 081**

On the right is the Agilent logo. Below this is a section titled "Select Desired Option" with a drop-down menu currently showing "H08 - Pulse Capability". To the right of the drop-down is a green "Enable" button. Below the drop-down is a section for "Option Type" with two radio buttons: "Permanent" (selected) and "Temporary". Below these is a date input field labeled "dd-Mmm-yyyy" and a yellow "Enter Date" button. To the right of the "Option Type" section is a text input field labeled "Enter Keyword". Below this field is a note: "Keywords are NOT case sensitive. Use only chars: 0-9 and A-F." To the right of the keyword field are three buttons: a red "Remove" button, a grey "Keyboard" button, and a grey "Exit" button.

1. To enable or remove an option, select it from the drop-down list of available options. If the desired option is not listed, select the last choice in the list, labeled **Enter Unlisted Option**.
2. Enter the 3-character option name and click **Enter**

If a software option was chosen, the following occurs.

- The **Remove** button will be enabled.
- The keyword entry area becomes visible.
- The permanent/temporary selection is enabled.

If a hardware option is selected, the following occurs.

- With the hardware option already installed, the **Remove** button is enabled.
- With the hardware option not installed, the **Enable** button is enabled.

## Removing an Option

1. To remove an option, click **Remove**.
2. After the option is removed, restart the network analyzer application for the changes to take effect.

**Note:** Removal of a licensed option (such as Option 010, Time Domain) will permanently remove the license

keyword. If this option **may** be needed in the future, then record the license keyword before removing the option. Do this by copying the file “gen.lic” to another location (such as a floppy disk), or print it using notepad. The file, located at “C:/Program Files/Agilent/Network Analyzer” contains all the information needed to recreate the license.

## Installing an Option

1. If the keyword entry area is visible, enter a keyword. (The keyword is not case sensitive.)
2. Click **Enable**.
3. After the option is installed, restart the network analyzer application for the changes to take effect.

**Note:** If a desired option is not visible, it may be because a prerequisite option has not yet been installed. For example, Option 083 will not be visible if Option 080 is not already present. [See PNA options.](#)

## Repairing and Recovering Option Data

Use this part of the Option Enable Utility in the following situations:

- If the hard drive is replaced
- If the frequency reference assembly is replaced

This routine rebuilds the option information contained on the hard drive and frequency reference assembly (primary and backup).

1. Select **Repair** from the **Option Enable** menu bar.

**Note:** If you are unsure if this routine needs to be done, run it; no harm will result.

2. The model and serial number are displayed, along with four check boxes.
3. Select the boxes that apply.
4. Click **Begin Repair**. The routine checks all data files and performs any needed repairs. You may be asked to verify certain information and processes.
5. If the routine finds that the model number is incorrect or invalid, you will be asked to select the correct model number.
  - Along with this model number, a model keyword will be required. If this is not labeled on the analyzer, or is not otherwise known, contact Agilent
  - After you have entered the requested data, click **Change Model**. This process takes about 30 seconds.
6. When done, click **Exit Repair**.
7. If you do not need to install any other options, click **Exit**.

## Installing or Changing the Serial Number

It may be necessary to install or change a serial number if certain assemblies are replaced.

1. To change the serial number, select **Change Serial** from the **Option Enable** menu bar. The current serial number will be displayed. If no serial number has previously been entered, the word "NONE" will be displayed.
2. Type the new serial number into the space provided, and click **Change Serial**. (The serial number is not case sensitive.)

**Note:** Use extreme care when entering the serial number; only one entry chance is allowed!

3. To change an incorrect serial number, a clear-code password is required. Contact Agilent to obtain this clear code and have the existing serial number available. Enter the clear code in the space provided, along with the new serial number, then click **Change Serial**.

---

Last Modified:

20-Sep-2007    Added Install note

## Instrument Calibration

---

An instrument calibration is a process where the PNA performance is measured to ensure that it operates within specifications. If any performance parameter does not conform to the published specifications, adjustments are made to bring the performance into conformance.

### Why Should I Get an Instrument Calibrated?

Over time, the active components in the analyzer age and the performance may degrade or drift.

To ensure that the analyzer is performing to the published specifications, you should have an instrument calibration performed periodically.

### How Often Should I Get an Instrument Calibrated?

It is your responsibility to determine the calibration period which best meets your requirements. However, a 12 to 18 month calibration cycle is appropriate for most users.

There are two things to consider: performance drift and connector wear.

- The instrument specifications are set to consider the performance drift that may occur over a 24 month period. Therefore, getting the instrument calibrated at 24 month intervals ensures that the analyzer maintains performance within the operating specifications. If you need the analyzer to maintain more consistent operation, you may want to have the instrument calibrated more often than the recommended 24-month interval.
- Connector wear is a bigger factor and depends on the number of connections that are made. The test ports become noticeably worn after 500 to 700 connections. This could represent about 12 months with average use. With more frequent connections, the calibration cycle should be sooner. You can extend the time between calibrations and thereby save money by using [connector savers](#) and by performing proper [Connector Care](#).

### How Do I Get an Instrument Calibrated?

To get the instrument calibrated, send it to one of the Agilent Technologies service centers. See [Technical Support](#).

The PNA must be fully functional when it is sent to the service center, or they will charge for their repair services. If the PNA is being used in a secure environment where the hard drive can not be sent with the PNA, a second hard drive must be purchased and configured for use with the PNA in an "unclassified" environment before the PNA is sent to the service center.

To perform the instrument calibration yourself, you must have the following required items:

- Instrument Calibration Test Equipment
- Performance Test Software

### What Are My Choices of Instrument Calibration?

The following types of instrument calibration are available from Agilent Technologies at the time of initial order:

<b>Standard</b>	Includes a certificate of calibration stating the instrument has been calibrated and is operating within the published specifications.
<b>Option UK6</b>	Available ONLY at the initial shipment. Includes the test data from the calibration and the certificate of calibration stating the instrument has been calibrated and is operating within the published specifications.
<b>Option A6J</b>	Available ONLY at the initial shipment. Includes the test data and measurement uncertainties from the calibration and the certificate of calibration stating the instrument has been calibrated using a process in compliance with ANSI Z540 and is operating within the published specifications.
<b>Option 1A7</b>	Available ONLY at the initial shipment. Includes the test data and measurement uncertainties from the calibration and the certificate of calibration stating the instrument has been calibrated using a process in compliance with ISO 17025 and is operating within the published specifications.

The following types of instrument calibration are available from Agilent Technologies service center:

<b>Agilent Calibration</b>	Includes the test data from the calibration and the certificate of calibration, stating the instrument has been calibrated and is operating within the published specifications.
<b>ANSI Z540 Calibration</b>	Includes the test data from the calibration and the certificate of calibration, stating the instrument has been calibrated using a process in compliance with ANSI Z540.1 and is operating within the published specifications.
<b>ISO 17025 Calibration</b>	Includes the test data from the calibration and the certificate of calibration, stating the instrument has been calibrated using a process in compliance with ISO 17025 and is operating within the published specifications.

For more information on these options, visit [www.agilent.com/find/calibration](http://www.agilent.com/find/calibration).

Last Modified:

24-Mar-2010      Updated options and cal cycle

## Other Resources

---

The following network analysis resources are also available.

### Document Resources

#### Application Notes

You can also access application notes at this URL:

<http://www.agilent.com/find/PNA>

### Third-Party Resources

For information about test fixtures and part handlers, contact:

**Inter-Continental Microwave**

[www.icmicrowave.com](http://www.icmicrowave.com)

For information about probing equipment and accessories, contact:

**Cascade Microtech, Inc.**

[www.cascademicrotech.com](http://www.cascademicrotech.com)

## SCPI Errors

---

### SCPI Errors

- [-100 to -200 Command Errors](#)
- [-200 to -299 Execution Errors](#)
- [-300 to -399 SCPI Specified Device-Specific Errors](#)
- [-400 to -800 Query and System Errors](#)
- [100 to 200 PNA-specific Errors](#)

### See Also

[PNA Error messages.](#)

---

#### **-100 to -200 Command Errors**

A command error indicates that the test set's GPIB parser has detected an IEEE 488.2 syntax error. When one of these errors is generated, the command error bit in the event status register is set.

-100	std_command	Command - This event bit (Bit 5) indicates a syntax error, or a semantic error, or a GET command was entered, see IEEE 488.2, 11.5.1.1.4.
-101	std_invalidChar	Invalid character - Indicates a syntactic elements contains a character which is invalid for that type.
-102	std_syntax	Syntax - Indicates that an unrecognized command or data type was encountered. For example, a string was received when the device does not accept strings.
-103	std_invalidSeparator	Invalid separator - The parser was expecting a separator and encountered an illegal character. For example, the semicolon was omitted after a program message unit.
-104	std_wrongParamType	Data type -The parser recognized a data element different than one allowed. For example, numeric or string data was expected but block data was encountered.
-105	std_GETNotAllowed	GET not allowed - Indicates a Group Execute Trigger was received within a program message. Correct the program so that the GET does not occur within the program code.
-108	std_tooManyParameters	Parameter not allowed - Indicates that more parameters were received than expected for the header. For example, *ESE common command only accepts one parameter, so *ESE 0,1 is not allowed.
-109	std_tooFewParameters	Missing parameter - Indicates that less parameters were received than required for the header. For example, *ESE requires one parameter, *ESE is not allowed.

-110	std_cmdHeader	Command header - Indicates an error was detected in the header. This error is used when the device cannot detect the more specific errors -111 through -119.
-111	std_headerSeparator	Header separator - Indicates that a character that is not a legal header separator was encountered while parsing the header.
-112	std_IDTooLong	Program mnemonic too long - Indicates that the header contains more than twelve characters, see IEEE 488.2, 7.6.1.4.1.
-113	std_undefinedHeader	Undefined header - Indicates the header is syntactically correct, but it is undefined for this specific device. For example, *XYZ is not defined for any device.
-114	std_suffixOutOfRange	Header suffix out of range - Indicates the value of a header suffix attached to a program mnemonic makes the header invalid.
-120	std_numericData	Numeric data - This error, as well as errors
-121	std_invalidCharInNumber	Invalid character in number - Indicates an invalid character for the data type being parsed was encountered. For example, an alpha in a decimal numeric or a "9" in octal data.
-123	std_exponentTooLarge	Exponent too large - Indicates the magnitude of an exponent was greater than 32000, see IEEE 488.2, 7.7.2.4.1.
-124	std_decimalTooLong	Too many digits - Indicates the mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros, see IEEE 488.2, 7.7.2.4.1.
-128	std_numericNotAllowed	Numeric data not allowed - Indicates that a legal numeric data element was received, but the device does not accept one in this position for the header.
-130	std_suffix	Suffix - This error, as well as errors -131 through -139, are generated when parsing a suffix. This particular error message is used if the device cannot detect a more specific error.
-131	std_badSuffix	Invalid suffix - Indicates the suffix does not follow the syntax described in IEEE 488.2, 7.7.3.2, or the suffix is inappropriate for this device.
-134	std_suffixTooLong	Suffix too long - Indicates the suffix contains more than 12 characters, see IEEE 488.2, 7.7.3.4.
-138	std_suffixNotAllowed	Suffix not allowed - Indicates that a suffix was encountered after a numeric element that does not allow suffixes.
-140	std_charData	Character data - This error, as well as errors
-141	std_invalidCharData	Invalid character data - Indicates that the character data element contains an invalid character or the particular element received is not valid for the header.
-144	std_charDataTooLong	Character data too long - Indicates the character data element contains more than twelve characters, see IEEE 488.2, 7.7.1.4.



-148	std_charNotAllowed	Character data not allowed - Indicates a legal character data element was encountered where prohibited by the device.
-150	std_stringData	String data - This error, as well as errors
-151	std_stringInvalid	Invalid string data - Indicates that a string data element was expected, but was invalid, see IEEE 488.2, 7.7.5.2. For example, an END message was received before the terminal quote character.
-158	std_stringNotAllowed	String data not allowed - Indicates that a string data element was encountered but was not allowed by the device at this point in parsing.
-160	std_blockData	Block data - This error, as well as errors -161 through -169, are generated when parsing a block data element. This particular error message is used if the device cannot detect a more specific error.
-161	std_badBlock	Invalid block data - Indicates a block data element was expected, but was invalid, see IEEE 488.2, 7.7.6.2. For example, an END message was received before the end length was satisfied.
-168	std_blockNotAllowed	Block data not allowed - Indicates a legal block data element was encountered, but not allowed by the device at this point in parsing.
-170	std_expr	Expression - This error, as well as errors -171 through -179, are generated when parsing an expression data element. This particular error message is used if the device cannot detect a more specific error.
-171	std_invalidExpression	Invalid expression - Indicates the expression data element was invalid, see IEEE 488.2, 7.7.7.2. For example, unmatched parentheses or an illegal character.
-178	std_exprNotAllowed	Expression data not allowed - Indicates a legal expression data was encountered, but was not allowed by the device at this point in parsing.
-180	std_macro	Macro - This error, as well as error -181 through -189, are generated when defining a macro or execution a macro. This particular error message is used if the device cannot detect a more specific error.
-181	std_validOnlyInsideMacro	Invalid outside macro definition - Indicates that a macro parameter place holder was encountered outside of a macro definition.
-183	std_invalidWithinMacro	Invalid inside macro definition - Indicates that the program message unit sequence, sent with a *DDT or a *DMC command, is syntactically invalid, see IEEE 488.2, 10.7.6.3.
-184	std_macroParm	Macro parameter - Indicates that a command inside the macro definition had the wrong number or type of parameters.

#### **-200 to -299 Execution Errors**

These errors are generated when something occurs that is incorrect in the current state of the instrument. These errors may be generated by a user action from either the remote or the manual user interface

-200	std_execGen	Execution - This event bit (Bit 4) indicates a PROGRAM DATA element following a header was outside the legal input range or otherwise inconsistent with the device's capabilities, see IEEE 488.2, 11.5.1.1.5.
-201	std_invalidWhileInLocal	Invalid while in local
-202	std_settingsLost	Settings lost due to rtl
-203	std_commandProtected	Command protected - Indicates that a legal password-protected program command or query could not be executed because the command was disabled.
-210	std_trigger	Trigger
-211	std_triggerIgnored	Trigger ignored
-212	std_armIgnored	Arm ignored
-213	std_initIgnored	Init ignored
-214	std_triggerDeadlock	Trigger deadlock
-215	std_armDeadlock	Arm deadlock
-220	std_parm	Parameter - Indicates that a program data element related error occurred.
-221	std_settingsConflict	Settings conflict - Indicates that a legal program data element was parsed but could not be executed due to the current device state.
-222	std_dataOutOfRange	Data out of range - Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range defined by the devices
-223	std_tooMuchData	Too much data - Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.
-224	std_illegalParmValue	Illegal parameter value - Indicates that the value selected was not part of the list of values given.
-225	std_noMemoryForOp	Out of memory - The device has insufficient memory to perform the requested operation.
-226	std_listLength	Lists not same length - Attempted to use LIST structure having individual LIST's of unequal lengths.
-230	std_dataCorruptOrStale	Data corrupt or stale - Indicates invalid data, a new reading started but not completed since the last access.
-231	std_dataQuestionable	Data questionable - Indicates that measurement accuracy is suspect.

-232	std_invalidFormat	Invalid format
-233	std_invalidVersion	Invalid version - Indicates that a legal program data element was parsed but could not be executed because the version of the data is incorrect to the device. For example, a not supported file version, a not supported instrument version.
-240	std_hardware	Hardware - Indicates that a legal program command or query could not be executed because of a hardware problem in the device.
-241	std_hardwareMissing	Hardware missing - Indicates that a legal program command or query could not be executed because of missing device hardware. For example, an option was not installed.
-250	std_massStorage	Mass storage - Indicates that a mass storage error occurred. The device cannot detect the more specific errors described for errors -251 through -259.
-251	std_missingMassStorage	Missing mass storage - Indicates that a legal program command or query could not be executed because of missing mass storage.
-252	std_missingMedia	Missing media - Indicates that a legal program command or query could not be executed because of missing media. For example, no disk.
-253	std_corruptMedia	Corrupt media - Indicates that a legal program command or query could not be executed because of corrupt media. For example, bad disk or wrong format.
-254	std_mediaFull	Media full- Indicates that a legal program command or query could not be executed because the media is full. For example, there is no room left on the disk.
-255	std_directoryFull	Directory full - Indicates that a legal program command or query could not be executed because the media directory was full.
-256	std_fileNotFound	File name not found - Indicates that a legal program command or query could not be executed because the file name was not found on the media.
-257	std_fileName	File name - Indicates that a legal program command or query could not be executed because the file name on the device media was in error. For example, an attempt was made to read or copy a nonexistent file.
-258	std_mediaProtected	Media protected - Indicates that a legal program command or query could not be executed because the media was protected. For example, the write-protect switch on a memory card was set.
-260	std_expression	Expression
-261	std_math	Math in expression
-270	std_macroExecution	Macro - Indicates that a macro related execution error occurred.

-271	std_macroSyntax	Macro syntax - Indicates that a syntactically legal macro program data sequence, according to IEEE 488.2, 10.7.2, could not be executed due to a syntax error within the macro definition.
-272	std_macroExec	Macro execution - Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition, see IEEE 488.2, 10.7.6.3.
-273	std_badMacroName	Illegal macro label - Indicates that the macro label was not accepted, it did not agree with the definition in IEEE 488.2, 10.7.3
-274	std_macroPlaceholder Ma	cro parameter - Indicates that the macro definition improperly used a macro parameter placeholder, see IEEE 488.2, 10.7.3.
-275	std_macroTooLong	Macro definition too long - Indicates that a syntactically legal macro program data sequence could not be executed because the string of block contents were too long for the device to handle, IEEE 488.2, 10.7.6.1.
-276	std_macroRecursion	Macro recursion - Indicates that a syntactically legal macro program data sequence count not be executed because it would be recursive, see IEEE 488.2, 10.7.6.6.
-277	std_cantRedefineMacro	Macro redefinition not allowed - Indicates that redefining an existing macro label, see IEEE 488.2, 10.7.6.4.
-278	std_macroNotFound	Macro header not found - Indicates that a legal macro label in the *GMS?, see IEEE 488.2, 10.13, could not be executed because the header was not previously defined.
-280	std_program	Program
-281	std_cantCreateProgram	Cannot create program
-282	std_illegalProgramName	Illegal program name
-283	std_illegalVarName	Illegal variable name
-284	std_programRunning	Program currently running
-285	std_programSyntax	Program syntax
-286	std_programRuntime	Program runtime
-290	std_memoryUse	Memory use
-291	std_execOutOfMemory	Out of memory
-292	std_nameNotFound	Referenced name does not exist

- 293 std\_nameAlreadyExists      Referenced name already exists
- 294 std\_incompatibleType      Incompatible type

### **-300 to -399 SCPI Specified Device-Specific Errors**

A device-specific error indicates that the instrument has detected an error that occurred because some operations did not properly complete, possibly due to an abnormal hardware or firmware condition. For example, an attempt by the user to set an out of range value will generate a device specific error. When one of these errors is generated, the device specific error bit in the event status register is set.

- 300 std\_deviceSpecific      Device specific - This event bit (Bit 3) indicates that a device operation did not properly complete due to some condition, such as overrange see IEEE 488.2, 11.5.1.1.6.
- 310 std\_system      System
- 311 std\_memory      Memory - Indicates some physical fault in the devices memory, such as a parity error.
- 312 std\_PUDmemoryLost      PUD memory lost - Indicates protected user data saved by the \*PUD command has been lost, see IEEE 488.2, 10.27.
- 313 std\_calMemoryLost      Calibration memory lost - Indicates that nonvolatile calibration data used by the \*CAL? command has been lost, see IEEE 488.2, 10.2.
- 314 std\_savRclMemoryLost      Save/recall memory lost - Indicates that the nonvolatile data saved by the \*SAV command has been lost, see IEEE 488.2, 10.33.
- 315 std\_configMemoryLost      Configuration memory lost - Indicates that nonvolatile configuration data saved by the device has been lost.
- 320 std\_storageFault      Storage fault - Indicates that the firmware detected a fault when using data storage. This is not an indication of physical damage or failure of any mass storage element.
- 321 std\_outOfMemory      Out of memory - An internal operation needed more memory than was available
- 330 std\_selfTestFailed      Self-test failed - Indicates a problem with the device that is not covered by a specific error message. The device may require service.
- 340 std\_calFailed      Calibration failed - Indicates a problem during calibration of the device that is not covered by a specific error.
- 350 std\_queueOverflow      Queue overflow - Indicates that there is no room in the queue and an error occurred but was not recorded. This code is entered into the queue in lieu of the code that caused the error.
- 360 std\_comm      Communication - This is the generic communication error for devices that cannot detect the more specific errors described for error -361 through -363.
- 361 std\_parity      Parity in program message - Parity bit not correct when data received for example, on a serial port.

- 362 std\_framing                      Framing in program message - A stop bit was not detected when data was received for example, on a serial port (for example, a baud rate mismatch).
- 363 std\_inputBufferOverflow      Input buffer overrun - Software or hardware input buffer on serial port overflows with data caused by improper or nonexistent pacing.

### **-400 to -800 Query and System Errors**

A Query error is generated either when data in the instrument's GPIB output queue has been lost, or when an attempt is being made to read data from the output queue when no output is present or pending.

- 400 std\_queryGen                      Query - This event bit (Bit 2) indicates that an attempt to read data from the Output Queues when no output is present or pending, to data in the Output Queue has been lost see IEEE488.2, 11.5.1.1.7.
- 410 std\_interrupted                  Query INTERRUPTED - Indicates the test set has been interrupted by a new program message before it finishes sending a RESPONSE MESSAGE see IEEE 488.2, 6.3.2.3.
- 420 std\_unterminated                Query UNTERMINATED - Indicates an incomplete Query in the program see IEEE 488.2, 6.3.2.2.
- 430 std\_deadlocked                  Query DEADLOCKED - Indicates that the Input Buffer and Output Queue are full see IEEE 488.2, 6.3.1.7.
- 440 std\_responseNotAllowed      Query UNTERMINATED after indefinite response - Indicates that a query was received in the same program message after a query requesting an indefinite response was executed see IEEE 488.2, 6.5.7.5.
- 500 std\_powerOn                      Power on
- 600 std\_userRequest                  User request
- 700 std\_requestControl              Request control
- 800 std\_operationComplete          Operation complete

.

### **PNA Specific (Positive) SCPI Errors**

- 100 dupWindNum                      "Duplicate window number"
- 101 windNumNotFound                "Window number not found"
- 102 failedWindCreate                "Window creation failed"
- 103 noCalcParamSelection            "CALC measurement selection set to none"  
[See CALC:PAR:SEL](#)
- 104 dupMeasName                      "Duplicate measurement name"

105 dataNotFound	"Requested data not available"
106 measNotFound	"Requested measurement not found"
107 traceNotFound	"Requested trace not found"
108 notImplemented	"Mnemonic not yet implemented"
109 noDocument	"No measurement container found"
110 dupTraceNum	"Duplicate trace number"
111 titleStrTooLong	"Title string exceeds 50 characters"
112 memoryNotFound	"Requested memory not found"
113 exceedMaxTraces	"Exceeded the maximum number of traces per window"
114 SerNumNotFound	"The serial number was not found. Please store the serial number."
115 LoadFailed	"The state was not loaded. Please check the file name."
116 StoreFailed	"The state was not stored. Please check the file and path names."
117 File	"An in the File operation occurred. Please check file and path names."
118 measChanConflict	"Measurement does not belong to specified channel."
119 exceedMaxWindows	"Exceeded the maximum number of data windows"
120 markerNotFound	"The specified marker was not found."
121 diagnostic	"Diagnostic ."
122 channelNotFound	"The specified channel was not found."
123 exceedMaxMeasurements	"Exceeded the maximum number of allowed measurements."
124 parameterOutOfRange	"The specified value was out of range."
125 userRangeNotValid	"The currently selected user range is not valid."
126 referenceMarkerNotFound	"The reference marker is not active."
127 sweepSegmentNotFound	"The sweep segment was not found."
128 markerNotDelta	"The specified marker is not a delta marker."
129 printoutFailed	"Attempt to output to a printer failed."
130 memory_trace_not_compatible	"Memory not compatible. Trace Math not applied."

131 trace_math_reset	"Memory not compatible. Trace Math turned off."
132 hw_read_failed	"Hardware read failed."
133 hw_write_failed	"Hardware write failed."
134 dsp_active	"Failed because DSP was not halted."
135 secure_memory	"Attempt to access secure memory region."
136 snum_protected	"The serial number is protected."
137 snum_format_bad	"The serial number format is bad."
138 snum_already_set	"The serial number is already set."
139 hw_setting_failed	"Hardware setting failed."
140 cal_access_failed	"Calibration data access failed."
141 db_access_failed	"Database access failed."
142 memory_range_exceeded	"Command exceeds usable memory range."
143 lost_phase_lock	"Phase lock has been lost."
144 over_power	"Detected too much power at input."
145 ee_wrt_failed	"EEPROM write failed."
146 yig_cal_failed	"YTO calibration failed."
147 ramp_cal_failed	"Analog ramp calibration failed."
148 dspcom_bad	"DSP communication failed."
149 no_license_found	"Request failed. The required license was not found."
150 argLimited	"The argument was out of range"
151 markerBWNotFound	"The Marker Bandwidth was not found."
153 peakNotFound	"The Peak was not found."
154 targetNotFound	"The Target search value was not found."
155 calNotImpl	"The Calibration feature requested is not implemented."
156 calClassNotValidForCalType	"SENS:CORR:CCH measurement selection set to none"



158	calNotValidForConfidenceChe	"Selected measurement does not have a calibration valid for Confidence Check"
159	invalidPort	"Specified port is out of range"
160	invalidPortPath	"ROUT:PATH:DEF:PORT x, y does not match measurement; setting to defaults"
161	iolInvalidWrite	"Attempted I/O write while port set to read only."
162	iolInvalidRead	"Attempted I/O read from write only port."
163	calsetNotFound	"Requested Cal Set was not found in Cal Set Storage."
164	noCalSetSelected	"There is no Cal Set currently selected for the specified channel."
165	cantDeleteCalSetInUse	"Cannot delete a Cal Set while it is being used."
166	calsetStimChange	"Channel stimulus settings changed to match selected Cal Set."
167	exceedMaxCalSets	"Exceeded the maximum number of cal sets."
168	calCouldNotTurnOn	"A valid calibration is required before correction can be turned on."
169	standardMeasurementRequired	"The attempted operation can only be performed on a standard measurement type."
170	noDivisorBuffer	"A valid divisor buffer is required before normalization can be turned on."
171	InvalidReceiverPowerCalParagraph	"Receiver power cal requires the measurement to be of unratioed power."
172	ecalCouldNotConfigure	"Could not configure the Electronic Calibration system. Check to see if the module is plugged into the proper connector."
173	measHasNoMemoryAlg	"This measurement does not support memory operations"
174	measHasNoNormalizeAlg	"This measurement does not support normalize operations."
175	userCharacterizationNotFound	"User characterization was not found in the Electronic Calibration module."
176	measInvalidBufferSize	"The data provided has an invalid number of points. It could not be stored."

---

Last Modified:

4-Aug-2009    Cosmetic mods

## Technical Support

---

Click on the region of interest.



For more contact information, visit <http://www.agilent.com/find/contactus>

### Other Support Topics

#### **United States:**

(tel) (+1) 800-829-4444

(alt) (+1) 303 662 3999

(fax) (+1) 888 900 8921

#### **Canada**

(tel) 1 877 894 4414

(fax) 1 (905) 206 4120

#### **Europe:**

##### **Austria**

(tel) 0820 87 44 11\*

(fax) 0820 87 44 22

##### **Belgium**

(tel) (+32) (0)2 404 9340

(alt) (+32) (0)2 404 9000

(fax) (+32) (0)2 404 9395

## **Denmark**

(tel) (+45) 7013 1515  
(alt) (+45) 7013 7313  
(fax) (+45) 7013 1555

## **Finland**

(tel) 08 0052 4000  
(alt) (+358) 10 855 2100  
(fax) (+358) 92 536 0176

## **France**

(tel) 0825 010 700\*  
(alt) (+33) (0)1 6453 5623  
(fax) 0825 010 701\*

## **Germany**

(tel) 01805 24 6333\*  
(alt) 01805 24 6330\*  
(fax) 01805 24 6336\*

## **Ireland**

(tel) (+353) (0)1 890 924 204  
(alt) (+353) (0)1 890 924 206  
(fax) (+353) (0)1 890 924 024

## **Israel**

(tel) (+972) 3 9288 500  
(fax) (+972) 3 9288 501

## **Italy**

(tel) (+39) (0)2 9260 8484  
(fax) (+39) (0)2 9544 1175

## **Luxemburg**

(tel) (+32) (0)2 404 9340  
(alt) (+32) (0)2 404 9000  
(fax) (+32) (0)2 404 9395

## **Netherlands**

(tel) (+31) (0)20 547 2111

(alt) (+31) (0)20 547 2000  
(fax) (+31) (0)20 547 2190

### **Russia**

(tel) (+7) 095 797 3963  
(alt) (+7) 095 797 3900  
(fax) (+7) 095 797 3901

### **Spain**

(tel) (+34) 91 631 3300  
(alt) (+34) 91 631 3000  
(fax) (+34) 91 631 3301

### **Sweden**

(tel) 0200 88 22 55\*  
(alt) (+46) (0)8 5064 8686  
(fax) 020 120 2266\*

### **Switzerland (French)**

(tel) 0800 80 5353 opt. 2\*  
(alt) (+33) (0)1 6453 5623  
(fax) (+41) (0)22 567 5313

### **Switzerland (German)**

(tel) 0800 80 5353 opt. 1\*  
(alt) (+49) (0)7031 464 6333  
(fax) (+41) (0)1 272 7373

### **Switzerland (Italian)**

(tel) 0800 80 5353 opt. 3\*  
(alt) (+39) (0)2 9260 8484  
(fax) (+41) (0)22 567 5314

### **United Kingdom**

(tel) (+44) (0)7004 666666  
(alt) (+44) (0)7004 123123  
(fax) (+44) (0)7004 444555

### **Japan:**

(tel) 0120 421 345

(alt) (+81) 426 56 7832

(fax) 0120 421 678

## **Latin America:**

### **Mexico**

(tel) (+52) 55 5081 9469

(alt) 01800 5064 800

(fax) (+52) 55 5081 9467

### **Brazil**

(tel) (+55) 11 4197 3600

(fax) (+55) 11 4197 3800

## **Australia:**

((tel) 1800 629 485

(alt) 1800 143 243

(fax) 1800 142 134

## **New Zealand**

(tel) 0 800 738 378

(fax) 64 4 495 8950

## **Asia Pacific:**

### **China**

(tel) 800 810 0189

(alt) (+86) 10800 650 0021

(fax) 800 820 2816

### **Hong Kong**

(tel) 800 930 871

(alt) (+852) 3197 7889

(fax) (+852) 2 506 9233

### **India**

(tel) 1600 112 929

(fax) 000800 650 1101

**Malaysia**

(tel) 1800 888 848

(alt) 1800 828 848

(fax) 1800 801 664

**Singapore**

(tel) 1800 375 8100

(fax) (+65) 6836 0252

**South Korea**

(tel) 080 769 0800

(alt) (+82) 2 2004 5004

(fax) (+82) 2 2004 5115

**Taiwan**

(tel) 0800 047 866

(alt) 00801 651 317

(fax) 0800 286 331

**Thailand**

(tel) 1800 226 008

(alt) (+66) 2 268 1345

(fax) (+66) 2 661 3714

---

Last Modified:

1-Dec-2008    Changed US number

## 3.8 GHz Frequency Adjustment

---

This routine adjusts the internal fixed-frequency YIG Oscillator to 3.8 GHz by changing a DAC value. This DAC value is stored in the analyzer's non-volatile memory. This adjustment is only needed on some PNA models.

Typically, the oscillator can be set to within 12 kHz of 3.8GHz; it is not necessary for it to be exactly 3.8GHz.

### Spectrum Analyzers Compatibility

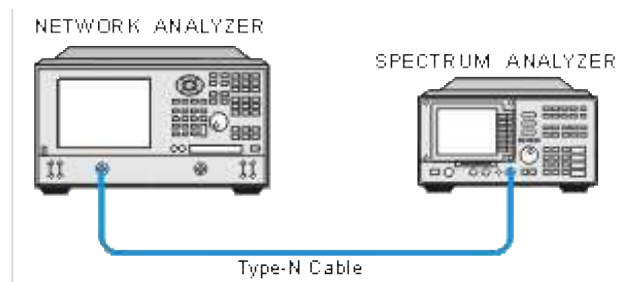
This routine is compatible with Agilent 856x and 859x spectrum analyzers, and may also work on some other Agilent spectrum analyzers.

If no compatible analyzer is available, select "NONE" for the spectrum analyzer. You can then adjust the DAC manually by viewing the 3.8 GHz signal on another analyzer.

### Procedure (For Compatible Spectrum Analyzers Only)

**Note:** The viewable 3.8 GHz signal level will be low; typically be around -70dBm. Do not use any attenuators in the adjustment, other than the default 10 dB attenuation used in most spectrum analyzers.

1. Connect spectrum analyzer input to the network analyzer's PORT 1 output.
2. Connect GPIB cable from analyzer to spectrum analyzer. Make sure no other controllers are active on the same connection.



3. Set the spectrum analyzer GPIB address to 18.
4. In the analyzer **System** menu, point to **Service, Adjustments**, and click **3.8 GHz Freq. Adjust**.
5. Click **Begin Adj.** for the program to adjust the internal oscillator for minimal error and store the results. When the status area indicates the adjustment is complete, select **Exit**.

### Procedure (For Non-Compatible Spectrum Analyzers Only)

**Note:** The viewable 3.8 GHz signal level will be low; typically be around -70dBm. Do not use any attenuators in the adjustment, other than the default 10 dB attenuation used in most spectrum analyzers.

1. Connect the spectrum analyzer input to the network analyzer's PORT 1 output.

2. Set the spectrum analyzer to the following settings:
  - Center frequency=3.8 GHz
  - Span= 100 MHz
  - Bandwidth= 10 kHz
  - Scaling where a signal of -70 dBm will be clearly visible
3. In the analyzer **System** menu, point to **Service, Adjustments**, and click **3.8 GHz Freq. Adjust**.
4. Under **Spectrum Analyzer**, select **NONE** option for spectrum analyzer.
5. Click **Begin Adj.**
6. The application presets the DAC to an initial value equal to the current value stored. View the spectrum analyzer to see if the signal is above or below 3.800 GHz.
  - If the signal frequency is above 3.8 GHz, move the slider to adjust the DAC to a lower value (left).
  - If the frequency is below 3.8 GHz, move the slider to adjust the DAC to a higher value (right).

**Note:** The valid DAC values are from 0 to 4095. The oscillator will shift about 23 kHz per DAC value.

7. Set the DAC value to reach a frequency very close to 3.8 GHz. If you made large changes in DAC values, allow several seconds for thermal effects to stabilize.
8. Change the spectrum analyzer settings to better view the frequency signal:
  - Frequency span = 500 kHz
  - Bandwidth = 3 kHz
9. Change the DAC value to keep the signal centered at 3.8 GHz.
10. Once you have determined the correct DAC value, click **SAVE DAC** to permanently store that value into EEPROM. Click **Exit**.

**Note:** If large changes are made to the existing DAC value, then this test should be repeated again after 15-30 minutes. This allows the thermal effects to fully stabilize.



## 10 MHz Reference Frequency Adjustment

---

This routine adjusts the analyzer's internal time-base to exactly 10 MHz by changing a DAC value. This DAC value is stored in the analyzer's non-volatile memory. This routine should only be necessary in the following situations:

- The frequency reference assembly is replaced.
- The 10 MHz reference has drifted significantly from the factory adjusted value.

**WARNING:** The range of this adjustment is only about 20 Hz. It is highly recommended that a very accurate frequency standard be used to measure this 10 MHz signal.

### Frequency Counter Compatibility

This procedure uses SCPI commands (over GPIB) to communicate with the frequency counter. It should work with the Agilent 5313xA, 5315xA, 53181A series of counters as well as the older 5350 series.

If no compatible counters are available, select the "Manual" mode of operation. If you do choose the manual mode, you must input the measured frequency manually.

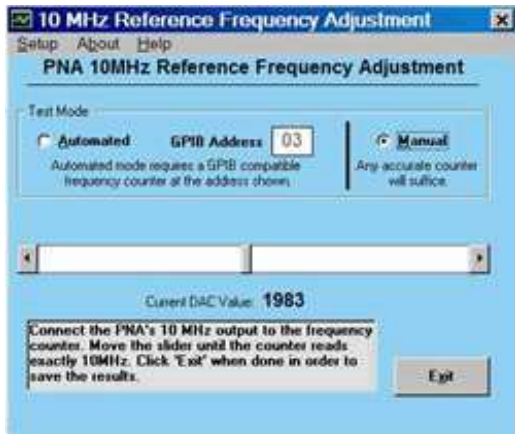
### Procedure for GPIB Counters Only

1. Connect the analyzer rear panel 10 MHz Reference output to the frequency counter .
2. Connect a GPIB cable from the analyzer to the counter. Make sure no other controllers are active on the same connection.
3. If applicable, connect the house frequency standard to the counter reference input.
4. Set the counter GPIB address to 03. Ensure that the counter is the only device at this address.
5. In the analyzer **System** menu, point to **Service, Adjustments**, and click **10 MHz Freq. Adjust**.
6. Click **Begin Adj.** The application adjusts the internal reference for minimal error and stores the results.
7. Click **Read Freq** to trigger another reading of the 10 MHz signal.
8. Read the current DAC value stored in the analyzer's non-volatile memory (value = 0 - 4095).
9. When the status area indicates the adjustment is complete, click **Exit**.

### Procedure for Non-GPIB Counters

1. Connect the counter input to the rear panel 10 MHz Reference Output.
2. Set the counter to at least 1 Hz resolution.
3. If applicable, connect the house-frequency standard to the counter reference input.
4. In the analyzer **System** menu, point to **Service, Adjustments** and click **10 MHz Freq. Adjust**.

- Under **Frequency Counter**, select **Manual**.



- Adjust the slider bar **arrows** until the frequency counter reads 10.0 MHz at your desired level of accuracy.
- Click **Exit** to save the results.

**Note:** If the counter is misreading the frequency, it may be necessary to attenuate the input, or set the input impedance to 50 ohms, or both.

Last Modified:

9-May-2008     Updated with new UI

## Display Test

---

The PNA screen should be bright with all annotations and text readable. The display test allows you to check for non-functioning pixels and other problems.

**Note** If the display is dim or dark, refer to “Troubleshooting LCD Display Problems” in the PNA Service Guide.

### What Is a Damaged Pixel?

A pixel is a picture element that combines to create the image on the display. They are about the size of a small pin point. Damaged pixels can be either “stuck on” or “dark.”

- Stuck on pixel - red, green, or blue; always displayed regardless of the display setting. It will be visible on a dark background.
- Dark pixel - always dark; displayed against a background of its own color.

### How to Run the Display Test

On the **System** menu, point to **Service**, and then click **Display Test**.

A multi-color screen is displayed. Be prepared to look for the symptoms described below. Click the Start Test button. To continue to the next test, click the moving Next Test button. The button moves to allow you to see all of the display. After the test is completed, the display defaults to the network analyzer screen.

### How to Identify a Faulty Display

One or more of the following indicate a bad display:

- Complete row or column of “stuck on” or “dark” pixels
- More than six “stuck on” pixels (but not more than three green)
- More than twelve “dark” pixels (but not more than seven of the same color)
- Two or more consecutive “stuck on” pixels or three or more consecutive “dark” pixels (but no more than one set of two consecutive dark pixels)
- “Stuck on or “dark” pixels less than 6.5 mm apart (excluding consecutive pixels)

If any of these symptoms occur, your display is considered faulty. See the Service Guide for your PNA model.

## LO Power Adjustment

---

This procedure adjusts the receiver's LO input power to a specific level by changing DAC values. These DAC values are then stored in the analyzer's non-volatile memory. The procedure will vary depending upon the model number. This adjustment is only applicable to some PNA models.

### Power Meter Compatibility

This routine is only compatible with the Agilent EPM series of power meters. Different sensors may be used. For 9 GHz analyzers and below, an 8482 or E4412A sensor can be used. For the higher frequency units (20 GHz or above), a sensor must be able to measure a maximum of 20 GHz. At no time during this test will a frequency higher than 20 GHz be measured, even if the PNA has a maximum frequency of 50 GHz.

If the older HP 84xx series of sensors are used, the correct calibration data should be entered into the appropriate cal table of the EPM series power meter, although for this adjustment, high accuracy is not required. Inaccuracies in the order of several tenths of a dB are acceptable.

### Procedure

1. Allow the analyzer and power meter to warm up for 30 minutes.
2. Manually zero and calibrate the power sensor. (This allows you to skip this step later)
3. Connect a GPIB cable from the analyzer to the power meter. Make sure no other controllers are active on the same connection.
4. Set the power meter GPIB address to 13. (others can also be used; 13 is the default)
5. Remove the outer cover on the analyzer.
6. In the PNA **System** menu point to **Service**, then **Adjustments**, and click **LO Power Adjust**.
7. Connect the power sensor to the LO output, using adapters if needed.. The LO output location varies with model number. Click on the LO Power Adjust **Setup** menu selection to see a diagram of the exact location.
8. **For 9 GHz units and below:**  
Click **Begin Adj** to start the LO power cal routine. The routine adjusts the power level for each band (1 through 3) to fall within certain bounds. If any changes are made, it automatically stores them.

#### **For 20GHz units and above:**

If using an 84xx power sensor, click **Configure** and select the proper sensor model number. Click **Close**.

Click **Calibrate** to begin the adjustment. The entire calibration process takes about 5 minutes. Once completed, you can verify the current calibration accuracy by clicking **Verify Cal**.

**Note:** Correction constants are defaulted at the beginning of calibration. Once the calibration process has started, it must be completed in order to regenerate proper data.

9. Click **Read DAC** to view the current DAC values (0-4095) stored in the PNA non-volatile memory for each band (0-7).

10. When the message/status area indicates the adjustment is complete, click **Exit**.
11. Reconnect the semi-rigid cable and replace the covers.

## Offset LO Power Adjustment

---

**Note:** This adjustment is only performed on PNAs with Frequency Offset Mode (option 080), and only on certain models.

The Offset LO Adjustment sets the LO power for the offset mixer to a consistent value across all bands. It requires access to the internal components of the PNA so that the power sensor can be connected to the LO output.

Because the LO frequency does not exceed 3 GHz, almost any power sensor can be used. The adjustment is relatively simple and only takes a couple of minutes.

### When to perform

This adjustment should be performed when any of the following occur:

- the A13 Frequency Offset Receiver is significantly modified or replaced
- the A9 Synthesizer is replaced (that is where the correction data resides)

### How to perform Offset LO Power adjustment

1. To start the Offset LO adjustment, click **System**, point to **Service, Adjustments**, then click **Offset LO Adjust**.
2. You will be prompted to zero and cal the power sensor. (You can do this before beginning.)
3. Connect the sensor to J3 of the A13 LO output by removing the existing cable (or simply disconnecting one end) as shown in the Set-up diagram in the program.
4. Connect the power meter to the PNA using a GPIB cable. Make sure the GPIB address shown in the program matches the actual power meter address (default is 13.)
5. Click **Adjust** to begin the test and follow the instructions.

The program automatically adjusts all bands; no user input is needed. The program repeats several times as this is an iterative process. The progress of the adjustment is shown on the screen.

The Configure menu selection is for factory personnel ONLY.

Once completed, to verify the actual results, click Verify.

Upon exiting, the PNA application will restart; this takes several seconds.

# Operator's Check

---

## [Overview](#)

## [How to Run the Operator's Check](#)

## [Operators Check Dialog Box Help](#)

### Overview

The Operator's Check should be performed when you first receive your PNA, and any time you wish to have confidence that the PNA is working properly.

#### Notes

- The Operator's Check does not verify performance to specifications. To verify PNA performance to specifications, run [System Verification](#).
- Allow the PNA to warm up for 90 minutes before considering a failed test to be valid.
- The Operator's Check can NOT be run with a Multiport test set enabled. However, you can run a performance check as described in the Test Set User's Guide. [See the N44xx User's Guide](#).

The Pass/Fail criteria used in the Operator's Check identifies **obvious failures** in the following portions of the PNA hardware:

- Repeatability of the RF switch in the test set
- Attenuation ranges of the test port attenuators (if installed)
- Calibration of the receivers
- Frequency response of the receivers
- Phase lock and leveling
- Noise floor and trace noise

## How to Run the Operator's Check

Using front-panel  
HARDKEY [softkey] buttons

Using a mouse with PNA Menus

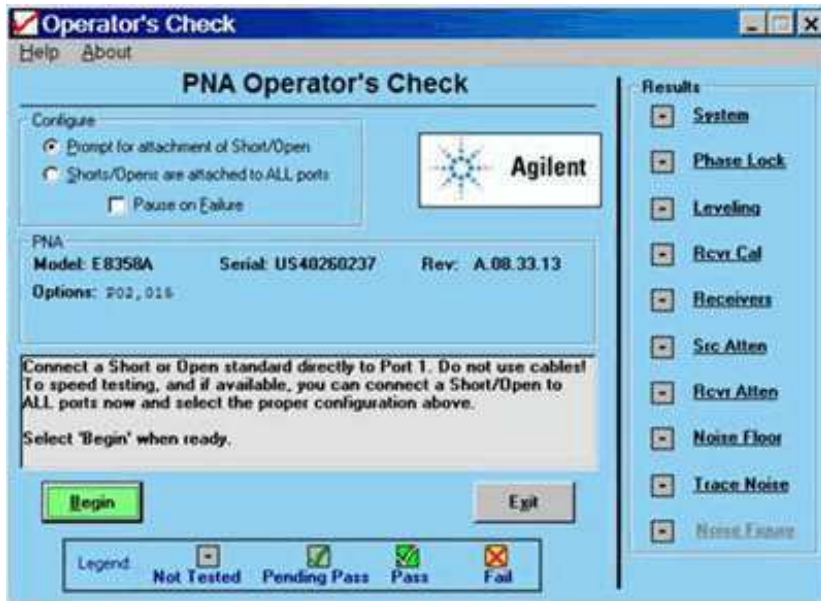
For [PNA-X](#) and ['C' models](#)

◀ **Programming Commands** ▶

1. Press **SYSTEM**
2. then **[Service]**
3. then **[Operator's Check]**

1. Click **Utility**
2. then **System**
3. then **Service**
4. then **Operator's Check**

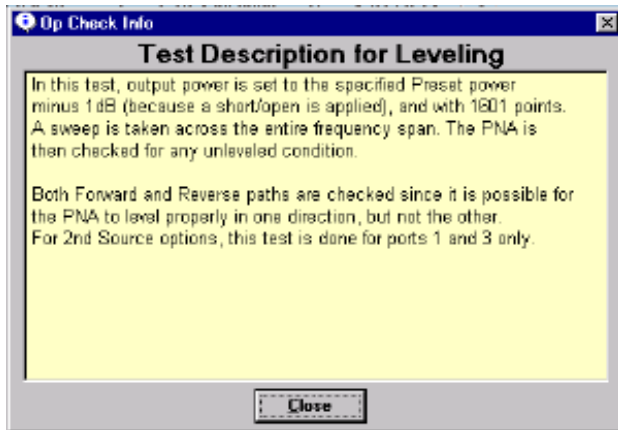
1. Connect one or more standards (see [Configure](#)).
2. Click **Begin** and **Continue** (if necessary) until "Operator's Check is complete!" appears.



This dialog box will look slightly different, depending on PNA model number and installed options. Some of the tests are not run if the appropriate option is not installed.

To learn about how each test is performed, click one of the tests on the right of the dialog. For example, the following information dialog is launched when **Leveling** is clicked:





## Operators Check dialog box help

**Note:** It is normal for a momentary unlevelled condition to appear during portions of the Operators Check.

### Configure

**Prompt for attachment of Short / Open** If you do not have enough shorts or opens for all test ports, you will be prompted to move the standard to the next test port. Connect either a short or open to port 1, then click Begin.

**Shorts / Opens are attached to all ports** Connect either a short or open for each test port, then click Begin. All ports are tested without interruption. You can mix shorts and opens on the test ports.

**PNA** Shows information about the PNA that is being tested.

**Legend** Shows the status icons used in the Operator's Check and their meaning. **Pending Pass** means that a portion of the testing has been completed successfully.

**Results** Shows the current status of each test. Click on the test name to learn how that test is performed. This may help in troubleshooting failed tests. If any tests Fail, refer to Chapter 3 of the PNA service guide.

**Begin** Starts the Operator's Check.

**View Results** Shows all results in text format. Failed items are preceded by ==>>>.

This text file can be printed or saved with a unique file name to compare results with previous or subsequent testing.

**Exit** Ends the program and closes the window.

Last Modified:

4-Sep-2008    Removed legacy content

17-Apr-2008    Added test description note

5-Feb-2008    Added multiport test set note

## Option H11 Test Verification

The PNA Option H11 Test / Verification utility verifies the PNA's option H11 functionality. The associated [H11 inputs and outputs](#) are tested and the results are compared against [expected levels at the factory](#). These tests are used in conjunction with the [Operator's Check](#) and [System Verification](#) programs to check PNA functionality.

### How to Run Option H11 Verification

Using front-panel HARDKEY [softkey] buttons	Using a mouse with PNA Menus
<ol style="list-style-type: none"><li>1. Press <b>SYSTEM</b></li><li>2. then <b>[Service]</b></li><li>3. then <b>[Rear Panel/H11 Test or H11 Verification]</b></li></ol>	<ol style="list-style-type: none"><li>1. Click <b>System</b></li><li>2. then <b>Service</b></li><li>3. then <b>Rear Panel/H11 Test or H11 Verification</b></li></ol>



### Option H11 Verification dialog box help

The Option H11 Verification software leads you through each of the tests listed on the Configure Tab.

**Important:** Required Configuration of the PNA being tested:

- All connections to external devices and test sets must be removed from the Option H11 connectors and Test Set I/O connector on the PNA rear panel.
- On the [Millimeter Module Configuration](#) dialog box, make the following settings:
  - PNA Model E8361A - Select **Agilent N5250A**, and check **Use Standard PNA operation when N5260A is NOT connected**.
  - All other PNA models - Select **Standard PNA**.

## Run Tab

The list of test equipment is the complete list that is required to run ALL of the tests (recommended).

**Start** Click to run the tests that are selected on the Configure Tab.

**Exit** Exits Option H11 Verification.

## Configure Tab

All of the tests are selected by default. Tests can be run separately for troubleshooting purposes by checking or clearing the boxes.

### Power Tests

Output power is measured at each of the selected [RF and LO Test Set connectors](#) on the rear panel. The results are displayed against limit lines that represent the [expected levels at the factory](#).

Equipment Required:

- [PNA supported power meter and 26.5 GHz power sensor](#) (Recommended: Agilent E4418/19 Power Meter and E4413A Power Sensor). Set GPIB address = 13.
- GPIB and Sensor cables.
- On E8361A ONLY, a device is required to simulate an N5260A Test Set is connected to the Test Set I/O. All other PNA models do NOT require this device for this test.
  - Agilent Service personnel: Use the E8361-60063 Test Set I/O Tester.
  - All others: On a commercially available DB-25 male connector, connect a jumper between pins 12 and 22. See the [Test Set I/O connector diagram](#). Insert the male connector into the PNA rear panel Test set I/O connector.

### External IF Input and Crosstalk Tests

A 8.33 MHz signal is injected into each of the selected [8.33 MHz IF IN connectors](#) on the rear panel. The signal is measured at each of the selected receivers. The results are displayed against limit lines that represent the [expected levels at the factory](#).

Equipment Required:

- 33120A Function Generator. Set GPIB address = 10.
- 10 dB Attenuator (Agilent 8493B or equivalent) and BNC adapters.
- GPIB and BNC cables

### External Pulse Input Test

A pulse train is injected into each of the selected [Pulse IN connectors](#) on the rear panel. The signal is measured at each of the selected receivers. The results are displayed against limit lines that represent the [expected levels at the factory](#).

**Note:** A noisy pulse generator can cause a false failure of the "1% Duty Cycle Pulse Input" portion of the

## External Pulse Input Test.

### Equipment Required:

- 8110A or 81110A Pulse Generator. Set GPIB address = 11.
- GPIB and BNC cables

## Test Set I/O Connector Test

Tests the ability of the PNA to detect and control an external test set through the PNA rear panel TEST SET I/O connector.

### Equipment Required:

- Agilent Service personnel: Use the E8361-60063 Test Set I/O Tester.
- All others: Use a DVM to measure the TTL voltage level on the following pin numbers using pin 1 as ground:
  - Pins: 3, 4, 5, 6, 8, 9, 10, 11, 17, 19, 20, 21, 22, 23, 24
  - See [Test Set I/O connector diagram](#) to determine the pin number location.
  - The software toggles all pins HIGH, then LOW.
  - TTL HIGH should read approximately +5.0 volts.
  - TTL LOW should read approximately +0.1 volts.

## Help Tab

Displays the Option H11 Verification program revision number and information about the PNA being tested.

**Advanced Help** Click to display this help topic.

**View Results** Appears after the test is run. Click to show all results in text format. Failed items are preceded by ==>>> **Fail**. This text file can be printed or saved with a unique file name to compare results with previous or subsequent testing.

Last Modified:

3-Jun-2010    MX added UI

## Phase-Lock IF Gain Adjustment for E836x models only!

---

The E836x A/B PNA models have a variable gain control for the phase-lock loop IF signal. By dynamically changing the gain as a function of frequency and power, the phase-lock signal amplitude can be adjusted to a constant level for the entire operating range of the instrument. This constant level is important for phase-lock acquisition and stability.

### When to perform

Phase-Lock IF Gain Adjustment should be performed when any of the following occur:

- A source calibration
- An assembly in the reference receiver path (R1,R2) is replaced.
- The Test Set Motherboard is replaced
- The Phase Lock board is replaced
- **Phase Lock Lost** error message appears after replacing a source or receiver assembly
- The external R path has changed. For example, when a multiport test set with R channel path has been added or removed.

### How to perform Phase-Lock IF Gain adjustment

Ensure the Reference Channel paths are properly configured and the connections are properly torqued.

1. From the **System** menu, click **Service**, then **Adjustments**, then **IF Gain Adjustment**.
  2. Under Select Specials, select **None**.
  3. No connections to the test ports are required.
  4. Click **Begin Adj.** The adjustment takes about a minute to complete.
  5. The advanced screen is for factory personnel only.
- 
- 

## IF Gain Adjustment for N524x models only!

---

The N524x PNA models have a variable gain control for the loop IF signal. By dynamically changing the gain as a function of frequency and power, the phase-lock signal amplitude can be adjusted to a constant level for the entire operating range of the instrument. This constant level is important for phase-lock acquisition and stability.

### When to perform

IF Gain Adjustment should be performed when any of the following occur:

- An assembly in the reference receiver path (R1,R2) is replaced.
- The Test Set Motherboard is replaced
- The IF Mux assembly or SPAM assembly is replaced

### How to perform IF Gain adjustment

Ensure the Reference Channel paths are properly configured and the connections are properly torqued.

1. From the **System** menu, click **Service**, then **Adjustments**, then **IF Gain Adjustment**.
2. An appropriate power meter and sensor will be required. The sensor must cover the upper frequency range of the PNA. A good quality cable will also be needed. Adapters can be used as needed.
3. Connect the power meter to the GPIB port labeled System Controller.
4. Click **Begin Adj.**
5. Set the GPIB address in the program accordingly and select the sensor being used.
6. Follow the instructions displayed in the program.

The adjustment takes about 5 minutes to complete for a 26.5 GHz PNA. Higher frequency units may take longer.

The Default menu selection is for factory personnel only. This will preset all values to default levels for troubleshooting purposes only. If this is selected, a full IF gain adjustment will need to be performed.

---

Last modified:

3/7/07    Added N5242x information.

## System Verification

---

The performance of the network analyzer is specified in two ways: system specifications, and instrument specifications. It is the end user's responsibility to determine which set of specifications is applicable to their use of the PNA.

A network analyzer measurement "system" includes the analyzer, calibration kit, test cables, and any necessary adapters. The system verification software in the PNA is used to verify the system's conformance to the "system" specifications. A "pass" result demonstrates that the analyzer, test cables, and adapters, perform correctly as a system. It DOES NOT demonstrate that any one component performs according to its individual specifications. A change to any part of this measurement system requires a re-verification of the system.

Instrument specifications specify the network analyzer's uncorrected measurement port characteristics and its output and input behavior. The PNA performance tests are used to verify the analyzer's conformance to "instrument" specifications.

The system verification utility verifies the PNA system specifications by automatically measuring the magnitude and phase for all four S-parameters for each verification device, and comparing the values against the following:

- Factory measured data from files on the verification disk
- Limit lines based on the measurement uncertainty

System Verification requires the use of a calibration kit and verification kit which has been certified within the past 12 months by Agilent. System Verification can NOT be used to perform this kit certification.

[Operator's Check](#) should also be performed to verify the basic operation of the PNA.

### [Equipment Used in the System Verification](#)

### [Precautions for Handling Airlines](#)

### [Flow Diagram of Procedure](#)

### [Procedure for System Verification](#)

### [If the System Fails the Verification Test](#)

### [Interpreting the Verification Results](#)

#### Notes

- Although the performance for all S-parameters is measured, the S-parameter phase uncertainties are less important for verifying system performance. Therefore, the limit lines will not appear on the printouts.
- System Verification can NOT be run with a Multiport test set enabled. However, you can run a performance check as described in the Test Set User's Guide. [See the N44xx User's Guide.](#)

## Equipment Used in the System Verification

For PNA models:

**E8356A, E8357A, E8358A**  
**N3381A, N3382A, N3383A**  
**E8801A, E8802A, E8803A**  
 (Type-N test ports)

Equipment Type	Type-N	3.5 mm
Calibration kit <b>or</b> ECAL Module	85032F	85033E
Verification kit	85092B	85093B
RF Cable	85055A	85053B
	N6314A	<a href="#">See Cable substitution</a>

**E8362A/B/C**  
**N5230A (20 GHz)**  
 (3.5 mm test ports)  
**N5241A and N5242A**

Equipment Type	3.5 mm	Type-N
Calibration kit <b>or</b> ECAL Module	85052B/C/D	85054B/D
Verification kit	N4691A	N4690A
RF Cable(s)	85053B	85055A
	Single: 85131C/E Pair: 85131D/F	Single: 85132C/E Pair: 85132D/F
Adapters	None	Single: 85130C and one 7mm-to-Type-N from 85054B cal kit <u>Pair:</u> Two 7mm-to-Type-N from 85054B cal kit

**E8363A/B, E8364A/B/C**  
**N5230A (40 or 50 GHz)**  
 (2.4 mm test ports)



Equipment Type	2.4 mm	3.5 mm	Type-N
Calibration kit or ECAL Module	85056A/D N4693A	85052B/C/D N4691A	85054B/D N4690A
Verification kit	85057B	85053B	85055A
RF Cable(s)	Single: 85133C/E Pair: 85133D/F	Single: 85134C/E Pair: 85134D/F	Single: 85135C/E Pair: 85135D/F
Adapters	None	<u>Single:</u> 85130F <u>Pair:</u> None	<u>Single:</u> 85130E and two 7mm-to-Type-N from 85054B cal kit <u>Pair:</u> Two 7mm-to-Type-N from 85054B cal kit

### E8361A/C

(1.85 mm test ports)

Equipment Type	1.85 mm	2.4 mm 3.5 mm Type-N
Calibration kit or ECAL Module	85058B N4694A	See 2.4 mm test port table above
Verification kit	85058V	See 2.4 mm test port table above
RF Cable(s)	Single: N4697E Pair: N4697F	See 2.4 mm test port table above
Adapters	None	See 2.4 mm test port table above

### Cable Substitution

The test port cables specified for the PNA have been characterized for connector repeatability, magnitude and phase stability with flexing, return loss, insertion loss, and aging rate. Since test port cable performance is a significant contributor to the system performance, cables of lower performance will increase the uncertainty of your measurement. It is highly recommended that the test port cables be regularly tested.

If the system verification is performed with a non-Agilent cable, ensure that the cable meets or exceeds the operation of the specified cable. Refer to the cable User's Guide for specifications.

### Cable Flex Factor

Flex Factor determines how much of the cable phase uncertainty to include in determining the limit lines.

- Set to **0% (zero)** if the cables are held down in a fixture and are not allowed to move during the calibration and verification.

- Set to **100%** if the cables are allowed to move a lot.

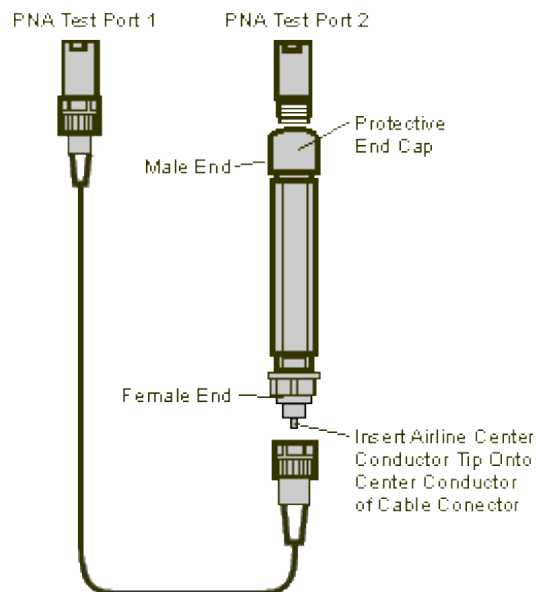
### Calibration Kit Substitution

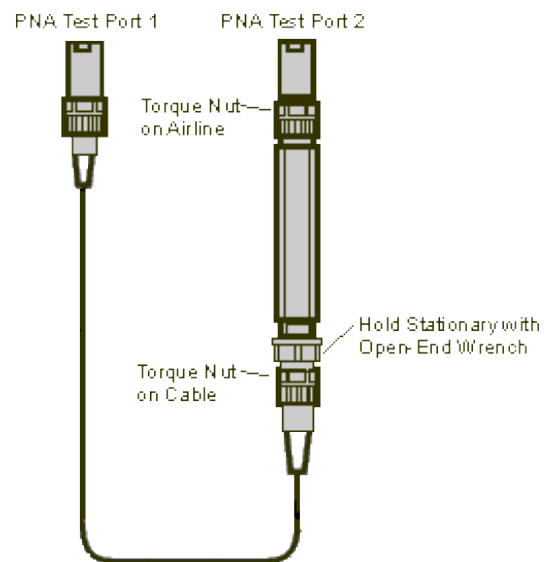
Non-Agilent calibration kits are not recommended nor supported.

### Precautions for Handling Airlines

When you are using the airlines in the verification kit, observe the following practices to ensure good measurement techniques.

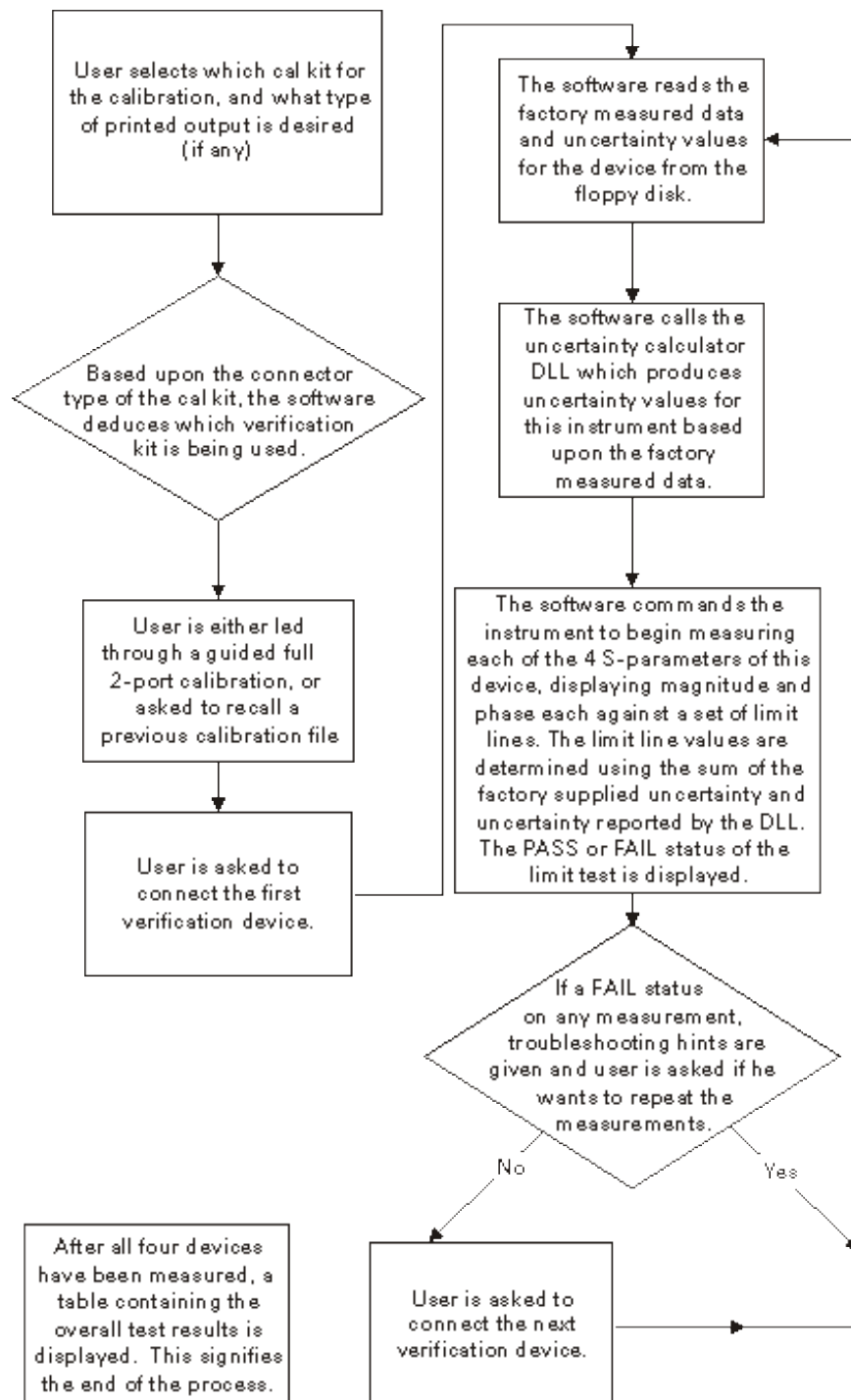
- Be very careful not to drop the airline's center or outer conductor. Damage will result if these devices are dropped.
- Use proper Electro-Static Discharge (ESD) procedures.
- Clean your hands or wear gloves as skin oils will cause a change in electrical performance.





### Flow Diagram of Procedure

The operational flow of the software is depicted by the flowchart shown below.



## Procedure for System Verification

1. If you want printed test outputs, connect a printer to the analyzer. Let the analyzer warm up for at least 30 minutes.
2. Insert the PNA verification kit floppy disk into the analyzer disk drive.

3. On the **System** (or **Utility**) menu, point to **Service**, and click **System Verification**. The System Verification window similar to this will be displayed.



4. In the **Calibration Kit** box, select the calibration kit or ECal module that is being used. The corresponding verification kit to use appears in the **Verification Kit** box.
5. Under **Printer Output** click on any of the following options.
  - **Print Tabular Data:** Prints the verification data in tabular form which includes measured data and uncertainty limits. Refer to a tabular data example, later in this topic.
  - **Print Graphs:** Prints the verification data in graphical form. The graphic form includes the measured data trace, factory supplied data trace and uncertainty limits. Refer to a plot data example, later in this topic.
  - **File Tabular Data:** Writes the verification data in tabular form to a text file in the C:/Program Files/Agilent/Network Analyzer/Documents/ directory.
  - **File Graphs:** Saves a screen image in .PNG format in the C:/Program Files/Agilent/Network Analyzer/Documents/ directory.

**Note:** If you want printed output, it is assumed you have already installed the Windows driver for your particular printer, and have tested that you can print to the printer from the network analyzer. This software is designed to print to whichever printer is currently set as the Default printer (see Printers in the Windows Control Panel).

6. To modify the number of ports to be verified, to change the number of devices to measure, or to use a previously stored verification calibration, click on the **Configure** tab and make the desired selections.
  - For the system verification to be truly adequate, the software must measure all devices in the kit with a recent calibration applied. Removing and reattaching any test port cables or adapters invalidates all

previous calibrations.

7. Click **Run**.
8. Follow the instructions on the analyzer for performing the system verification, inserting the verification devices as prompted.

**Note for 3 Port PNA:**

The System Verification Procedure is **repeated three times**. The first time, **Ports 1 and 2** are measured as a pair; then **Ports 1 and 3** are measured; and lastly, **Ports 2 and 3** are measured.

**Note for 4 Port PNA:**

The System Verification Procedure is **repeated two times**. The first time, **Ports 1 and 2** are measured as a pair, then **Ports 3 and 4** are measured.

## Step-by-Step Process Description

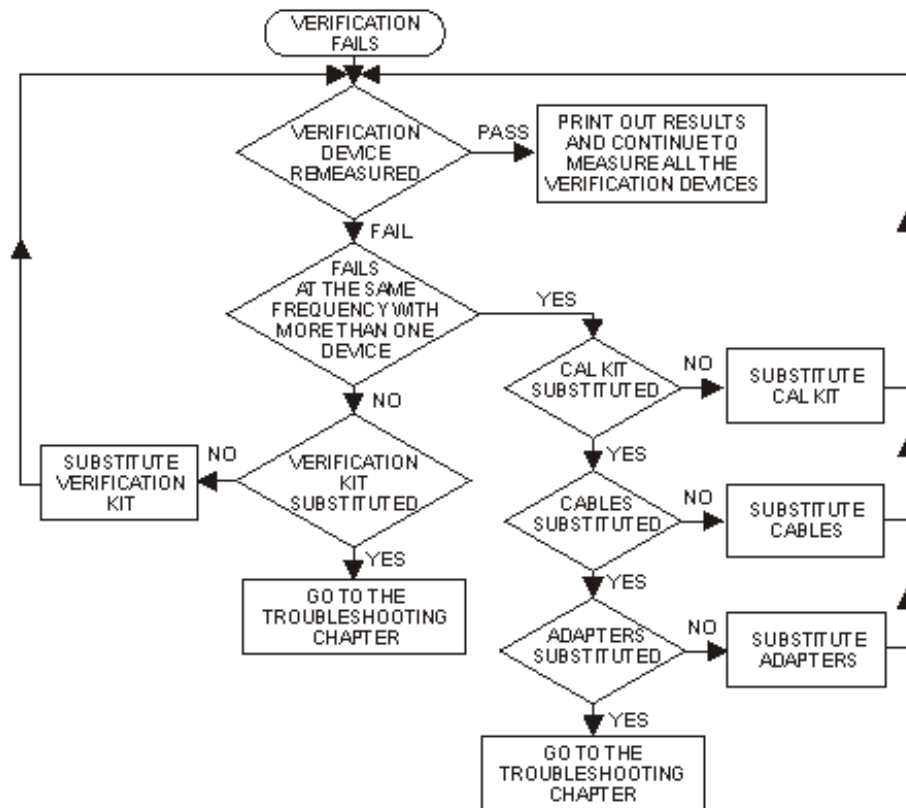
1. Depending upon the selected choice in the Calibration submenu of the Configure menu, the user is either prompted to recall a previous calibrated instrument state, or is guided through a full 2-port calibration using the selected calibration kit. For ECal, the ECal module is connected just once; a standby message is posted while the software is performing the calibration.
2. The user is prompted to connect the first verification device.
3. The software reads the factory measured data for that device and uncertainty values for that data (CITIfiles) from the floppy disk supplied with the verification kit.
4. The software sends the factory measured data, calibration kit and instrument state information to the uncertainty calculator DLL, which generates uncertainty values specific to the PNA.
5. The analyzer first sets up for magnitude measurements of all four S-parameters, each parameter in a separate window (lin mag for  $S_{11}$  and  $S_{22}$ , log mag for  $S_{21}$  and  $S_{12}$ ). Each of the factory measured S-parameters are fed to the appropriate window as a memory trace. Limit line offsets are calculated as the sum of the factory measured data uncertainties and PNA uncertainties reported by the DLL. Upper and lower limits are displayed (factory measured data + uncertainty sum, factory measured data - uncertainty sum). The PNA takes a sweep, limit test is turned on and PASS/FAIL status is reported in each of the four windows.
6. The user clicks a button when ready to view phase measurements. The four windows get updated for phase format, phase memory traces, phase limits and PASS/FAIL result.
7. If the limit test of any of the four S-parameters (magnitude or phase) indicates a FAIL status, the software suggests troubleshooting tips and asks if the user would like to repeat measurement of that device or proceed to the next device. If proceeding to the next device, the factory measured data and uncertainties for the next device are read from floppy, the uncertainty DLL gets called with this next set of factory measured data, and the four measurement windows get updated for magnitude measurement of the next device.
8. The software follows this same process until all selected devices have been measured, at which point a summary window is displayed containing the set of PASS/FAIL results for all four parameters of each device.

## If the System Fails the Verification Test

**IMPORTANT:** Inspect all connections. Do not remove the cable from the analyzer test port. This will invalidate the calibration that you have done earlier.

1. Repeat this verification test. Make good connections with correct torque specifications for each verification device.
2. Disconnect, clean and reconnect the device that failed the verification test. Then measure the device again.
3. If the analyzer still fails the test, check the measurement calibration by viewing the error terms as described in "Front Panel Access to Error Terms" on page 4-7 of the Service Guide.
4. Refer to the graphic below, for additional troubleshooting steps.

**Verification Fails Flowchart**



## Interpreting the Verification Results

The graphic below shows an example of typical verification results with **Tabular Data** selected in the **Printer Output** area of the **System Verification** window. A graphic later in this topic shows an example of typical verification results with **Measurement Plots** selected in the **Printer Output** area of the **System Verification** windows. These printouts include a comparison of the data from your measurement results with the traceable data and corresponding uncertainty specifications. Use these printouts to determine whether your measured data falls within the total uncertainty limits at all frequencies.

**The tabular data consists of:**

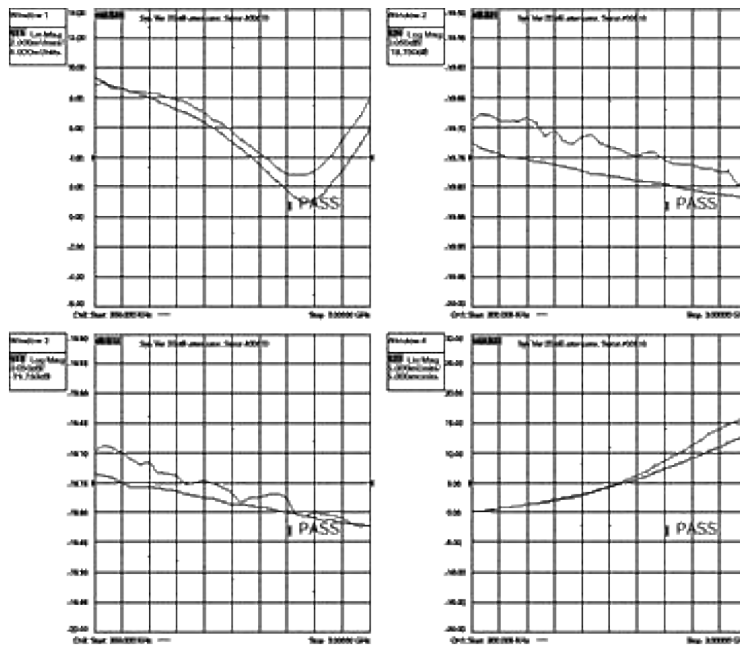
- ## Printout of Tabular Verification Results

[illegible]

- Upper limit points as defined by the total system uncertainty specifications.
- Lower limit points as defined by the total system uncertainty specifications.
- Data measured at the factory.
- Results of measurements.
- Measurement parameter names and formats (Lin Mag or Log Mag).
- Serial number of device (00810).
- Device being measured (Sys Ver 20 dB attenuator).

**www.valuetronics.com**





Last Modified:

- 6-Apr-2009 Updated for N5241A
- 8-Apr-2008 Updated for 'C' Models
- 11-Feb-2008 Added note about multiport
- 15-Jan-2008 Added Flex Factor and image

## Source Adjustment

---

Source Adjustment is a **SERVICE** Routine which should be performed when a component in the source chain is replaced, or when the PNA fails an annual calibration. It adjusts the PNA source power for flatness across its full frequency range.

This topic does **NOT** discuss [Source Power Calibration](#), which calibrates a PNA source over the current measurement range.

### Required Equipment

Preferred Power Meter: E4419B

Alternate Power Meters: E4419A or EPM-442A

**Note:** The power sensor depends on the PNA frequency range. Depending on the PNA model, two power sensors may be required to test the full frequency range.

The PNA front panel connector type will determine the cable used and if an adapter is required with the power sensor(s).

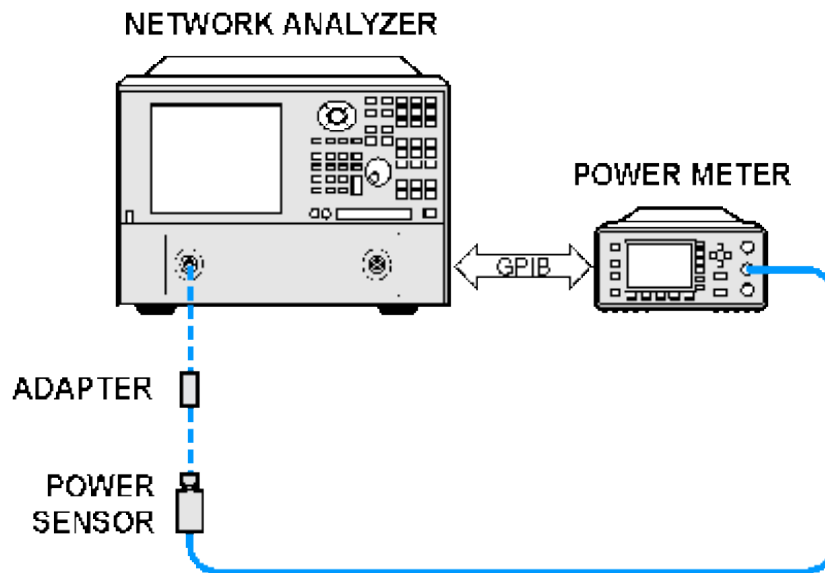
PNA Model	Power Sensor(s)	Cable
E8356A	8482A	N6314A
E8357A E8358A	8482A and E4412A	N6314A
E8801A	8482A	N6314A
E8802A E8803A	8482A and E4412A	N6314A
N3381A	8482A	N6314A
N3382A/N3383A	8482A and E4412A	N6314A
E8361A	8487A (use with adapter 11900B) and V8486A (use with adapter V281B)  ** <a href="#">See E8361A procedure</a> below	N4697-60001
E8362A/B/C	E4413A	85131E
E8363A/B/C E8364A/B/C	8487A (use with adapter 11900B)	85133E
N5241A N5242A	E4413A	85131E

[See PNA Accessories](#)

## Procedure

1. Refer to your power meter documentation to ensure the proper calibration factors for the power sensor have been entered into the table on the power meter.
2. Connect a GPIB cable between the power meter and network analyzer (use the System Controller GPIB port if applicable.)
3. Ensure the power sensor(s) are connected to the power meter.
4. In the analyzer **System** menu, point to **Service, Adjustments**, and click **Source Adjustment**.
5. There are 3 different version of the Source Calibration software; all are slightly different. All have a button that is labeled "Calibrate" or "Adjust". This is the button that will begin the calibration process. Some versions will also have a button labeled "Verify" that will test the source calibration without making any changes. Other selections are for factory personnel use only.
6. Once begun, you must enter the power meter and sensor information. The software will verify the power meter and sensor. You are then prompted to connect the sensor(s) and cable as needed.

### Connecting sensors to the PNA



## Additional Information

All ports are tested on all PNAs. Source calibration takes approximately 10 to 45 minutes to complete depending on the frequency range and model number of the PNA. The E8361 models may take up to 90 minutes.

## Troubleshooting

In the event there is a problem with Source Adjustment, please refer to the "Troubleshooting" chapter in the PNA

## E8361 Procedure

Source and Receiver adjustment requires the power meter to measure the source power over the full range of each of the PNA internal bands. Because the 8487A can not measure accurately above 50 GHz, it can only be used up to the next highest band switch frequency at 46.2 GHz. The V8486A sensor and V281B adapter are used from 46.2 GHz to 67 GHz.

For highest accuracy, the V8486A and V281B should be sent to Agilent for a custom calibration from 45 GHz to 70 GHz.

For the next highest accuracy level, the following procedure shows how to measure correction factors yourself from 46 to 50 GHz. This procedure assumes you have already loaded correction factors for both sensors into the power meter.

1. On your power meter, add 46 and 48 GHz to the Cal Factor Table.
2. Preset the PNA
3. Tune the PNA to 46 GHz (CW frequency)
4. Using the 8487A, measure power at port 1. Record this value.
5. Tune the PNA to 48 GHz (CW frequency)
6. Using the 8487A, measure power at port 1. Record this value.
7. Connect the V8486A, V281A, and 1.85 f-f adapter to the power meter.
8. Tune the PNA to 46 GHz (CW frequency)
9. Adjust the cal factor table 46 GHz setting until the power meter reading matches the power readings from step 4.
10. Tune the PNA to 48 GHz (CW frequency)
11. Adjust the cal factor table 48 GHz setting until the power meter reading matches the power readings from step 8.

---

Last Modified:

6-Apr-2009      Updated models

10-Mar-2009      Changed to Source Adjustment

## Receiver Calibration

---

Receiver calibration adjusts the network analyzer receivers for a flat response across its full frequency range. This adjustment is for service only; not for measurement calibration.

### Required Equipment

Preferred Power Meter: E4419B

Alternate Power Meters: E4419A or EPM-442A

**Note:** The power sensor depends on the PNA frequency range. Depending on the PNA model, two power sensors may be required to test the full frequency range.

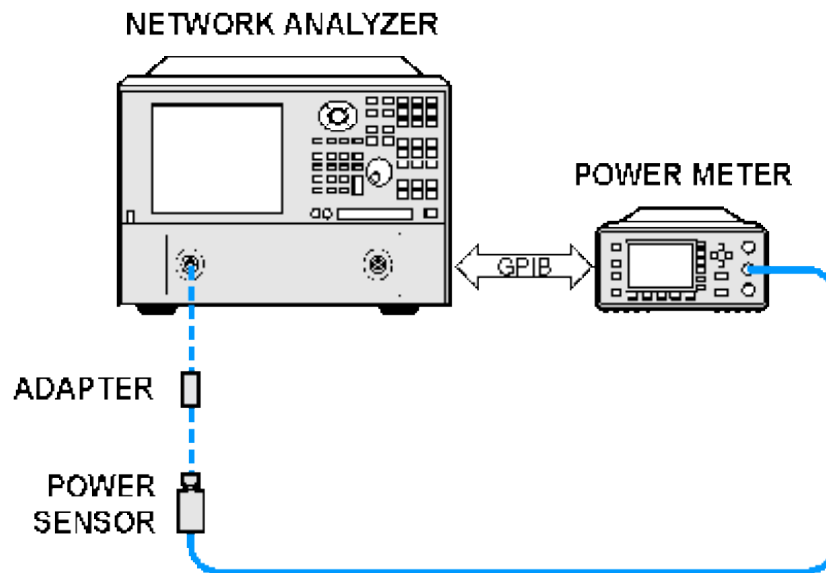
The PNA front panel connector type will determine the cable used and if an adapter is required with the power sensor(s).

See [PNA Accessories](#)

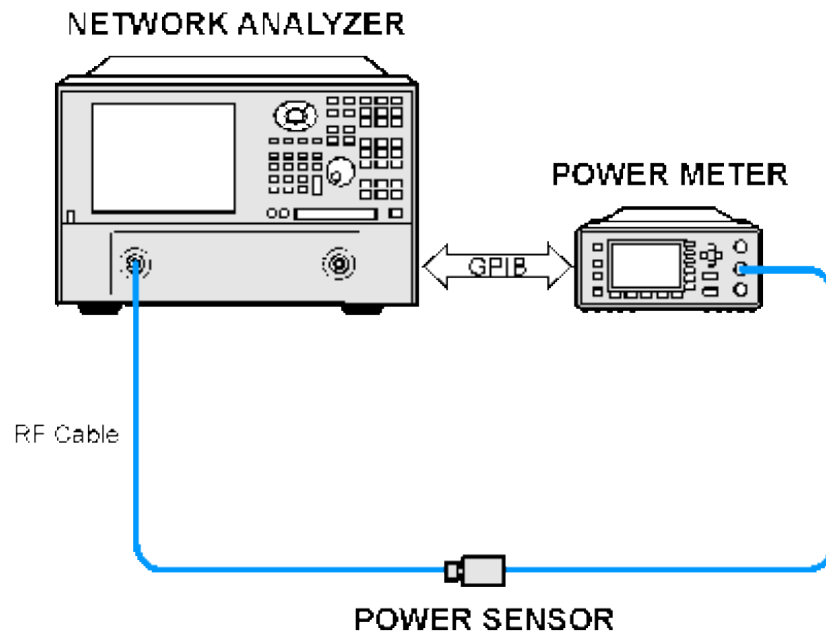
### Procedure

1. Refer to your power meter documentation to ensure the proper calibration factors for the power sensor have been entered into a table on the power meter.
2. Connect a GPIB cable between the power meter and network analyzer.
3. Ensure the power sensor(s) are connected to the power meter.
4. In the analyzer **System** menu, point to **Service, Adjustments**, and click **Receiver Calibration**.
5. The software presents you with two choices:
  - a. Click **Inspect Flatness** to observe flatness of receiver response versus frequency. Although there is no explicit specification for receiver flatness, Receiver Calibration should improve Transmission and Reflection Tracking error terms which are specified.
  - b. Click **Calibrate** to begin the receiver calibration process. The software prompts you to connect the sensor(s), cable and adapter as needed (see the following graphics).

### Connecting sensor(s) to the PNA

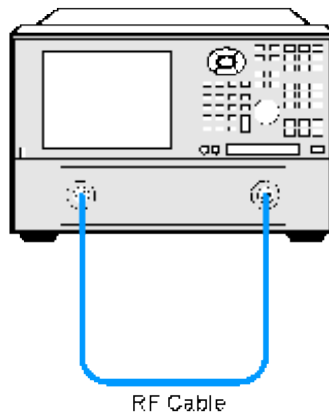


**Connecting adapter and cable between sensor and PNA**



**Through connection using the specified cable**

## NETWORK ANALYZER



RF Cable

### Additional Information

Receiver Calibration tests all PNA receivers, taking approximately 15 and 45 minutes. Length is dependent on frequency range and number of ports.

### Troubleshooting

In the event there is a problem with Receiver Calibration, please refer to the "Troubleshooting" chapter in your PNA Service Guide.

---

Last Modified:

12-Jul-2007    Removed outdated table of supported power meters

## Receiver Display

---

- [The Receiver Display as a Troubleshooting Tool](#)
- [How to start the Receiver Display](#)

### Other Support Topics

## The Receiver Display as a Troubleshooting Tool

The Receiver Display is a Troubleshooting Tool. It enables the analyzer to isolate faulty functional groups within its own Measurement System. Traces for each Receiver are Displayed in individual windows. Identifying discrepancies of the traces in these windows can help isolate the faulty assembly.

For a thorough description of Receiver Display and the troubleshooting steps see Chapter 3 of the PNA Service Guide. You can download the Service Guide for your PNA model from our website: <http://na.tm.agilent.com/pna/>

### How to Start the Receiver Display

#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Service]**
3. then **[Utilities]**
4. then **[Receiver Display]**

#### Using a mouse with PNA Menus

1. Click **Utilities**
2. then **System**
3. then **Service**
4. then **Utilities**
5. then **Receiver Display**

◀ [Programming Commands](#) ▶



## PNA-X IF Path Configuration Settings

---

This dialog is used to set many IF receiver settings for PNA-X models. See the [IF Path settings for "C" models](#).

In this topic:

- [PNA-X IF Frequencies](#)
- [How to Start the IF Path Configuration dialog](#)
- [IF \(Receiver\) Path Configuration dialog box](#)
- [Expanded PNA-X Block Diagram and Descriptions](#)
- [PNA-X IF Path Configuration using COM and SCPI](#)

### Other IF Access Topics

#### PNA-X IF Frequencies

**Wideband/Normal IF path:**

- RF < 53 MHz: IF = 2.535211 MHz
- RF >= 53 MHz: IF = 7.605634 MHz

**Narrowband IF path:**

- IF = 10.70 MHz
- Bandwidth = 30 kHz

Change the PNA-X IF frequency to any value between +14.9999 MHz and -14.9999 MHz using [SENS:IF:FREQ](#) or [IFFrequency](#) commands.

## How to start the IF Path Configuration dialog

To provide quicker access, use the Setup softkey. [Learn how.](#)

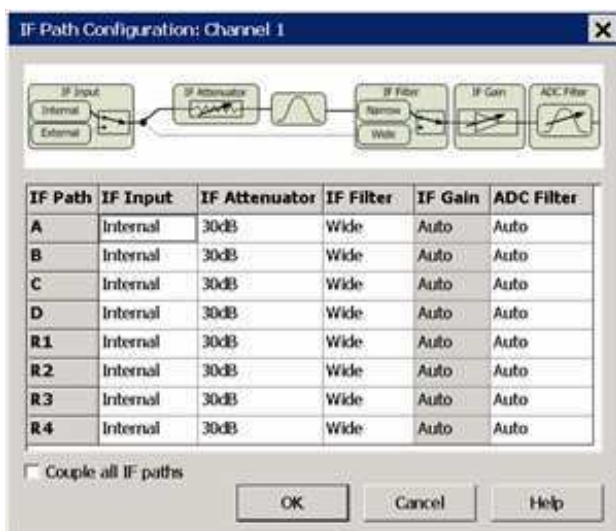
## Programming Commands

### Using front-panel HARDKEY [softkey] buttons

1. Press **CHANNEL**
2. then **[Hardware Setup]**
3. then **[IF Config]**

### Using a mouse with PNA-X Menus

1. Click **Trace/Chan**
2. then **Channel**
3. then **Hardware Setup**
4. then **IF Config**



## IF (Receiver) Path Configuration dialog box help

The IF path, represented in the block diagram at the top of the dialog, is duplicated for each of the receivers (A, B, C, D, R1, R2, R3, R4). In addition, each path can be configured differently for each channel.

Switch - Indicates element in the PNA-X [expanded PNA-X block diagram](#).

**IF Input** (1) - Available on the [PNA-X with Opt 020](#). Internal input is a test port or reference receiver input. External Input is through the [PNA-X rear-panel connectors](#).

**IF Attenuator** (3) - Specify IF attenuation for the narrowband path of the selected receiver.

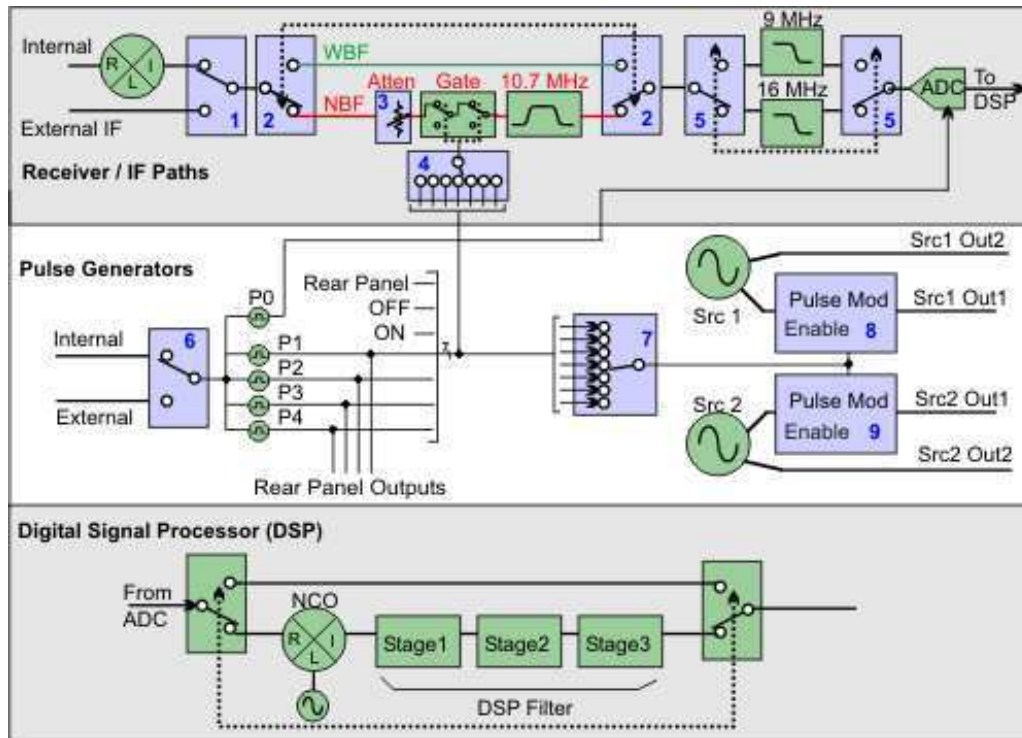
**IF Filter** (2) - Select Wideband or Narrowband (includes the ability to pulse gates).

**IF Gain** - ONLY Auto is available on PNA-X models.

**ADC Filter** (5) - Select Auto, 9 MHz, or 16 MHz filter. The same setting is made for all receivers.

**Couple all IF paths** - Check to make the same setting for all receivers.

## Expanded PNA-X Block Diagram (IF / Pulse Generators / DSP)



**Blue boxes** are configurable elements.

Click a blue box, or scroll down, to see how to make settings using [SCPI](#) and [COM](#) commands.

### Description:

#### Receiver / IF Path (top block)

[Scroll up](#) for descriptions of the Receiver / IF Paths. Most of these elements can be set from the front-panel User Interface (UI).

#### Pulse Generators (middle block in [above diagram](#))

[See how to make these settings remotely](#)

**Switch 6:** Represents **Internal** or **External** triggering for the pulse generator. If set to Internal, the pulse generator is internally triggered and puts out a periodic pulse train with a period defined by the pulse generator SCPI or COM commands. If set to External, the period is ignored, and the pulse generator puts out one set of pulses (P0-P4) per external trigger. All five pulse outputs have unique delay and pulse width settings. See all Pulse Generator [SCPI](#) and [COM](#) commands.

- **External** trigger input is on the [Pulse I/O connector](#) pin 7 (PulseSyncIn). The PulseSyncIn line provides a configurable trigger signal into the Pulse Generators. If a level trigger is still valid when the first pulse set is finished, another set will be generated. Only one set of pulses is emitted when edge triggering is used. The length of time that it takes to emit one set of pulses is the end time of the last enabled pulse (largest of width + delay of all the pulses P0-P4).

**New with A.09.20** The pulse input polarity (positive or negative) and type (edge or level) is configurable only

with DSP version: 4.0 FPGA: 34 or higher. [Learn more](#). Otherwise, the pulse generators respond only to positive, level input trigger signals. Externally triggering is configured using the Pulse Generator 'Polarity' and 'Type' [SCPI](#) and [COM](#) commands.

**P0:** When P0 is enabled, it is hardwired to trigger the data acquisition ADCs. If the data acquisition system is not ready, the P0 trigger is ignored. If the pulse generator is internally triggered, then the data acquisition system receives periodic triggers. If the pulse generator is externally triggered, then the data acquisition system receives a trigger each time the pulse generator is triggered. In either case, data acquisition is synchronized to the pulse generator. Data acquisition begins on the rising edge of P0. The width of P0 does NOT directly matter as data acquisition does not stop when P0 goes false. The width of P0 DOES matter when determining the time for one set of pulses when externally triggered (see [External trigger](#) above). The following describes how the P0 generator triggers data acquisition:

- **Step mode sweeps** of any sweep type: By default, each P0 rising edge triggers a single data point. When [point averaging](#) is on, all of the measurements (subpoints) that are required to average each point are made with a single trigger. To individually trigger the acquisition of each subpoint, send the subPointTrigger ([SCPI](#) or [COM](#)) command.
- **CW sweeps:** Each trigger initiates acquisition for the entire sweep. This is currently used for wideband pulse profiling.

**P1 thru P4** These four pulse generator outputs are hardwired to rear panel outputs on the [Pulse I/O connector](#) (pins 10 - 13). They are also routed to two switches (#4 and #7 on the above diagram) along with the following three lines:

- **Rear Panel** External pulse generator input from [Pulse I/O connector](#) pin 8 (RFPulseModIn).
- **OFF** Pulse is constantly in LOW state causing gate and source to be OFF.
- **ON** Pulse is constantly in HIGH state causing gate and source to be ON.

**Switch 7** Pulse Modulation - 1 of 7 lines to each of the sources. **Important:** When internally modulating the sources, [source leveling must be set to Open-loop](#).

**Rear-panel Outputs:** [Pulse I/O connector](#) (pins 10 - 13) hardwired.

**Source1 and Source2 pulse modulators:** (#8 and #9 on the above diagram)

#### **DSP (bottom block)**

- Filters the ADC (digital) output from top block and outputs data to the PNA display.
- See [SCPI](#) and [COM](#) commands to control DSP settings.

**See Also**

- [Pulse Settings](#)
- [Remote RF Path Configuration](#)
- [Rear Panel Pulse I/O connector](#)

## PNA-X IF Path Configuration using COM and SCPI

Most of the following elements, highlighted in BLUE in the above diagram, have settings that are made using SCPI or COM commands. These are the same commands that are used to make settings in the [RF Path Configurator](#). In general the command specifies an element name and a setting.

[See SCPI command](#)

See [COM object](#) and [example](#).

Ref#	Element Name Description	Settings
<b>1</b>	<p>"IFSWn"</p> <p>For 2-port PNA-X, n = <b>A, B, R1, R2</b></p> <p>For 4-port PNA-X, n = <b>A, B, C, D, R</b> (for R1 to R4)</p> <p><b>For example: "IFSWB"</b></p> <p><b>Requires Opt 020 external IF inputs on the rear panel</b></p>	<ul style="list-style-type: none"> <li>• <b>"Internal"</b></li> <li>• <b>"External"</b> <a href="#">Rear Panel IF connectors</a>. 4-port PNA-X use R for Ref 1 to 4</li> </ul>
<b>2</b>	<p>"IFSigPathn"</p> <p>For 2-port PNA-X, n = <b>A, B, R1, R2</b></p> <p>For 4-port PNA-X, n = <b>A, B, C, D, R1, R2, R3, R4</b></p> <p>"IFSigPathAll" makes setting for ALL receivers.</p>	<ul style="list-style-type: none"> <li>• <b>"WBF"</b> Wide Band Filter Path (default)</li> <li>• <b>"NBF"</b> Narrow Band Filter Path</li> </ul>
<b>3</b>	<p>"NBFATNn"</p> <p>For 2-port PNA-X, n = <b>A, B, R1, R2</b></p> <p>For 4-port PNA-X, n = <b>A, B, C, D, R1, R2, R3, R4</b></p> <p><b>For example: "NBFATNB"</b></p>	<p>0 to 31 in 1 dB steps</p> <p><b>For example: "28"</b></p>
<b>4</b>	<p>"IFGaten"</p> <p>For 2-port PNA-X, n = <b>A, B, R1, R2</b></p> <p>For 4-port PNA-X, n = <b>A, B, C, D, R1, R2, R3, R4</b></p> <p><b>For example: "IFGateB"</b></p>	<ul style="list-style-type: none"> <li>• <b>"On"</b> Gate is always ON</li> <li>• <b>"Off"</b> Gate is always OFF</li> <li>• <b>"RearPanel"</b> (use <a href="#">Pulse IO pins 1 to 5</a>)</li> <li>• <b>"Pulse1"</b></li> <li>• <b>"Pulse2"</b></li> </ul>

		<ul style="list-style-type: none"> <li>• "Pulse3"</li> <li>• "Pulse4"</li> </ul>
5	<p>"IFAntiAliasFilter"</p> <p>This setting affects ALL receivers. It can NOT be made for individual receivers.</p>	<ul style="list-style-type: none"> <li>• "Auto" PNA selects which filter to use based on other IF settings.</li> <li>• "9MHZ"</li> <li>• "16MHZ"</li> </ul>
6	<p>"PulseTrigInput"</p> <p><b>Requires Opt 025 - Four Internal Pulse Generators</b></p>	<ul style="list-style-type: none"> <li>• "Internal" Internal Pulse In - pulse generators are triggered each period.</li> <li>• "External" External Pulse Synch In - <a href="#">Pulse I/O pin 7</a>) - An external trigger signal is required to trigger the pulse generators for each pulse.</li> </ul>
7	<p>"PulseModDrive"</p> <p>Select from 1 of 7 lines to modulate the OUT1 path of Sources 1 and 2 .</p> <p><b>Important:</b> When Pulse 1-4 is selected to modulate the sources, <a href="#">source leveling must be set to Open-loop</a>.</p>	<p>"On" Pulse Mod drive is always ON, leaving "SRC1 2 Out 1" ON and not modulated. Default setting.</p> <p>"Off" Pulse Mod drive is always OFF, leaving "SRC1 2 Out 1" OFF.</p> <p>"RearPanel" (use <a href="#">Pulse IO pin 8</a>)</p> <p>"Pulse1"</p> <p>"Pulse2"</p> <p>"Pulse3"</p> <p>"Pulse4"</p>
8	<p>"Src1Out1PulseModEnable"</p> <p><b>Requires Opt 021 - Source1 Pulse Modulator</b></p>	<p>"Enable"</p> <p>"Disable"</p>
9	<p>"Src2Out1PulseModEnable"</p> <p><b>Requires Opt 022 - Source2 Pulse Modulator</b></p>	<p>"Enable"</p> <p>"Disable"</p>

Last Modified:

8-Mar-2010 Modified for UI (A.09.20)

9-Dec-2009 Added external pulse input polarity and type

20-Jul-2009 Added subpointtrig content

5-Sep-2007 New Image and minor edits

February 5, 2007 MX New topic



## External Millimeter Module Configuration

---

This feature, when used with the N526xA test sets and external mmWave Modules, extends the frequency coverage of your PNA.

In this topic:

- [Features and Limitations](#)
- [How to Configure Millimeter Modules](#)
- [Mixer Mode](#)
- [MM Module Power Level Control](#)

### See Also

[N5261A / N5262A User Manual](#) (Requires internet connection).

[mmWave Measurement with No Test Set](#)

### Other IF Access Topics

**Note:** In the PNA user interface and in this help file, the N526xA Millimeter Head Controller is referred to as a **test set**. OML test head modules are referred to as **mmWave modules**.

**CAUTION:** Turn OFF test set power before connecting or disconnecting the DC cable to the mmWave modules.

### Features

- Controls N5260A, N5261A, and N5262A Test Sets.
- Compatible with [iTMSA \(True Mode Stimulus\)](#).
- Several methods available to provide [Leveled power to the DUT Input](#)
- Compatible with [Integrated Pulse Application](#)
- Allows mixer measurements using SMC. [Learn more.](#)

The following configurations are supported:

- PNA-X with options [Opt 020](#) works with N5260A, N5261A, and N5262A Test Sets.
- When using the N5262A test set, a 2-port PNA-X requires [Opt 551](#).
- [PNA 'C' models](#) with option H11 works with N5260A ONLY.



## Limitations

- **Power Settings** When using mmWave modules with an N5260A test set, the PNA can NOT control the power level into your DUT. Your mmWave modules may have a variable attenuator on them.

When used with an N5261A or N5262A, after performing a Source Power Cal, then the PNA power settings may be used to control the power into the DUT. See [Leveled Power Capabilities](#).

- To protect your mmWave modules from damage, the settings on this dialog can ONLY be changed manually. They can NOT be reset or changed by performing a Preset, by recalling an Instrument State, or from a remote program.

## PNA-X Notes

- **CAUTION:** Connect a 10 dB attenuator to the N5260A LO input from the [PNA-X LO Output](#). Otherwise, damage will occur to the N5260A test set.
- The PNA-X rear panel [IF Inputs use 5 SMA connectors](#). Previous PNA models use BNC connectors. Adapters may be required.
- Beginning with A.09.00, Frequency Offset and SMC Measurements are supported when using mmWave modules. [Learn more](#).

### How to Configure Millimeter Modules

Using front-panel HARDKEY [softkey] buttons	Using a mouse with PNA Menus
<ol style="list-style-type: none"><li>1. Press <b>SYSTEM</b></li><li>2. then <b>[Configure]</b></li><li>3. then <b>[Millimeter Module]</b></li></ol>	<ol style="list-style-type: none"><li>1. Click <b>Utility</b></li><li>2. then <b>System</b></li><li>3. then <b>Configure</b></li><li>4. then <b>Millimeter Module Config</b></li></ol>

There are NO programming commands to configure mmWave modules and test set.

## Millimeter Module Configuration dialog box help

**Note:** To protect your mmWave modules from damage, settings on this dialog can ONLY be changed manually. They can NOT be reset or changed by performing a Preset, or by recalling an Instrument State, or by a remote program.

**Available Configurations** Lists the Standard PNA configuration and others that you have created. Select **Standard PNA** to exit mmWave module operation.

**Selected Configuration** Shows the currently selected configuration. Edit this field to change the configuration name. Type a unique name using only alphanumeric characters and underscore.

**New** Click to create a new Millimeter Module configuration. A name is automatically selected. Edit the Selected Configuration field to change the configuration name.

**Remove** Deletes a Millimeter Module Configuration.

### Test Set Properties

**Selected Test Set** Select a test set to use in the current configuration. The firmware does NOT check to ensure that the selected test set is connected.

**Route PNA RF to rear panel "SW SRC OUT"** Available ONLY on PNA-X with [option 224 or 423](#) AND when using N5261A and N5262A test sets.

When checked, Port 1 source is switched to J11 and Port 3 source is switched to J8 on the [PNA-X rear panel](#). Use this configuration to quickly switch the RF Output back to the PNA front panel.

**Mixer Mode** Check to allow mixer testing using [SMC](#). [Learn more.](#)

**Enable Test Set RF ALC** Available for N5261A and N5262A ONLY. When checked, power is automatically leveled at the mmWave module RF input when using the standard cables and non-pulsed measurements. Clear this box to use non-standard cables or when making pulse measurements. When cleared, the following fields become available:

**Max Power Limit** The maximum mmWave module RF input is limited to this value when **Test Set RF ALC**

is OFF. When you exit this dialog box using **OK**, set the power out of the PNA using the [Power and Attenuator dialog](#).

**Power Offset** Sets the loss of the cables. The mmWave module RF input is adjusted by this amount. Positive offset increases the power.

**Power Slope** Helps compensate for cable and test fixture power losses at increased frequency. The mmWave module RF input power increases as the sweep frequency increases in dB/GHz. The slope is defined relative to the mmWave module RF input frequency. The slope starts at 0Hz and a positive slope will increase the power level. Range is +/- 2 dB/GHz.

**Important:** To set Test Port Frequency, first set the appropriate **Multiplier** values that are specified in your mmWave module documentation.

**Multiplier RF IN** RF Frequency Range (displayed in grey fields) multiplied by this value = test port frequency range.

**Multiplier LO IN** LO Frequency Range (displayed in grey field) multiplied by this value plus the IF frequency equals the test port frequency. The IF frequency is:

- 'C' Models = 8.333 MHz
- PNA-X models = 7.605 MHz

**Test Port Frequency** Set the Start and Stop frequencies of the selected configuration at the test ports. This becomes the displayed Start and Stop frequency of the PNA.

### Source

Click a button to launch the [External Devices dialog](#) where you can select an internal or external source to be used for the PNA LO source or PNA RF source.

**Cancel** Closes dialog box without saving changes.

**OK** Saves the configuration and the **PNA is Preset** before making the appropriate settings.

## Mixer Mode

Mixers measurements can be made at mmWave frequencies using [SMC](#). ([VMC](#) measurements are NOT supported.)

The Mixer Mode checkbox may be enabled ONLY when the number of PNA test ports match the number of ports on the mmWave test set. This means that an N5261A (2 port test set) must be connected to a 2-port PNA-X and an N5262A (4 -port test set) must be connected to a 4-port PNA-X.

The N5250A shortwave system has limited support for mixer mode.

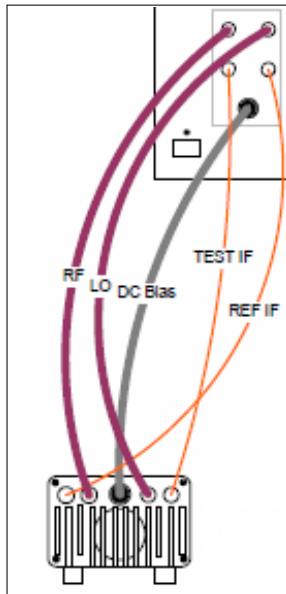
## Procedure

1. Connect your DUT to the mmWave system as described below.
2. Configure this dialog ([Millimeter Module Configuration](#)). Check **Mixer Mode**, then press **OK**. This presets the PNA.
3. [Create an SMC measurement](#).

4. [Make mixer settings](#). As with standard SMC measurements, only two DUT ports can be swept in frequency. The remaining DUT port must be a fixed frequency. [See configuration used for harmonic mixers](#).
5. [Increase power](#) for mmWave modules that are connected directly to a PNA port or external source.
6. Calibrate using the [SMC Calibration Wizard](#). When calibrating in waveguide and a Thru standard is not available, check **Separate power cal for input and output (do not use thru)** at the SMC [Calibration Setup](#) dialog.

## Hardware Connections

The following image shows the standard connections from a N5261A or N5262A test set port to an OML mmWave module.



mmWave mixers usually require that two of the three mixer ports operate at mmWave frequencies. When **Mixer Mode** is checked on this dialog ([Millimeter Module Configuration](#)), the following restrictions apply:

- On a 2 port mmWave system, only **port 1** of the test set can be used as a mmWave-frequency port; the test set port 2 can NOT be used.
- On a 4 port mmWave system, only **ports 1 and 3** of the test set can be used as mmWave-frequency ports; the test set ports 2 and 4 can NOT be used.
- The [SMC parameter](#) being measured must be within the frequency range of the PNA or within the frequency range of the banded mmWave module. Frequencies in between these ranges are allowed by the [SMC mixer setup dialog](#), but the measurement results on the screen are not accurate.

## Connections with a 4-port mmWave system -

### Upconverters

- DUT Input - Connect to PNA port 2 or port 4.
- DUT LO - Connect mmWave module to test set Port 3.

- DUT Output - Connect mmWave module to test set Port 1.

### Downconverters

- DUT Input - Connect mmWave module to test set Port 1.
- DUT LO - Connect mmWave module to test set Port 3.
- DUT Output - Connect to PNA port 2 or port 4.

### Connections with a 2-port mmWave system

Although supported, testing mmWave mixers with a 2-port system can be challenging for the following reasons:

- Testing mmWave mixers requires that two of the three DUT ports be at mmWave frequencies.
- Only test set port 1 is capable of adequately driving a mmWave module when used as a receiver.
- Therefore, the second DUT port that requires mmWave frequencies must have the mmWave module connected directly to an external source or a PNA second source.
- When using the mmWave module as a source, only the DC Bias and RF cable is necessary. The LO cable to the mmWave module is NOT used. This is because the RF input frequencies are multiplied in the mmWave module to provide the source frequencies. So a mmWave module used as a source can use the the RF cable to connect directly to the PNA second source or an external source. About +5 dBm of RF power is required to adequately drive the mmWave module.

### Downconverters - requires two mmWave modules as **sources**

- DUT Input - Connect the mmWave module to the test set port 1.
- DUT LO - Connect the RF cable of the mmWave module to an external source or the PNA (SRC2) second source.
- DUT Output - Connect to PNA port 2.

**Upconverters** - requires a mmWave module as a **source** at the DUT LO and a mmWave module as a **receiver** at the DUT Output:

- DUT Input - Connect to PNA port 2.
- DUT LO - Connect the RF cable of the mmWave module to an external source or the PNA (SRC2) second source.
- DUT Output - Connect the mmWave module to the test set port 1.

### Measuring Harmonic Mixers

Harmonic mixers have a multiplier circuit in the LO port of the DUT. The [SMC mixer setup dialog](#) in the following image shows the configuration for an 8x multiplier. This will provide the correct LO frequencies out of the

appropriate source.



## MM Module Power Level Control

These features can be used to provide leveled power to the input of your DUT.

- Receiver Leveling provides a sweep-to-sweep leveled power. [Learn more.](#)
- In addition, a special MM wave Source Power Calibration can be performed which measures and compensates the source power at many power levels using a calibrated PNA receiver.

### How to perform a MM wave Source Power Cal

Check [ALC Enabled](#) (if available) on the Millimeter Module Configuration dialog.

1. Press **Cal**, then **Source Power Cal** , then **Options** to launch the following dialog:



2. Do one of the following two steps depending on whether you have a power sensor that spans the frequency range of your measurement. Power sensors are rare above 110 GHz.
  - i. If you have a power sensor that spans the frequency range of your measurement, then...
    - a. Select **Use power meter for first iteration** to calibrate the PNA receiver.

- b. Connect the power sensor to the output of the MM module.
- ii. If a power sensor is NOT available, or if you prefer to use the Max power table for speed or convenience, then...
  - a. Create or use an existing Max power table to be used to calibrate the PNA receiver. See [How to build a Max power table / \\*.prn file](#).
  - b. Select **Use PNA receiver only**, then select **R1** as the receiver.
  - c. Check **Use millimeter power table**
  - d. Click **Power Table**, then navigate to the \*.prn file.
  - e. Click **OK**
- 3. Check **Produce receiver power calibration of PNA reference receiver**, then click **OK**.
- 4. On the Source Power Cal dialog, click **Take a sweep**. The output of the test set is set to Max power and a sweep is performed to calibrate the reference receiver.
- 5. Power is dropped for several subsequent sweeps. The calibrated reference receiver is used to fully characterize the source power.
- 6. The entire correction table **can** be saved along with the instrument state in a \*.csa file. [Learn how](#).
- 7. Power out of the input module should be flat and accurate.

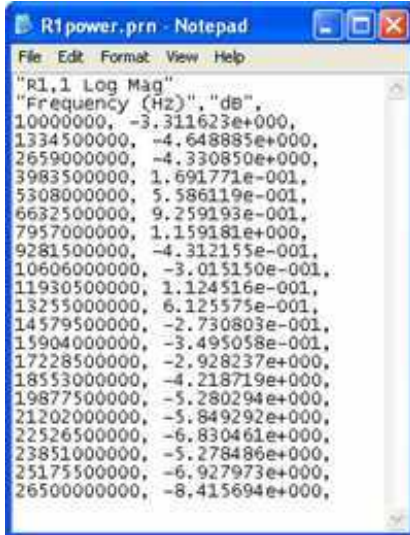
#### How to build a Max power table in a \*.prn file

This file contains the data that describes the output power of the module as a function of frequency. This is valid when the mmWave module is driven at high levels (+11 dBm). This file may have been created for you by a third party or shipped with you MM module. If not, you can create this \*.prn file from measured values using the following procedure:

- 1. Setup the measurement as you plan to make MM wave measurements with the module. This includes the MM Module Configuration dialog settings.
- 2. If a power sensor is available that spans the frequency range of your measurement, then do a source power calibration using a power meter for the first iteration. Otherwise, skip this step.
- 3. Set PNA power to the maximum value (+11 dBm).
- 4. Create an R1 trace and measure. [Learn how](#). The loss through the MM Module is small enough to not be considered.
- 5. Click **File** then **Save As** to save the R1 trace to a \*.prn file. Make note of the folder.
- 6. The file is recalled in the above procedure.

Now, this file can be used instead of connecting the power meter for this module.

This file can also be created manually, using a text file program such as Notepad. Copy the header information, and create the file with two columns, one for frequency and one for output power.



```
R1,1 Log Mag
"Frequency (Hz)", "dB",
10000000, -3.311623e+000,
1334500000, -4.648885e+000,
2659000000, -4.330850e+000,
3983500000, 1.691771e-001,
5308000000, 5.586119e-001,
6632500000, 9.259193e-001,
7957000000, 1.159181e+000,
9281500000, -4.312155e-001,
10606000000, -3.015150e-001,
11930500000, 1.124516e-001,
13255000000, 6.125575e-001,
14579500000, -2.730803e-001,
15904000000, -3.495058e-001,
17228500000, -2.928237e+000,
18553000000, -4.218719e+000,
19877500000, -5.280294e+000,
21202000000, -5.849292e+000,
22526500000, -6.830461e+000,
23851000000, -5.278486e+000,
25175500000, -6.927973e+000,
26500000000, -8.415694e+000,
```

### **Example .prn file**

Last modified:

9-Dec-2009	Added Cal note for waveguide
28-Aug-2009	Added Mixer Mode (A.09.00)
11-Mar-2009	Added Leveled power (A.08.5)
26-Feb-2009	Added Opt 551 requirement
14-May-2008	Major updates for PNA-X 8.2
9-Apr-2007	Updated for PNA-X
9/12/06	Note - NOT compatible with Freq Offset

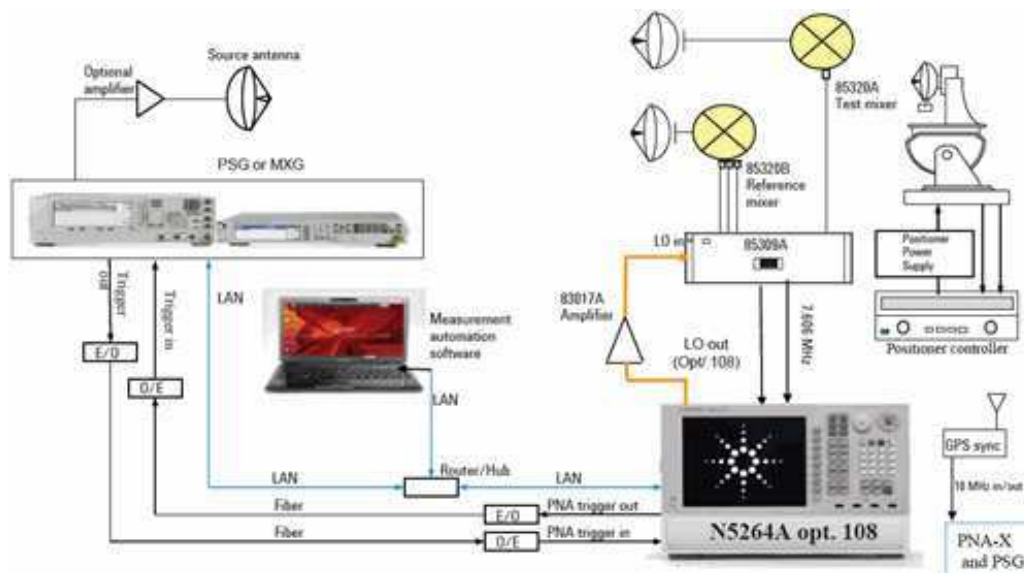


## N5264A Measurement Receiver

---

The N5264A is a PNA-X with no Source Ports or Tunable receivers. This makes the N5264A a very fast and very sensitive IF receiver that has been designed specifically for antenna and radar cross-section (RCS) measurements. When used with external sources, the N5264A can make frequency-scan measurements of antennas, or make RCS measurements in time domain.

- [N5264A Options and Limitations](#)
- **Configuring the N5264A**
  - [IF Receivers](#)
  - [Internal LO Source](#) (Opt 108)
  - [External Sources](#)
  - [How to Create Measurements](#)
  - [Calibrating the N5264A](#)
- **See Also**
  - [FIFO and other Antenna Features](#)
  - [Antenna Selection Guide](#)
  - 85309 Manual updated for PNA -P/N 85310-90002
  - [Antenna and RCS Measurement Configurations](#)
  - [N5264A Specs](#)



**Typical N5264A Application**  
**Far Field Outdoor Antenna Measurements**

## N5264A Options and Limitations

The N5264A has the following options:

- **IF Receiver** (base model) - The A,B,C,D, and R receivers are always set at the specified IF frequency.
- **IF Receiver + LO source** (Opt 108) - In addition to the receiver-only option, this option adds a 10 MHz to 26.5 GHz LO source.
- **Fast-Sweep / FIFO Mode** (Opt 118) - These features together allow you to make very fast measurements and save the data to a remote computer. [Learn more.](#)

## Limitations

The following PNA features are NOT available on the N5264A

- No Source ports
- No S-Parameters - [Arbitrary ratioed parameters ARE allowed.](#)
- [No Application support](#)
- [Limited Calibrations](#)

## Configuring the N5264A

### IF Receivers

- The five IF receiver inputs are on the N5264A [rear panel](#).
- All five IF receivers are measured at the same time. However, only those measurements that are displayed are updated.
- External mixers are always required to down-convert signals to the IF frequency.
- The default IF frequency for the N5264A is 7.605634 MHz.
- Change the IF frequency to any value between +14.9999 MHz and -14.9999 MHz using [SENS:IF:FREQ](#) (SCPI) or [IFFrequency](#) (COM) commands.

### Internal LO Source (Option 108)

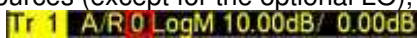
The LO source output connector is on the N5264A [rear panel](#).

The power level of the internal LO source is typically about +10 dBm and can NOT be changed.

To change the frequency of the LO Source:

1. Change the IF frequency of the measurement if necessary.
2. On the FOM dialog, change Receivers frequencies to the RF frequency range to be measured.
3. The LO Source frequency range is set automatically. This frequency value can not be viewed.

### External Sources

Because the N5264A has no internal sources (except for the optional LO), newly created measurements are displayed with a source port = 0 (zero). 

External sources can be configured so that they are controlled by the N5264A. [Learn how to configure an external source.](#)

Once configured, all existing and new measurements are changed to use the external source as the source port. When two or more sources are configured, the first configured source is displayed by default.

The external source settings can be changed from the following dialogs:

- [Power and Attenuators dialog](#) - Controls the ON | OFF state (set to Auto by default) and Power level of the external source.
- [FOM dialog](#) - Controls the frequency range. External sources are listed by name and uncoupled by default.

### How to Create Measurements

Create ratioed and unratioed measurements using the standard [Receivers tab](#).

S-parameters are not available.

### Calibration

The only calibrations available are Response, Source Power and Receiver calibrations.

These are performed from the [Unguided Cal Wizard](#).

Last Modified:

6-Apr-2009	Removed N5242A
23-Mar-2009	Removed Fast Sweep option. Put in Antenna Features topic.
28-Aug-2008	MX New topic

## Configure a Power Meter As Receiver (PMAR)

---

When a power meter is configured as a PNA receiver, you can...

- Extend the number of measurement receivers.
- Use the power meter as a scalar detector.
- Monitor the power at any point in a measurement system.
- Use the power meter to level the stimulus power at any point in a measurement system.
- Use the power sensor as a PMAR device to confirm the accuracy of a Source Power Cal. [Learn how](#).

The following Agilent Power Meters are supported as receivers:

- HP 437B / 438A
- E-Series power meter (E4418/9) and all supported sensors.
- P-Series power meters (N1911/2) and all supported sensors.
- EPM Series power meters (E4416/7) and all supported sensors.
- U2000 Series USB Power Sensors.

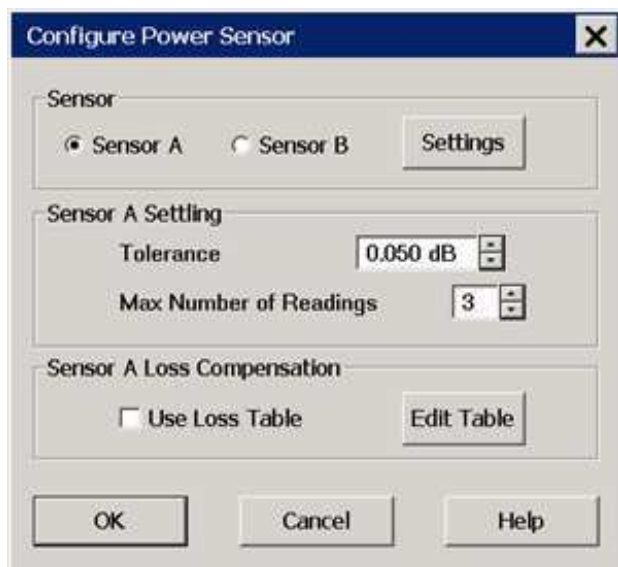
In general, a power meter that is configured as a receiver can be used like any other PNA receiver in the following dialogs:

- [New Trace dialog](#) - used in Ratioed and Unratioed measurements
- [Receiver Leveling](#)
- [Frequency Offset Mode](#) - Extend frequencies beyond PNA

### How to Configure a Power Meter as a Receiver

1. Perform a one-time configuration of the Power Meter as an External Device . [Learn how](#).
2. With the PMAR device selected in the Configuration dialog, click **Device Properties** to launch the following dialog:





### Power Sensor Configuration dialog box help

This dialog is used to configure a power meter / sensor for use as a receiver. To configure a single power meter for a Source Power Cal, use the [Power Meter Settings](#) dialog.

#### About Power Sensor Calibration

PMAR traces are NOT calibrated using standard PNA calibrations, including response corrections.

PMAR traces are calibrated using methods that are appropriate for the selected sensor. Follow the proper guidelines for zeroing or calibrating the sensors that are in use. Check to ensure that the selected sensor is appropriate for the frequency range and the power level at which PMAR measurements occur.

The PNA does not automatically prompt you to perform a calibration.

To calibrate a power sensor, click **Settings** on this dialog box, then click **Zero/Calibrate Sensor**. [Learn more](#).

**Note:** By default, a PMAR is de-activated when the PNA is Preset or when a Instrument State is recalled. This behavior can be changed with a [Preference setting](#).

#### Sensor

For power sensors that are connected to a power meter, select a sensor to configure.

**Settings** Click to launch the [Power Sensor Settings](#) dialog.

When pressed, communication with the sensor is tested. Sensor settings can NOT occur unless **Enable I/O** is checked on the [External Device Configuration dialog](#), and the sensor is properly connected and configured.

#### Sensor Settling

Each power meter reading is "settled" when either:

- two consecutive meter readings are within this Tolerance value **or**

- when the Max Number of Readings has been met.

The readings that were taken are averaged together to become the "settled" reading.

**Tolerance** When consecutive power meter readings are within this value of each other, then the reading is considered settled.

**Max Number of Readings** Sets the maximum number of readings the power meter will take to achieve settling.

### Sensor Loss Compensation

**Use Loss Table** Select this checkbox to apply loss data to Source Power calibration correction (such as for an adapter on the power sensor).

**Edit Table** Invokes the [Power Loss Compensation](#) dialog box.

Power Sensor Settings

Sensor B

Reference Cal Factor 100.0 %

	FREQUENCY	CAL FACTOR
1	10.000000 MHz	99.3 %
2	1.000000 GHz	99.1 %
3	2.000000 GHz	99.0 %
4	3.000000 GHz	98.9 %

Delete Cal Factor 1 Delete All

Load Cal Factors... Save Cal Factors...

☐ Limit Frequency Range

Minimum Frequency 10.000000 MHz

Maximum Frequency 100.000000 MHz

☐ Internal Zero ☒ External Zero

Zero/Calibrate Sensor

OK Cancel Help

### Power Sensor Settings dialog box help

This dialog appears when you click the **Settings** button on the [Configure Power Sensor](#) dialog.

**Note:** Be sure that the frequency range of your power sensor covers the frequency range of your measurement. This does NOT occur automatically.

**Sensor A (B)** Displays one of the following messages depending on type of sensor.

- **Not connected** The PNA is not detecting a power sensor.
- **Cal factors are contained within this sensor** Internal Reference Cal Factor and Cal Factor data are loaded automatically. The following settings do not apply.
- **Sensor Data** Allows the following entries for power sensor data:
  - **Reference Cal Factor** Specifies the Cal Factor for the 50 MHz reference signal.
  - **Cal Factor Table** Specifies the frequency and corresponding Cal Factor for the sensor.
  - **Delete Cal Factor** Deletes the indicated row in the table.
  - **Delete All** Deletes all data in the table.
  - **To Add a Row** to the table, click on a row in the table and press the down arrow on either the PNA front panel or keyboard. A row is added to the bottom of the table. The table is automatically sorted by frequency when OK is pressed.

### Limit Frequency Range

- Check to limit the use of the power sensor to those within the Minimum and Maximum frequency values.
- Clear to use the power sensor for all measurements. If the measurement frequency is not within the Minimum and Maximum frequency values, the closest min or max correction data is used for the measurement.

**Minimum Frequency** Specifies the minimum frequency range for the sensor.

**Maximum Frequency** Specifies the maximum frequency range for the sensor.

**Perform Sensor Zeroing and Calibration** Zero and/or calibrate the power sensor before measuring data. Follow prompts that may appear. Press **Zero/Calibrate Sensor** to perform required calibration steps.

If the following settings are 'greyed', Internal or External zeroing is selected automatically based on the power meter/sensor model.

- **Internal Zero** - A switch inside the power sensor removes the zero circuit from the incident power.
- **External Zero** - Requires that you physically remove the sensor from incident power.

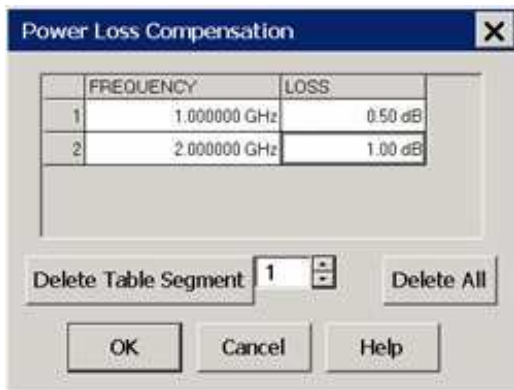


### Note for the U2000 Series USB power sensors

Select External Zero ONLY when the power to be measured is **below** the specified level. Otherwise, the U2000 series performs internal zeroing automatically when needed. See your power sensor documentation for more details.

- U200xA - below -30 dBm
- U200xH - below -20 dBm
- U200xB - below 0 dBm

If your U2000 power sensor 'hangs' when external zeroing, upgrade the power sensor firmware to Rev. A.01.02.00 or higher to fix this problem.



### Power Loss Compensation dialog box help

**To Add a Row** to the table, click on a row in the table and press the down arrow on either the PNA front panel or keyboard.

**To Edit a value**, double-click in the cell to be edited.

These values can be loaded from an S2P file using the [Characterize Adaptor Macro](#).

Compensates for losses that occur when using an adapter or coupler to connect the power sensor to the measurement port. These components will be removed when the calibration is complete. To account for components that will remain during the measurement, use the [Power Offset setting](#).

The Frequency / Loss pairs define the amount of loss for the entire frequency range. For example, using the entries in the above dialog image:

- 0.5 dB is used to compensate power sensor measurements up to 1 GHz.
- Each data point between 1 GHz to 2 GHz is linearly interpolated between 0.5 dB and 1 dB.
- 1 dB is used above 2 GHz.
- A single frequency/loss segment is applied to the entire frequency range.

Enter up to 100 segments to achieve greater accuracy.

**Frequency** Enter a frequency in Hz.

**Loss** Enter a loss as a POSITIVE value in dB. To compensate for gain, use NEGATIVE values.

**Delete Table Segment** Deletes row indicated in the field.

**Delete All** Deletes all data in the table.

The Power Loss Compensation table survives PNA Preset and Power OFF. To NOT use Loss compensation, clear the Use Loss table checkbox on the [Configure Power Sensor](#) dialog.

### Use a PMAR Device to confirm a Source Power Cal

After a Source Power Cal has been performed, use the same sensor as a configured PMAR to analyze the accuracy of the Calibration.

1. Create a PMAR device with the power sensor that will be used for the Source Power Cal.
2. Perform a Source Power Cal. [Learn how](#).
3. Create an unratiod measurement with the PMAR device. [Learn how](#).
4. With the power sensor still connected to the test port, monitor the corrected source power using [Min and Max markers](#) or the [Trace Statistics peak-to-peak](#) feature.

---

Last Modified:

25-Aug-2009    MX New topic

## E5091 Test Set Control

---

The E5091A is a popular Agilent Technologies 7-port / 9-port test set. Although the test set was originally designed to work with the ENA Network Analyzer, it also works well with the PNA. This topic describes how to control the test set from the PNA. For more information about the test set, refer to your E5091A documentation.

- [Overview](#)
- [Connecting the E5091A](#)
- [How to make E5091A test set Control Settings](#)
- [Calibrating with the E5091A](#)

### Other System Configuration Topics

## Overview

When connected to the PNA, the E5091A test set provides full 7-port or 9-port test capability. The E5091A can be configured to switch a different test set path for each PNA channel. When all channels have been configured, the entire measurement setup and calibration can be [saved to a .cst or .csa file](#) to be recalled later. In addition, the [Channel Settings Table](#) that is appended to a printed hardcopy of a measurement includes the E5091A Port Control settings.

### Notes:

- ONLY the 7-port and 9-port test sets are supported with the PNA.
- Works with all 4-port PNA models.
- The E5091A test set has a maximum useful frequency of 11 GHz.
- The E5091A test set Control can be automated using [SCPI](#) and [COM](#) commands.
- When [enabled](#), a second status bar row appears which indicates the test set that is being controlled and the current switch state.
- Test set path switching occurs just before a channel is triggered. If a [channel trigger state is Hold](#), switching for that channel does not occur.
- PNA sweep speed will be slightly slower when using the E5091A to switch measurement paths.

## Connect and Configure the E5091A

The E5091A can be connected to any one of the PNA USB ports. When first installed, Windows will automatically launch the "Add New Hardware" wizard. Click **Next** to install the E5091A test set.

**Note:** See the [power handling limitations](#) of the PNA USB ports.

Connect the PNA test ports to the E5091A test ports. Match PNA test port 1 to E5091A test port 1, and so forth.

The PNA can control up to two E5091A test sets. Set the Instrument ID bit switch to 1 or 2. The test sets will then be identified automatically and referred to by the DIP switch setting on the E5091A rear-panel. Change the ID bit switch setting before connecting to the PNA USB.

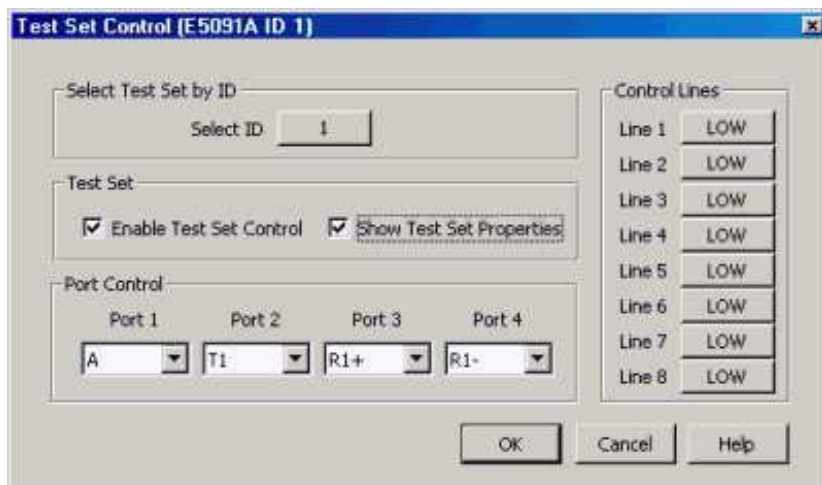


## How to make E5091A test set Control Settings

## Using a mouse with PNA Menus

1. Click **Trace/Chan**
2. then **Channel**
3. then **More**
4. then **External test set**
5. then **E5091A**

**www.valuetronics.com**



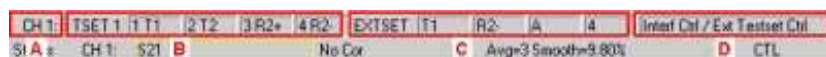
### E5091A test set control dialog box help

The title of the dialog shows the test set model and ID number of the active test set..

**Select ID** ID of the test set to be configured. Up to two E5091A test sets can be controlled. Click to change test set ID. [Learn how to set the test set ID.](#)

**Enable Test Set Control** When cleared, port switching and control line settings are disabled. This selection affects all channels using the selected test set.

**Show Test Set Property** When checked, a second row on the status bar appears which indicates the test set that is being controlled and the current port control selection. For example, the following image shows the status bar when controlling an E5091A test set.



- A. Configured channel
- B. Port Control settings for E5091A
- C. Port Control settings for Z5623A K64
- D. Test set Label. E5091A control does not use this field. It is shared between [Interface Control](#) and [External test set Control](#). The two labels are separated by /.

Control of the second status bar is completely separate from the first status bar, which is controlled from the [View, Status Bar](#) menu.

**Port Control** Controls **mapping** of Physical ports to Logical ports.

- Physical ports are the port numbers that are labeled on the test set front panel. ([see N44xx test sets](#))
- Logical ports are the port numbers that are referred to by most of the PNA application prompts and dialog boxes.

### Port Mapping Notes

- Port Control and Control Line settings affect the channel of the active (selected) measurement. These settings will occur as the channel is being measured.
- Correction is NOT turned OFF when port mappings are changed. However, the **calibration is NO LONGER VALID!**

**Control Lines** Specifies the values of individual control lines. These general purpose control lines on the test set front-panel can be used in your test setup. Each button toggles the control line HIGH and LOW. When first opened, the selections reflect the current control lines. See your test set documentation for more information about the control lines.

**OK** When clicked, the changes to the dialog box are implemented and the port selections and control values are immediately sent to the specified test sets. The Port Control and Control line settings are stored with other channel data and used when those channels are swept.

**Cancel** (or Escape) Changes to the dialog are not implemented and revert to the settings before the dialog box was opened.

## Calibrating with the E5091A

The following are a few changes in the way you calibrate the PNA with the E5091A connected:

1. Create the measurements for the channel and configure the Port Control (switching) on the E5091A Test Set Control dialog box. Enable **Show Test Set Property**.
2. To calibrate, start the Calibration wizard and select a Calibration method (ECAL, SmartCal, Unguided).
3. Select the DUT connectors that are used at the E5091A measurement reference plane.
4. When prompted to connect a standard to a PNA port, instead connect the standard to the E5091A port as indicated on the test set status bar. For example, when the cal wizard prompts to connect the standard to port 1, if the status bar indicates **1 A**, the connect the standard to port A of the E5091A.

---

Last Modified:

7-Apr-2009	Added 7/9 port only, and 4-port PNAs
4-Sep-2008	Removed legacy content
18-Jun-2007	MX added UI

## External Multiport Test Set Control

---

- [Supported Test Sets](#)
  - [Option 551](#)
  - [Option 550](#)
  - [E5091A](#) (separate topic)
- Procedure
  1. [Connect Test Set](#)
  2. [Restart as Multiport](#)
  3. [Optional External Test Set Control Settings](#)
- [External Test Set Control and other PNA Functions](#)

### Other System Configuration Topics

## Supported Test Sets

The list of test sets that provide integrated solutions with the PNA is constantly growing. For a current list of supported multiport test sets, see [www.agilent.com/find/multiport](http://www.agilent.com/find/multiport)

### Option 551

- **With** Option 551 enabled on your PNA, **any supported multiport test set** (such as the U3042A E12) can be controlled directly from the PNA to make fully integrated measurements at ALL of the available test ports. To understand what test ports are available to source and receive, see the test set documentation.
- **Without** Option 551, basic operation depends on the number of PNA test ports.
  - For a 2-port PNA, configure two available test ports.
  - For a 4-port PNA, configure four available test ports.

### Option 550

- **With** Option 550 enabled on your PNA, the N44xx test sets and other supported test sets can be controlled directly from the PNA to make fully integrated **4-port** balanced measurements.
- **Without** Option 550, basic operation allows you to configure any **two** of the test ports to make standard S-parameter or receiver measurements. A different pair of ports can be configured for each channel.

N44xx Test Set Model	PNA Model (must have front-panel loops)	Frequency Range
N4419B	E8362B and N5230A Opt 225	10 MHz to 20 GHz
N4420B	E8363B and N5230A Opt 425	10 MHz to 40 GHz
N4421B	E8364B and N5230A Opt 525	10 MHz to 50 GHz
N4421B Opt H67	E8361A	10 MHz to 67 GHz

**Note:** By default, the system logical test ports are mapped as follows:



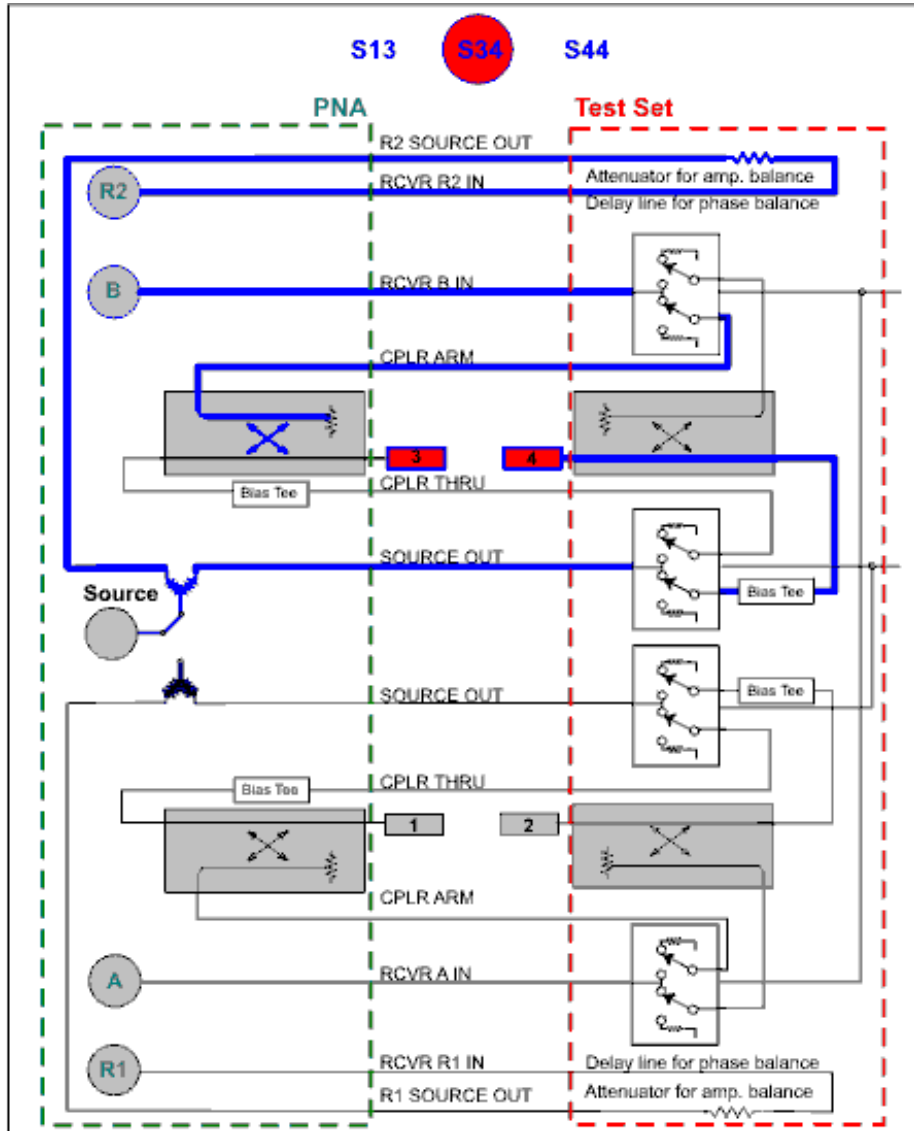
- **Port 1** - PNA port 1
- **Port 2** - Test Set port 2
- **Port 3** - PNA port 2
- **Port 4** - Test Set port 4



The ports can be **remapped** using the [Port Control Setting](#).

## Block Diagram of a 2-port PNA and N44xx Test Set

Click one of the S-parameters to see switch and path changes. Because the test set does NOT contain receivers, [measurement speed](#) and [calibration](#) can be affected.



## Procedure - How to enable full Multiport Capability

1. [Enable Option 550 or 551.](#)
2. Connect the test set to the PNA using the documentation that was shipped with the test set.
3. [Restart as Multiport PNA](#)

#### 4. [Make optional External Test Set Control Settings](#)

### Connect and Configure the Test Set

Connect the test set to the PNA using the test set documentation. Most test set documentation can found at [www.Agilent.com](http://www.Agilent.com)

#### Test Set I/O-controlled test sets

Test sets that are controlled using the [Test Set I/O connector](#), have NO return communication capability. The PNA sends commands out the rear panel connector. It is assumed that the test set is responding appropriately. The "Active" LED, located on the test set front panel, should light when the test set is addressed in Multiport Mode or manual operation. When the test set is not in use, the Active LED will be OFF.

#### GPIO-controlled test sets

Connect the test set to the GPIB using one of the following methods:

- If the **PNA will NOT be controlled** by a remote computer using GPIB, then the test set can be connected directly to the PNA GPIB port. The PNA is automatically switched to [System Controller](#) mode.
- If the **PNA WILL be controlled** by a remote computer using GPIB, [then learn how to connect the test set](#)

### Restart as Multiport PNA

#### How to Enable Multiport capability

**Note:** If [Option 550](#) or Option 551 has not been enabled, the following **Multiport Capability** menu selection will NOT be available.

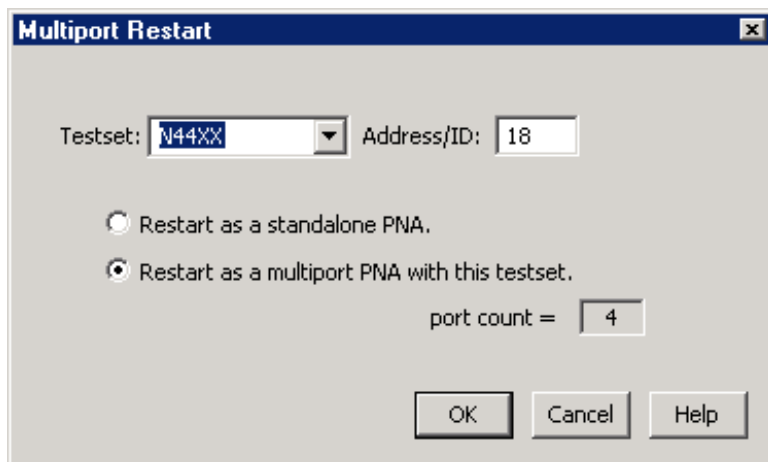
##### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Configure]**
3. then **[Multiport Capability]**

##### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Configure**
4. then **Multiport Capability**

[Programming Commands](#)



### Multiport Restart dialog box help

**See Also** [External Test Set Control and other PNA Functions](#)

After the test set is connected and [PNA Option 550 or 551 is enabled](#), the following settings are used to enable Multiport operation.

**Test Set** Select the test set file to load. Only the files that are appropriate for use on that PNA model are displayed.

#### To Add a new Test Set file:

1. On the PNA, click **File**, then **Exit** to quit the PNA application.
2. Download the Test Set file from: <http://na.tm.agilent.com/multiport/testsets/>
3. Save it to the PNA C:/program files/agilent/network analyzer/testsets
4. Start the PNA application.
5. Click **Utility**, then **System**, then **Configure**, then **Multiport Capability**.
6. In the Multiport Restart dialog, click **Restart as multiport PNA with this testset**.
7. The new test set should now be visible from the **Testset:** menu.

**Address** Enter the test set address if the test set is connected to GPIB. Connections over the Test Set I/O connector are determined by their sequence.

- **Restart as a standalone PNA.** The PNA shuts down and restarts as a standard PNA. If the test set is left connected to the PNA, switch the test set OFF, then back ON to ensure that the test set routes signals to ports 1 and 2 of the PNA. In this condition, there is more loss in the test paths than without a test set connected. If the power switch is OFF, there is SIGNIFICANTLY more loss in the test paths.
- **Restart as a multiport PNA with this testset.** The PNA shuts down and restarts as a multiport PNA with the selected test set.

Click **OK** The PNA shuts down and restarts in the selected configuration.

To learn how to change port mapping, see [Port Control](#).

## Problems

If the PNA cannot find the test set, the following error is displayed on the PNA:

**GPIB ERROR Address xx cannot open VISA session.**

To correct the problem, verify the following:

- The test set is connected to the PNA using [one of the methods described above](#).
- The correct test set address is set.
- The test set is turned ON.

**Important: After the problem has been fixed:**

1. On the External Test Set Control dialog, click [Enable Test Set Control](#).
2. Restart Triggering - click **Sweep, Trigger, Continuous**.
3. The PNA again tries to find the test set.

## External Test Set Control Settings

The following External Test Set Control Settings are used to configure Multiport test sets. For the N44xx test sets, the only setting that is necessary is port control.

### How to access the External Test Set Control Settings

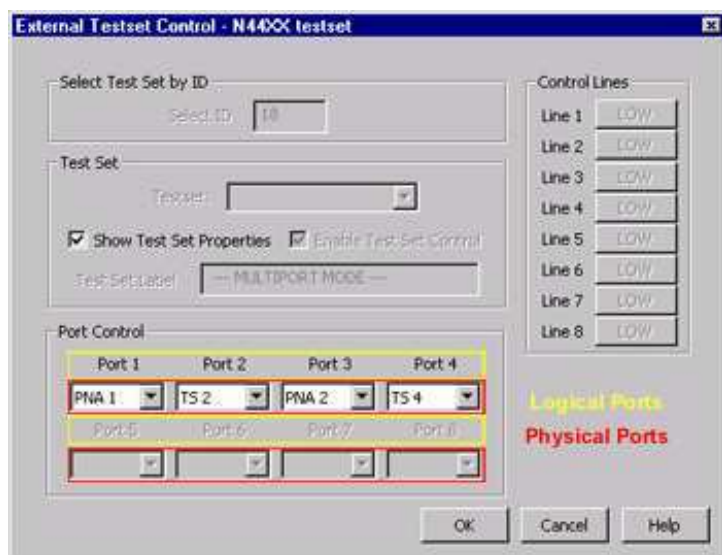
#### Using front-panel HARDKEY [softkey] buttons

1. Press TRACE/CHAN
2. then **[Channel]**
3. then **[Hardware Setup]**
4. then **[External Testset]**
5. then **[Other External Test Sets]**

#### Using a mouse with PNA Menus

1. Click **Trace/Chan**
2. then **Channel**
3. then **Hardware Setup**
4. then **External Testset**
5. then **Other**

◀ **Programming Commands** ▶



## External Test Set Control dialog box help

### Important Notes:

- With Options 500 and 551, **first** use the [Multiport Restart](#) dialog to **Restart as Multiport PNA with this test set**. The test set file is loaded and the test set is enabled automatically.
- When using GPIB to control an external test set, the PNA is automatically put in [System Controller mode](#). It can NOT also be in talker-listener mode. To have the PNA control a GPIB test set AND be controlled by a remote PC, use a [USB to GPIB adapter](#) to control the external test set. This does NOT apply for PNA models with a [1.1 GHz CPU board](#), which has both a GPIB controller port and a talker/listener port.
- See also [External Test Set Control and other PNA Functions](#)

### Select ID

- For N44xx test sets: the GPIB address
- Multiport test sets: either GPIB address or 0 for Test Set I/O controlled test sets.

**Enable Test Set Control** When cleared, port switching and control line settings are disabled. This selection affects all channels using the selected test set. When checked, the 'Show Test Set Properties' checkbox is also checked automatically.

**Load Test Set File** For operating **without** Options 500 or 551.

If your Test Set is not visible, see [Add a new Test Set](#).

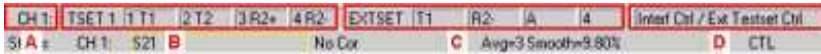
The selected test set file is loaded.

1. Navigate to the folder: C:/Program Files/Agilent/Network Analyzer/testsets/
2. Select a test set control file.

The title of the dialog shows the model of the test set file that is currently loaded.

[See a list of supported test sets.](#)

**Show Test Set Properties** This box becomes checked by default when the Enable Test Set Control is checked. When checked, a second row on the [status bar](#) appears which indicates the test set that is being controlled and the current port control selection. For example, the following image shows the status bar when controlling an [E5091A](#) test set and a Z5623A K64 test set:



- A. Configured channel
- B. Port Control settings for E5091A
- C. Port Control settings for Z5623A K64
- D. Test Set Label. This field is shared between [Interface Control](#) and External Test Set Control. The two labels are separated by /.

Control of the second status bar is completely separate from the first status bar, which is controlled from the [View, Status Bar](#) menu.

**Test Set Label** NOT available with options 550 and 551. Add text to appear on the second status bar when **Show Test Set Properties** is checked. See image above.

**Port Control** Controls **mapping** of Physical ports to Logical ports. ([Refer to image of dialog box above.](#))

- Physical ports are the port numbers that are labeled on the test set front panel.
- Logical ports are the port numbers that are referred to by most of the PNA application prompts and dialog boxes.

#### Port Mapping Notes

- Port Control and Control Line settings effect the channel of the active (selected) measurement. These settings occur as the channel is being measured.
- Correction is turned OFF when port mappings are changed.
- After the physical ports are mapped to logical ports, all PNA references to port numbers refer to LOGICAL port numbers. The only exception to this is during calibrations.

**Control Lines** For use with the U30xxA test sets. Specifies the values of individual control lines. These general purpose control lines on the test set front-panel can be used in your test setup. Each button toggles the control line HIGH and LOW. When first opened, the selections reflect the current control lines. See your test set documentation for more information about the control lines.

**OK** When clicked, the changes to the dialog box are implemented and the port selections and control values are immediately sent to the specified test set. The Port Control and Control line settings are stored with other channel data and used when those channels are swept.

**Cancel** (or Escape) Changes to the dialog are not implemented and revert to the settings before the dialog box was opened.

---

## External Test Sets and other PNA Functions

The following features may work differently with a test set connected to the PNA.

### Remote Commands

See [SCPI](#) and [COM](#) commands for controlling an External Test Set.

Use [\\*OPT?](#) (SCPI) or [NumberofPorts](#) (COM) to query the number of ports for a PNA/External Test set.

### Interface Control

When both [Interface Control](#) and External Test Set Control are configured, the commands on the Interface Control **Before Sweep Start** tab are sent out before any External Test Set Control commands are executed on that channel. Similarly, commands on the **After Sweep End** tab are sent after Test Set Control commands.

### Calibration

With an External Test Set connected, calibration is performed exactly like a PNA with the following exceptions:

- Correction is turned OFF when port mappings are modified. This also applies to Source Power Cal.
- Beginning with PNA Rev. A.07.50, for [TRL Cal](#), [QSOLT](#), or [Unknown Thru](#) cals with external test sets:
  - Use of a [Delta Match Calibration](#) from a User CalSet is NOT required. However, for a 4-port PNA-L a [Global Delta Match Cal](#) must be present. The Global Delta Match Cal can only be performed in stand-alone mode.
  - You can NOT perform any of those 3 cal types on JUST a pair of ports that share a test port receiver, such as Port 1 and Port 2 of a [4-port system](#). You would need to include an additional port in the calibration.
- With an External Test Set connected, you may be required to perform more than [3 THRU connections](#).
- A test set such as the Z5623A K44 which is used with 4-port PNA models, does not terminate ports that are not currently in the source path. Because a ports load match on this system is not constant for all possible ports it can be paired with, when calibrating more than two total ports it may be necessary to make Thru measurements on more than the usual minimum number of Thru paths for a PNA calibration. The PNA will ensure that multiport calibrations use a sufficient set of Thru paths so that the calibration can correct for those variations in load match on this type of multiport system.
- As with ALL PNA calibrations, when error correction is ON, both forward and reverse sweeps are required for EACH port pair that is corrected, even if only one reflection measurement is displayed. For example, any displayed measurement with full 4-port calibration ON will require 12 measurement sweeps. [Learn more.](#)

### Source Power Cal

[Source power calibration](#) involves adjusting the source so that the power at an output port is flat across a frequency range. Because of additional loss through some of the test set paths, it may NOT be possible to obtain corrected output power because of limitations on the source signal.

During a Source Power Cal, you are prompted when and where to connect the power sensor. When one of the supported test sets are connected, the prompt refers to the PHYSICAL port number, NOT the LOGICAL port

number. To help with translating physical to logical port mappings, enable [Show Test Set Properties](#).

### Measurements with Shared Receivers

External test sets do not contain receivers. The PNA receivers are always used to measure signals at the external test set ports. Therefore, when a channel contains two measurements that share a PNA [test port receiver](#), additional sweeps are necessary.

For example, to make S34 and S44 measurements in the same channel with correction OFF:

- On a 4-port PNA, only ONE sweep is required using the C (port 3), D (port4), and R (reference for All receivers).
- On a N44xx system, TWO sweeps are required since both measurements use the B and R2 receivers. [See interactive block diagram above.](#)

### Create Ratioed and Unratioed Measurements

When using an external test set, it IS possible to create a Ratioed measurement using two logical receivers that share the same physical PNA receiver. However, this measurement data is NOT valid. Invalid measurement traces show all data at -200 dB (in Log mag format). [Learn about Logical Receiver Notation](#)

### Logical Port References

When an external test set is enabled, all references to PNA port numbers and test set port numbers (except during calibrations) refer to LOGICAL port numbers. Logical ports can be remapped using the [Port Control settings](#). During a calibration, you are prompted to connect standards to physical port numbers.

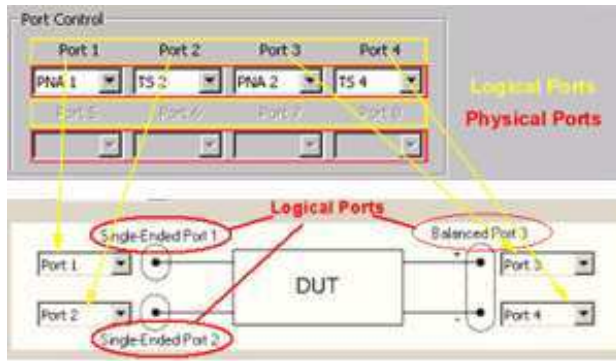
### Balanced Port Mapping

"Logical Ports" is a term that is used with both External Test Sets and balanced measurements. While the concept is the same, they refer to different scenarios. The two can be easily confused when making Balanced measurements with an External Test Set connected. The important principle to remember is the order in which the logical ports for each are mapped:

1. In the [External Test Set - Port Control settings dialog](#), the physical PNA ports and test set ports are mapped to logical ports as noted above.
2. In the [Balanced Topology Dialog](#), the new (step 1) logical ports are mapped again to become Balanced logical ports.



1.  
External  
Test Set  
Mapping  
Dialog



2.  
Balanced  
Topology  
Dialog

## Preset

[Instrument Preset](#) will reset [Port Control](#) settings to defaults and remove the [test set label](#). All other settings remain. To maintain port control settings and the test set label, create a [User Preset](#).

## Instrument State Save and Recall

[Instrument State files](#) include Test Set model, Enable and Status bar settings, and Port mappings and DUT control values for each channel.

If an Instrument State recall requires that a test set configuration file be loaded, recall time may be significant. For example, this would occur if a 2-port PNA with attached test set is configured as a 2-port PNA and then recalls a state file which requires 4-port operation.

## Recall Cal Sets

If a Cal Set is saved while an external test set is enabled, when the Cal Set is recalled, then the external test set must be enabled or an error message is displayed.

## Copy Channel

[Copy Channel](#) copies all relevant test set data from the source channel to the target channel.

## Applications

No PNA applications are supported with External Test Set Control. These include FCA (opt 083), SMC (opt 082), GCA (opt 086), NFA (opt 029), Pulsed (opt H08).

## Print

Port mapping information appears on the [Channel Settings Table](#) when printing.

## Save sNp Files

To save sNp data with an [external test set](#) enabled, click File, [Save As](#), then select **Snp File(\*.snp)**, then complete the [Choose ports dialog](#).

---

Last modified:

21-Apr-10	Updated per RD
23-Nov-2009	Edited Cals for 7.5
12-Sep-2008	Added K44 image and cal note
4-Sep-2008	Removed legacy content
15-Jan-2008	No App support
19-Nov-2007	Note to Add TS files
9-Nov-2007	Fixed S13 image and added delta match notes
17-Sep-2007	Added note for PNA-X test set files
23-Feb-2007	Modifed DM Cal for 6.20.
15-Jan2007	MX Added UI
10/16/06	Added clarification to opts.
9/18/6	MQ Many edits for Opt 551
9/12/06	Added link to programming commands

# Display Colors

You can modify the colors that are used to draw various elements on the PNA screen and on a hardcopy print of the display.

## See Also

[Print Preview](#)

### How to modify DISPLAY Colors

These settings can also be accessed from the [Preferences dialog box](#).

Using front-panel HARDKEY [softkey] buttons	Using a mouse with PNA Menus
<div><div>1. Press <b>DISPLAY</b></div><div>2. then <b>[More]</b></div><div>3. then <b>[Display Colors]</b></div></div>	<div><div>1. Click <b>Response</b></div><div>2. then <b>Display</b></div><div>3. then <b>Display Colors</b></div></div>

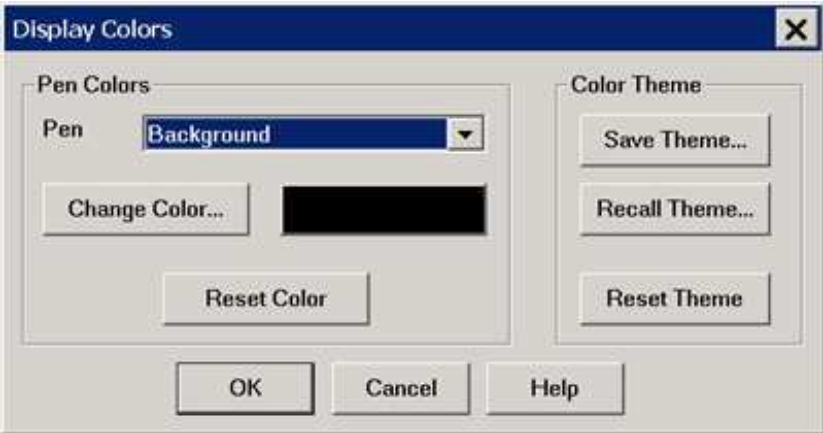
### How to modify PRINT Colors

<div><div>1. Press <b>PRINT</b></div><div>2. then <b>[Print Colors]</b></div></div>	<div><div>1. Click <b>Utility</b></div><div>2. then <b>Print</b></div><div>3. then <b>Print Colors</b></div></div>
---	--

◀

Programming Commands

▶



## Display and Print Colors dialog box help

The Display Colors and Print Colors dialog boxes function in exactly the same manner. See [Print Preview](#) procedure below.

### Pen

"Pen" is a term used to describe the various elements. Each pen can have a unique color.

You can change the color of the following pens:

- Background - The background color of the inactive windows.
- **New** Active Background - The background color of the active window.
- Grid - The inner lines of all grids in all windows, and the grid frame in inactive windows.
- Active Labels, Grid Frame - The labels and grid frame colors in the active window. **Note:** when this pen is selected, the current window becomes inactive. Therefore, changes for this pen color will not be visible until **OK** is pressed.
- Inactive Window Labels
- Failed Trace - [Limit Line](#) failed traces or failure indicators (dots) and the word Fail.
- The following pens for up to 8 Traces:
  - Data and Limits
  - Memory trace
  - Markers
  - Memory markers

### About Trace Pens

'1st Trace' is NOT always Trace1 (**Tr1**). For example, the first trace in a window might be **Tr2** which is drawn with the "1st Trace" pen.

The first 8 traces are drawn with the defined pen colors. The next eight traces reuse the same colors, and so forth. For example, if all traces are numbered sequentially, the 9th and 17th traces are drawn using the same color as the 1st trace.

**Change Color** Click the button or the color swatch to launch the [Change Color](#) dialog.

**Reset Color** Restores the default color for the selected pen.

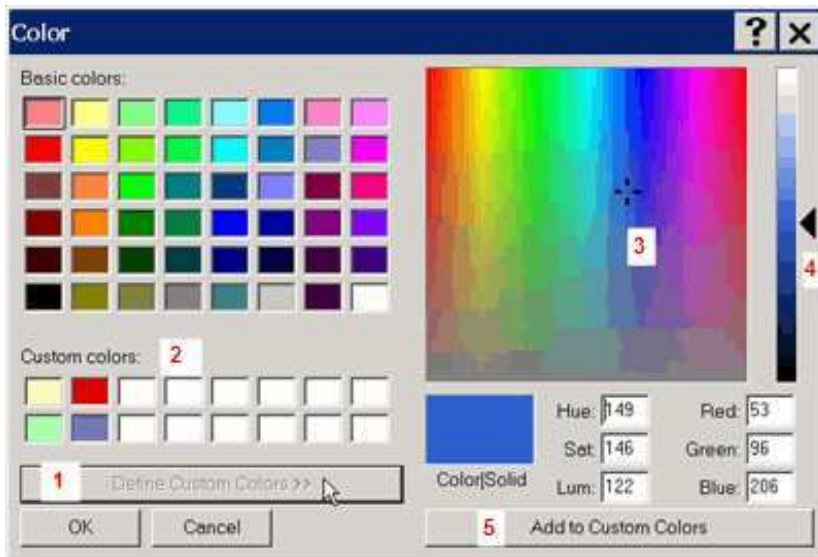
### Color Themes

A theme is a complete set of pens and their colors. The current theme persists until you change it. Themes can also be saved to a file and then later recalled.

- **Save Theme** Click to save the current set of pens to a file.
- **Recall Theme** Click to recall and use a saved theme.

- **Reset Theme** Click to recall the default PNA color theme.

The colors for the following Display elements can NOT be changed: toolbars, softkeys, menus, dialogs and popup messages.



#### Change Color dialog box help

To use a basic color, click the color from the 'Basic colors' palette, then click **OK**.

To define and use a custom color:

1. Click **Define Custom Colors>>** to open the right side of the dialog.
2. Optionally, pick a Custom color slot to replace. Otherwise, the replacement will occur at the first slot location and continue with subsequent custom color definitions.
3. Click the color pane, or drag the crosshairs, to the location of the custom color.
4. Drag the arrow to the desired saturation level of the custom color.
5. Click **Add to Custom Colors**
6. Continue to define more colors, or click **OK** to close the Color dialog.

After a custom color has been assigned to a PNA pen, the custom color can be changed. The PNA pen color remains unchanged.

#### Print Preview Procedure

Use the following procedure to preview your Print Colors on the PNA screen:

1. From the Print Colors dialog, select **Reset Theme** then **Save Theme**. Name the new theme "MyPrintTheme.colors". This will give you a starting point equal to the default print colors.
2. Launch the Display Colors dialog, select **Recall Theme**, then select "MyPrintTheme.colors". The display will now show the default print theme.
3. Customize the display colors. You will be previewing how the hardcopy will appear when printed.
4. Save the customized display colors to "MyPrintTheme.colors".
5. Go to the Print Colors dialog and Recall "MyPrintTheme.colors".

---

Last Modified:

22-Feb-2010    Added Active background (A.09.20)

7-Aug-2009    MX New topic

## Mechanical Devices

---

- [Overview](#)
- [How to access Mechanical Devices settings](#)
- [Mechanical Devices dialog](#)

### Other System Configuration Topics

## Overview

**Note:** To prevent premature wear, the PNA does not allow attenuators or other mechanical switches to switch continuously.

These mechanical devices are set for the entire channel. When more than one channel is used, and a mechanical device setting is NOT the same for all channels, only the ACTIVE channel is allowed to sweep. All other channels are **Blocked** - NOT allowed to sweep. Blocked channels will resume sweeping when they are made ACTIVE, or when the conflict is resolved.

Press **TRIGGER** then **[Restart]** to cause ALL channels to sweep once. Then the active channel will resume sweeping continuously.

Before PNA Rev. A.09.20, conflicting channels were put in trigger Hold.

The Mechanical Devices dialog shows the settings of all of the switches and attenuators in the PNA. The settings for all active channels are shown side-by-side for easy comparison. This dialog allows you to determine the settings which would cause mechanical devices to switch between states on consecutive sweeps, potentially leading to device wear-out. It also allows you to determine if the conflict can be resolved to enable continuous sweeps on all channels.

The following are the mechanical devices that are potentially shown in the dialog. These components may not appear in your PNA model:

- Port 1 through Port 4 Bypass Switches
- Port 1 through Port 4 Source Attenuator settings
- Receiver A through Receiver R Attenuator settings
- Port 1 Noise Tuner Switch and Port 2 Noise Receiver Switch

## How to access Mechanical Devices settings

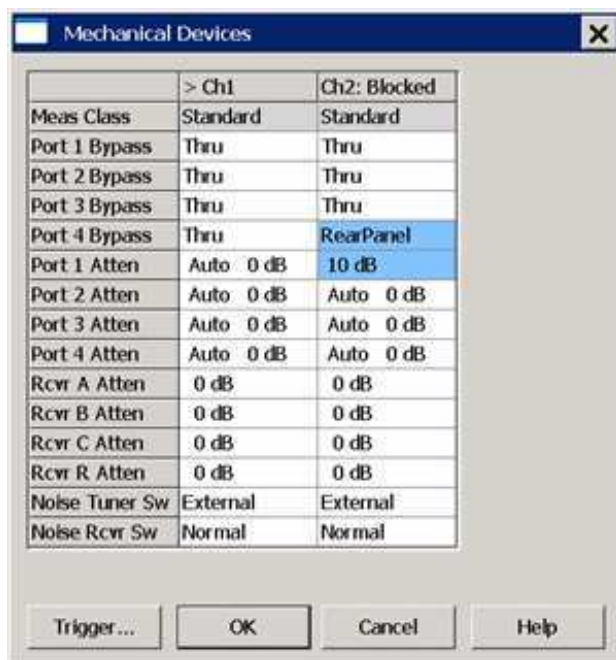
### Using front-panel HARDKEY [softkey] buttons

1. Press TRACE/CHAN
2. then [Channel]
3. then [Hardware Setup]
4. then [Mechanical Devices]

### Using a mouse with PNA Menus

1. Click **Trace/Chan**
2. then **Channel**
3. then **Hardware Setup**
4. then **Mechanical Devices**

No programming commands are available for this feature.





## Mechanical Devices dialog box help

[See Mechanical Devices Overview \(scroll up\)](#)

The devices that appear in the table depend on the PNA model and options.

Blue highlighted cells indicate the following:

- The channel is NOT able to sweep. **Blocked** is shown in the top row.
- The highlighted device settings differ from that of the sweeping channels.

To modify entries in the table, click a cell.

When a selection is changed, the new setting is applied immediately.

If [Port Power is coupled](#), a dialog prompts if coupling should be turned OFF.

### Not supported

- [Measurement Class](#) can NOT be changed from this dialog.
- The dialog does NOT report device settings for [multiport test sets](#).
- This dialog does NOT report device settings for [external sources](#).

**Trigger** Launches the Trigger dialog box.

**OK** Closes the dialog box.

**Cancel** Does not apply changes that were made, and closes the dialog box.

---

Last Modified:

23-Feb-2010    Added Channel Blocking (A.09.20)

31-Jul-2009    MX New topic

## Power Limit and Power Offset

---

- [Overview](#)
- [How to access Power Limit and Power Offset settings](#)

### Other System Topics

## Overview

### Power Limit (Global scope)

Global power limit sets a maximum source power level for individual PNA ports. This value limits port power for all channels and all applications. Power levels that attempt to exceed the power limit is clipped at the limit.

### Notes

- The power limit can NOT be set for power levels which are below the power level that is required by the PNA to achieve phase lock - approximately -30 dBm.
- Because [Fast Sweep mode](#) allows power spiking, it is NOT allowed when a power limit is enabled.
- Components that are added to the RF path are accounted for by entering their loss (negative) or gain (positive) in the [Power Offset](#) section of the dialog box.
- [PNA Applications](#) may change RF path components. For example, IMD for Converters may change the combiner path and add an amplifier for LO input. Compensation is NOT made for these changes and port power may exceed the power limit or port power may be clipped unnecessarily.
- Power limiting does NOT clip power spikes that may occur during [frequency band crossings](#).
- [External test set ports](#) are also included for power limiting.

### Power Offset (Channel scope)

Power Offset provides a method of compensating port power for added attenuation or amplification in the source path. The result is that power at the specified port, all dialogs, and annotation, reflects the added components.

## How to access the Offsets and Limits settings

Also accessed through the [PNA Preferences](#) dialog.

### Using front-panel HARDKEY [softkey] buttons

1. Press **POWER**
2. then **[Power and Attenuators]**
3. then **[Offsets and Limits]**

### Using a mouse with PNA Menus

1. Click **Stimulus**
2. then **Power**
3. then **Power and Attenuators**
4. then **Offsets and Limits**

## Programming Commands



### Offsets and Limits dialog box help

Click a WHITE cell to change values. **Shaded cells** can **NOT** be changed.

[Remote commands](#) can be sent to lock and unlock the dialog box (UI) settings.

#### Power Limit

Limits the source power at each PNA port for ALL channels. Use this feature to protect DUTs that are sensitive to overpowering at the input. Power levels that exceed the limit at the specified port are clipped at the limit and an error message is displayed on the screen.

The Power Limit settings survive [Instrument Preset](#). When an Instrument State is [recalled](#), the current Power Limit settings are applied to the recalled state.

To learn more, see [Power Limit Overview](#) (scroll up).

#### State / Limit

- **ON** - Power is limited to the adjacent value at the specified source port.
- **OFF** - Power is NOT limited to this value, but to the maximum power of the PNA source.

For PNA models with a second internal source, the **Port 1 Src2** Power Limit setting is NEVER available. Make the setting at the standard **Port 1**.

## Power Offset

Power Offset provides a method of compensating port power for added attenuation or amplification in the source path. The result is that power at the specified port, all dialogs, and annotation, reflects the added components.

- For amplification, use positive offset.
- For attenuation, use negative offset.

Then optionally change the Source Power or Port Power values so that the following equation reflects your requirement:

$$\text{Source Power} + \text{Power Offset} = \text{Port Power}$$

**Source Cal ON / OFF** - The power offset value is used as the default Offset in [Source Power Calibration](#). After a Source Power Calibration is performed, the minimum (offset+correction) value from the Source Power Cal is reported in this dialog and is NOT available to be changed.

## Notes

- Power Offset can be used with [Power Sweeps](#). When a power sweep is enabled, the Start and Stop power levels are reported in this dialog.
- When port power offsets are used, port powers are automatically [uncoupled](#). Port powers may not be coupled again until all port offsets are zero.

**OK** Closes the dialog box.

---

Last Modified:

31-Jul-2009    MX New topic

## Setting System Impedance

The system impedance can be changed for measuring devices with an impedance other than 50 ohms, such as waveguide devices. The PNA mathematically transforms and displays the measurement data as though the PNA ports were the specified impedance value. Physically, the test ports are always about 50 ohms.

### How to change the System Impedance

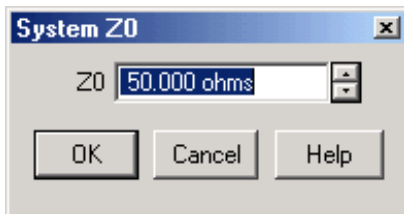
#### Using front-panel HARDKEY [softkey] buttons

1. Press **SYSTEM**
2. then **[Configure]**
3. then **[System Z0]**

#### Using a mouse with PNA Menus

1. Click **Utility**
2. then **System**
3. then **Configure**
4. then **System Z0**

### Programming Commands



#### System Z0 dialog box help

Allows you to change the system impedance (default setting is 50 ohms).

**Z0** Displays the current system impedance.

##### For 75 ohm devices:

1. Change the system Z0 to 75 ohms.
2. Connect minimum loss pads (75 ohm impedance) between the analyzer and the DUT to minimize the physical mismatch.
3. Perform a calibration with 75 ohm calibration standards.

##### For waveguide devices:

1. Change the system Z0 to 1 ohm.
2. Perform a calibration with the appropriate waveguide standards.

Last modified:

3-Sep-2008	Removed legacy content
9/27/06	MX Added UI
9/12/06	Added link to programming commands

## Synchronize PNA with an External (PSG) Source

---

Beginning with PNA Rev. 7.22, the PNA External Source Control feature can be used to automatically control external sources. However, this feature requires certain PNA options. [Learn more.](#)

Many PNA measurements require the use of at least one external source. For example, when measuring the insertion loss of a mixer, the LO must be swept at the same time as the RF input. This requires the PNA and external source to be synchronized.

The following procedure shows how to manually synchronize the PNA with an Agilent PSG Source. Although the settings will be different, the concept is useful with other sources.

### Hardware configuration

- Connect the PNA and PSG Time Base ([PNA 10 MHz OUT](#) to PSG 10 MHz IN)

### PNA-L, E836xB Models

Connect the PSG and [PNA Trigger Connectors](#) as follows:

- PNA Trigger IN to Source OUT
- PNA Trigger OUT to Source IN

### PNA-X Models

Connect either pair (1 or 2) of the [AUX Trigger I/O connectors](#) as follows:

- PNA AUX Trig IN to Source Trigger OUT
- PNA AUX Trig OUT to Source Trigger IN

[Learn more about the AUX Trigger capabilities.](#)

---

### PNA Settings

- [Number of points](#): **Same as PSG**
- [Frequency span](#): Does **NOT** have to be the same as PSG

### [PNA Trigger Settings](#)

- Trigger Source:
  - PNA-L and E836xB models: **External**
  - PNA-X: **Internal, Manual**

- Trigger Scope: **Channel**
- Channel Trigger State: **Same as PSG Sweep Repeat setting**. ([Continuous or Single](#))
- Point Sweep: **Checked**

#### External / Auxiliary Trigger Settings

	PNA-L and E836xB	PNA-X
<b>Input</b>		
Where settings are made:	<a href="#">External Tab</a>	<a href="#">Aux Trigger Tab</a>
Level / Edge:	<b>Edge</b>	<b>Same as PSG (Hi or Low)</b>
Accept Trigger Before Armed:	<b>Checked</b>	N/A
Handshake	N/A	Checked
<b>Output</b>		
Where settings are made:	<a href="#">I/O2 Trig Out Tab</a>	<a href="#">Aux Trigger Tab</a>
Enable	<b>Checked</b>	<b>Checked</b>
Polarity:	<b>Same as PSG</b>	<b>Same as PSG</b>
Position:	<b>After</b>	<b>After</b>
Per Point	<b>Checked</b>	<b>Checked</b>

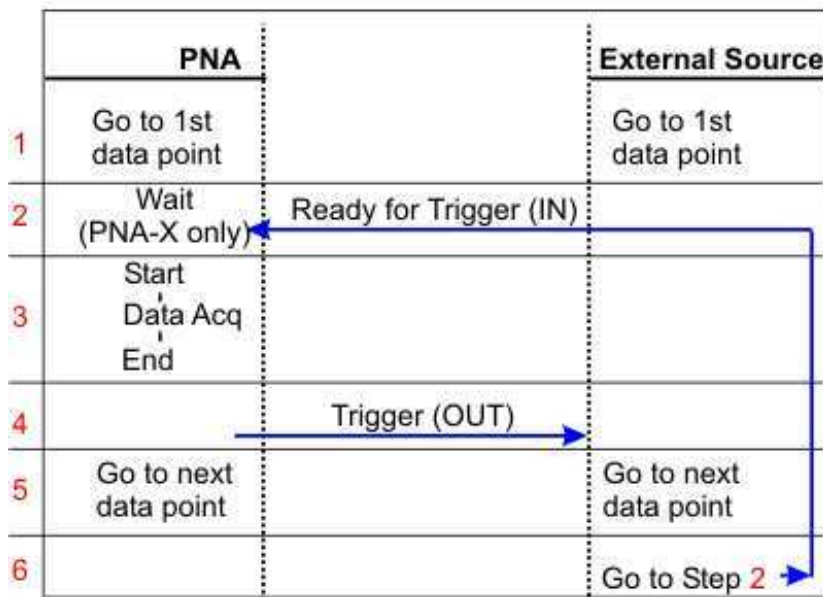
#### PSG Settings

- Number of points: **Same as PNA**
- Sweep: **Step** or **List**
- Sweep Trig: **Free Run**
- Sweep Repeat: **Same as PNA Channel Trigger State** ([Continuous or Single](#))
- Sweep Direction: **UP**
- Point Trig: **Ext**
- Manual Mode: **OFF**
- Trigger In / Out Polarity: **Same as PNA**



## What is Happening?

The following is a flow diagram showing the handshake / synchronization process between the PNA and an External Source.



### Text Description

- 1.** After the measurement setup is complete, both instruments wait on the first data point of a measurement sweep. Both instruments are configured for Continuous or Single sweep.
- 2.** ([see note below](#)) A trigger signal from the source starts the measurement. This is usually accomplished by a key press on the source front panel.
- 3.** PNA data acquisition (measurement) starts, and then stops AFTER the first data point acquisition.
- 4.** The PNA sends a trigger signal out to the source telling it to move to the next frequency data point. This signal can optionally be sent BEFORE data acquisition if required by your application.
- 5.** The external source and PNA move to the next data point. The source usually takes longer than the PNA.
- 6.** The source sends the Ready for Trigger signal to the PNA.
  - **PNA-L and E836xB models** - If the source arrives first, the **Accept Trigger Before Armed** setting is used to accept the trigger signal even if the PNA is not yet ready to start acquisition.
  - **PNA-X using AUX Triggering** - If enabled, the PNA waits indefinitely for a trigger signal from the source. Although AUX triggering does NOT have the **Accept Trigger Before Armed** setting, the Ready for Trigger signal is latched and has the same effect.

**Step 2 Note** PNA-X (Aux TRIG IN) The PNA looks for a level trigger at the start of each sweep, and an edge thereafter. This assumes that the external source ready line will remain in the ready state (high or low) until triggered (step 4) and will then transition to the NOT ready state while moving to the next frequency, and then transition again to the ready state.

## How do you know when the PNA and PSG are in synch?

The measurement results are the ultimate test of whether the source and PNA are synchronized. However, it is possible to see the PSG and PNA sweeping at exactly the same time.

First, lower the PNA [IFBandwidth](#) or increase the [sweep time](#) so that the sweep is slow enough to watch the sweep indicator moving across the PNA screen. At the same time, watch the PSG "progress bar" as it moves through the entire sweep.

If the PNA is stopped in the middle of a sweep, then retriggered, it returns to the first data point. The PSG continues from where it stopped. Therefore, to re-synch the two instruments, the PSG needs to return to the first data point. There are a number of ways to do this. One way is to press the PSG **Manual** button to ON, then OFF. Then trigger a new sweep.

## To trigger a sweep

- **Single** Trigger mode: Both the PNA and PSG Single (trigger) buttons must be pressed (in any order) for each trigger.
- **Continuous** Trigger mode: First, reset the PSG to the first data point, then press the PNA Continuous (trigger) button.

## Maintaining Synchronization

In general, the above setup should start the two instruments sweeping simultaneously. However, any interaction with the PNA could cause the PNA sweep to abort or delay, in which case the two instruments will be out of sync. To avoid this, you can use the PNA [Interface Control](#) feature to send an ABORT to the external device after each sweep.

When the PNA ends a sweep, it sends an ABORT to stop the source. A trigger signal is then sent, either Continuous (automatically) or Single (manual). In either case, both instruments start sweeping in synch.

This takes more time to sweep, but maintains synchronization.

For example, to use this feature with Agilent's PSG source, you would add the following:

On the "After Sweep End" tab, type:

```
24 :ABORT
```

Where 24 is the GPIB address of the source.

---

Last Modified:

11-Feb-2008	Updated note
1-Jun-2007	Added 7.22 update
1-Jan-2007	MX Updated for PNA-X

## PNA Application Notes

---

The following links require an **Internet connection**.

**Note:** Check out the multimedia PNA Demo presentations, including '[Network Analyzer Basics](#)'.

### Calibrations

[Improving Measurement and Calibration Accuracy Using the Frequency Converter Application \(5988-9642EN\)](#)

[On-Wafer Calibration Using a 4-port, 20 GHz PNA-L Network Analyzer \(N5230A Option 240/245\) \(5989-2287EN\)](#)

**New** [Specifying Calibration Standards and Kits for Agilent Vector Network Analyzers \(5989-4840EN\)](#)

### ECal

[Agilent Electronic vs. Mechanical Calibration Kits: Calibration Methods and Accuracy \(5988-9477EN\)](#)

[User Characterization: Electronic Calibration Feature Allows Users to Customize to Specific Needs \(5988-9478EN\)](#)

### Embedding / De-embedding

[De-embedding and Embedding S-Parameter Networks Using a Vector Network Analyzer \(5980-2784EN\)](#)

### Amplifier Measurements

[High-power measurements using the PNA \(5989-1349EN\)](#)

[Amplifier Linear and Gain Measurements \(5988-8644EN\)](#)

[Amplifier Swept-Harmonic Measurements \(5988-9473EN\)](#)

[Amplifier and CW Swept Intermodulation-Distortion Measurements \(5988-9474EN\)](#)

**New** [Making Accurate IMD Measurements with the PNA-X Network Analyzer \(5989-7265EN\)](#)

### Antenna Measurements

[Triggering PNA Microwave Network Analyzers for Antenna Measurements \(5988-9518EN\)](#)

[New Network Analyzer Methodologies in Antenna/RCS Measurements \(5989-1937EN\)](#)

[Pulsed Antenna Measurements Using PNA Network Analyzers \(5989-0221EN\)](#)

**Balanced Measurements** (Although the following refer to the ENA, they are also relevant to the PNA.)

[On-wafer Balanced Component Measurement with the Cascade Microtech Probing System \(5988-5886EN\)](#)

[Network De-embedding/Embedding and Balanced Measurement \(5988-4923EN\)](#)

### Mixer Measurements

[Mixer Transmission Measurements Using the Frequency Conversion Application \(5988-8642EN\)](#)

[Mixer Conversion-Loss and Group Delay Measurement Techniques and Comparisons \(5988-9619EN\)](#)

[Comparison of Mixer Characterization using New Vector Characterization Techniques \(5988-7827EN\)](#)

[Novel Method for Vector Mixer Characterization and Mixer Test System Vector Error Correction \(5988-7826EN\)](#)

[Measuring Absolute Group Delay of Multistage Converters Using PNA Microwave Network Analyzers \(5989-0219EN\)](#)

**New** [Measuring Group Delay of Frequency Converters with Embedded Local Oscillators \(5989-7385EN\)](#)

## **Pulsed Measurements**

[Pulsed-RF S-Parameter Measurements Using Wideband and Narrowband Detection \(AN 1408-12\)](#)

[Accurate Pulsed Measurements \(5989-0563EN\)](#)

[Pulsed Antenna Measurements Using PNA Network Analyzers \(5989-0221EN\)](#)

## **Other Measurements**

**New** [Time Domain Analysis](#)

[Using the PNA Series to Analyze Lightwave Components \(5989-3385EN\)](#)

[Using the PNA for Banded Millimeter-Wave Measurements \(5989-4098EN\)](#)

[PNA MM-Wave Network Analyzers: Analysis of Cable Length on VNA System Performance \(5989-1941EN\)](#)

[Basics of Measuring the Dielectric Properties of Materials \(5989-2589EN\)](#)

## **Automation**

[Connectivity Advances for Component Manufacturers \(5980-2782EN\)](#)

[Introduction to Application Development using the PNA \(5980-2666EN\)](#)

[The 'Need for Speed' in Component Manufacturing Test \(5980-2783EN\)](#)

---

Last Modified:

14-Apr-2008     Added IMD with PNA-X.

## Network Analyzer Basics

---

This self-paced two hour video discusses the basic concepts of Network Analysis.

The files are installed and should work on older PNA models. If the PNA link does not work, then use the internet link, which requires an internet connection.

- **From the PNA:** [Proceed with Network Analyzer Basics](#).
  - **From the Internet:** <http://wireless.agilent.com/networkanalyzers/pnademmo.htm> in both streaming and downloadable format.
- 

Last modified:

10/18/06    Added link to pnademmo.

## Connector Care

---

Proper connector care is critical for accurate and repeatable measurements. The following information will help you preserve the precision and extend the life of your connectors - saving both time and money.

- [Connector Care Quick Reference Guide](#)
- [Connector Cleaning Supplies](#)
- [Safety Reminders](#)
- [About Connectors](#)
- [Gaging Fundamentals](#)
- [Connector Care Procedures](#)

See also mmWave Connector Care at [http://na.tm.agilent.com/pna/connectorcare/Connector\\_Care.htm](http://na.tm.agilent.com/pna/connectorcare/Connector_Care.htm)

### Preventing Test Port Connector Damage

#### Handling and Storing Connectors

##### Do

Keep connectors clean  
Protect connectors with plastic end caps  
Keep connector temperature same as analyzer

##### Do Not

Touch mating-plane surfaces  
Set connectors contact-end down  
Store connectors loose in box or drawer

#### Visual Inspection

##### Do

Inspect connectors with magnifying glass.  
Look for metal debris, deep scratches or dents

##### Do Not

Use a connector with a bent or broken center conductor  
Use a connector with deformed threads

#### Cleaning Connectors

##### Do

Clean surfaces first with clean, dry compressed air  
Use lint-free swab or brush  
Use minimum amount of alcohol  
Clean outer conductor mating surface and threads

##### Do Not

Use high pressure air (>60 psi)  
Use any abrasives  
Allow alcohol into connector support beads  
Apply lateral force to center conductor

### Gaging Connectors

#### Do

Inspect and clean gage, gage master and device tested  
Use correct torque wrench  
zero gage before use  
Use multiple measurements and keep record of readings

#### Do Not

Use an out of specification connector  
Hold connector gage by the dial

### Making Connections

#### Do

Align connectors first  
Rotate only the connector nut  
Use correct torque wrench

#### Do Not

Cross thread the connection  
Twist connector body to make connection  
Mate different connector types

### Connector Care and Cleaning Supplies

Description	Web Site
Swabs	<a href="http://www.berkshire.com/swabs.shtml">http://www.berkshire.com/swabs.shtml</a>
Lint Free Cloths- Air dusters	<a href="http://www.ccrwebstore.com">http://www.ccrwebstore.com</a>
Isopropyl	<a href="http://www.techspray.com">http://www.techspray.com</a>
Nitrilite Gloves and Finger Cots	<a href="http://www.techni-tool.com">http://www.techni-tool.com</a>

### Safety Reminders

#### When cleaning connectors:

- Always use protective eyewear when using compressed air or nitrogen.
- Keep isopropyl alcohol away from heat, sparks and flame. Use with adequate ventilation. Avoid contact with eyes, skin and clothing.
- Avoid electrostatic discharge (ESD). Wear a grounded wrist strap (having a 1 M series resistor) when cleaning device, cable or test port connectors.
- Cleaning connectors with alcohol shall only be done with the instruments power cord removed, and in a well-ventilated area. Allow all residual alcohol moisture to evaporate, and the fumes to dissipate prior to energizing the instrument.

### About Connectors

- [Connector Service Life](#)
- [Connector Grades and Performance](#)
- [Adapters as Connector Savers](#)
- [Connector Mating Plane Surfaces](#)

## Connector Service Life

Even though calibration standards, cables, and test set connectors are designed and manufactured to the highest standards, all connectors have a limited service life. This means that connectors can become defective due to wear during normal use. For best results, all connectors should be inspected and maintained to maximize their service life.

**Visual Inspection** should be performed each time a connection is made. Metal particles from connector threads often find their way onto the mating surface when a connection is made or disconnected. See [Inspection](#) procedure.

**Cleaning** the dirt and contamination from the connector mating plane surfaces and threads can extend the service life of the connector and improve the quality of your calibration and measurements. See [Cleaning](#) procedure.

**Gaging** connectors not only provides assurance of proper mechanical tolerances, and thus connector performance, but also indicate situations where the potential for damage to another connector may exist. See [Gaging](#) procedure.

**Proper connector care and connection techniques yield:**

Longer Service Life

Higher Performance

Better Repeatability

## Connector Grades and Performance

The three connector grades (levels of quality) for the popular connector families are listed below. Some specialized types may not have all three grades.

- **Production** grade connectors are the lowest grade and the least expensive. It is the connector grade most commonly used on the typical device under test (DUT). It has the lowest performance of all connectors due to its loose tolerances. This means that production grade connectors should always be carefully inspected before making a connection to the analyzer. Some production grade connectors are not intended to mate with metrology grade connectors.
- **Instrument** grade is the middle grade of connectors. It is mainly used in and with test instruments, most cables and adapters, and some calibration standards. It provides long life with good performance and tighter tolerances. It may have a dielectric supported interface and therefore may not exhibit the excellent match of a metrology grade connector.
- **Metrology** grade connectors have the highest performance and the highest cost of all connector grades. This grade is used on calibration standards, verification standards, and precision adapters. Because it is a high precision connector, it can withstand many connections and disconnections and, thus, has the longest life of all connector grades. This connector grade has the closest material and geometric specifications. Pin diameter and pin depth are very closely specified. Metrology grade uses an air dielectric interface and a slotless female contact which provide the highest performance and traceability.



**Note:** In general, Metrology grade connectors should not be mated with Production grade connectors.

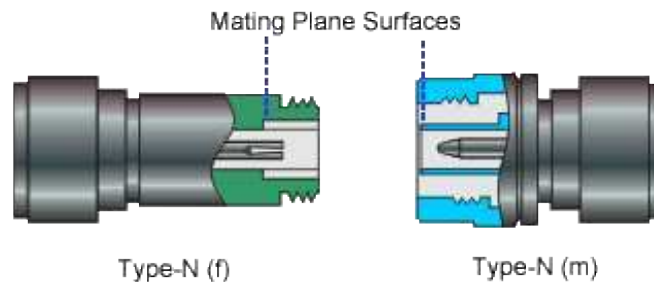
### Adapters as Connector Savers

Make sure to use a high quality (Instrument grade or better) adapter when adapting a different connector type to the analyzer test ports. It is a good idea to use an adapter even when the device under test is the same connector type as the analyzer test ports. In both cases, it will help extend service life, and protect the test ports from damage and costly repair.

The adapter must be fully inspected before connecting it to the analyzer test port and inspected and cleaned frequently thereafter. Because calibration standards are connected to the adapter, the adapter should be the highest quality to provide acceptable RF performance and minimize the effects of mismatch.

### Connector Mating Plane Surfaces

An important concept in RF and microwave measurements is the reference plane. For a network analyzer, this is the surface that all measurements are referenced to. At calibration, the reference plane is defined as the plane where the mating plane surfaces of the measurement port and the calibration standards meet. Good connections (and calibrations) depend on perfectly flat contact between connectors at all points on the mating plane surfaces (as shown in the following graphic).

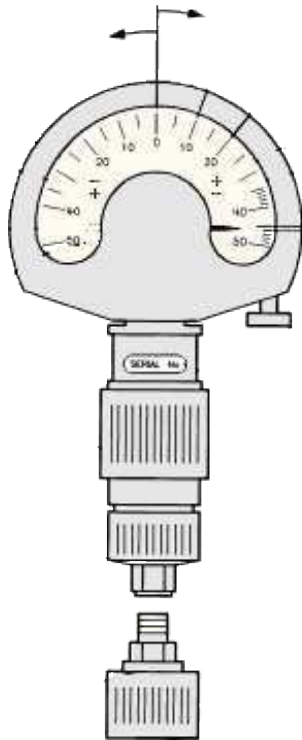


### Gaging Fundamentals

Connector gages are important tools used to measure center conductor pin depth in connectors. Connector pin depth, measured in terms of recession or protrusion, is generally the distance between the mating plane and the end of the center conductor, or the shoulder of the center conductor for a stepped male pin.

### Typical Connector Gage

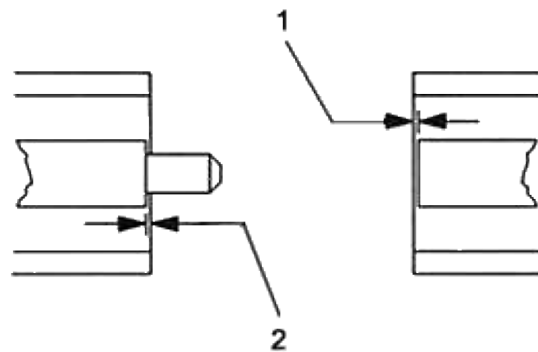
RECESSION    PROTRUSION



### Recession and Protrusion

Pin depth is negative (recession) if the center conductor is recessed below the outer conductor mating plane, usually referred to as the "reference plane". Pin depth is positive (protrusion) if the center conductor projects forward from the connector reference plane.

### Pin Depth



1. Recession of female contact
2. Recession of male pin shoulder

### Difference with Type-N Connectors

Type-N connectors have the mating plane of the center conductors offset from the connector reference plane. In this case the zero setting "gage masters" generally offset the nominal distance between the center conductor mating plane and the connector reference plane.

### When to Gage Connectors

- Before using a connector or adapter the first time.
- When visual inspection or electrical performance suggests the connector interface may be out of range.
- After every 100 connections, depending on use.

### Connector Gage Accuracy

Connector gages (those included with calibration and verification kits), are capable of performing coarse measurements only. This is due to the repeatability uncertainties associated with the measurement. It is important to recognize that test port connectors and calibration standards have mechanical specifications that are extremely precise. Only special gaging processes and electrical testing (performed in a calibration lab) can accurately verify the mechanical characteristics of these devices. The pin depth specifications in the Agilent calibration kit manuals provide a compromise between the pin depth accuracy required, and the accuracy of the gages. The gages shipped with calibration and verification kits allow you to measure connector pin depth and avoid damage from out-of-specification connectors.

**Note:** Before gaging any connector, the mechanical specifications provided with that connector or device should be checked.

### To Gage Connectors

1. Wear a grounded wrist strap (having a 1 M $\Omega$  series resistor).
2. Select proper gage for device under test (DUT).
3. Inspect and clean gage, gage master, and DUT.
4. Zero the connector gage.
  - a. While holding gage by the barrel, carefully connect gage master to gage. Finger-tighten connector nut only.
  - b. Use proper torque wrench to make final connection. If needed, use additional wrench to prevent gage master (body) from turning. Gently tap the barrel to settle the gage.
  - c. The gage pointer should line up exactly with the zero mark on gage. If not, adjust "zero set" knob until gage pointer reads zero. On gages having a dial lock screw and a movable dial, loosen the dial lock screw and move the dial until the gage pointer reads zero. Gages should be zeroed before each set of measurements to make sure zero setting has not changed.
  - d. Remove gage master.
5. Gage the device under test.
  - a. While holding gage by the barrel, carefully connect DUT to gage. Finger-tighten connector nut only.
  - b. Use proper torque wrench to make final connection and, if needed, use additional wrench to prevent DUT (body) from turning. Gently tap the barrel to settle the gage.

- c. Read gage indicator dial for recession or protrusion and compare reading with device specifications.

**Caution:** If the gage indicates excessive protrusion or recession, the connector should be marked for disposal or sent out for repair.

6. For maximum accuracy, measure the device a minimum of three times and take an average of the readings. After each measurement, rotate the gage a quarter-turn to reduce measurement variations.
7. If there is doubt about measurement accuracy, be sure the temperatures of the parts have stabilized. Then perform the cleaning, zeroing, and measuring procedure again.

## Connector Care Procedures

- [Inspecting Connectors](#)
- [Cleaning Connectors](#)
- [Making Connections](#)
- [Using a Torque Wrench](#)
- [Handling and Storing Connectors](#)

### To Inspect Connectors

Wear a grounded wrist strap (having a 1 M  $\Omega$  series resistor).

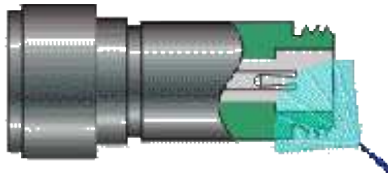
Use a magnifying glass ( $\geq 10\times$ ) and inspect connector for the following:

- Badly worn plating or deep scratches
- Deformed threads
- Metal particles on threads and mating plane surfaces
- Bent, broken, or mis-aligned center conductors
- Poor connector nut rotation

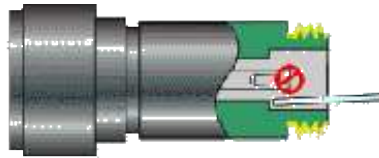
**Caution:** A damaged or out-of-specification device can destroy a good connector attached to it even on the first connection. Any connector with an obvious defect should be marked for disposal or sent out for repair.

### To Clean Connectors

1. Wear a grounded wrist strap (having a 1 M  $\Omega$  series resistor).
2. Use clean, low-pressure air to remove loose particles from mating plane surfaces and threads. Inspect connector thoroughly. If additional cleaning is required, continue with the following steps.



3. Moisten—do not saturate—a lint-free swab with isopropyl alcohol. See [Cleaning Supplies](#) for recommended type.
4. Clean contamination and debris from mating plane surfaces and threads. When cleaning interior surfaces, avoid exerting pressure on center conductor and keep swab fibers from getting trapped in the female center conductor.



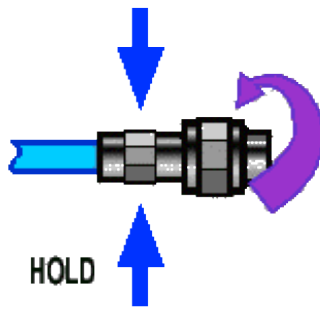
5. Let alcohol evaporate—then use compressed air to blow surfaces clean.
6. Inspect connector. Make sure no particles or residue remains.
7. If defects are still visible after cleaning, the connector itself may be damaged and should not be used. Determine the cause of damage before making further connections.

### To Make Connections

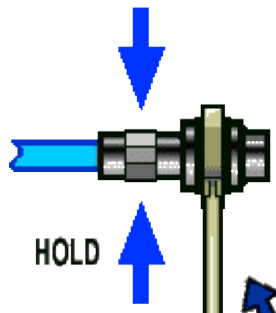
1. Wear a grounded wrist strap (having a 1 M series resistor).
2. Inspect, clean, and gage connectors. All connectors must be undamaged, clean, and within mechanical specification.
3. Carefully align center axis of both devices. The center conductor pin—from the male connector—must slip concentrically into the contact finger of the female connector.



4. Carefully push the connectors straight together so they can engage smoothly. Rotate the connector nut (not the device itself) until finger-tight, being careful not to cross the threads.



5. Use a torque wrench to make final connection. Tighten until the "break" point of the torque wrench is reached. Do **not** push beyond initial break point. Use additional wrench, if needed, to prevent device body from turning.

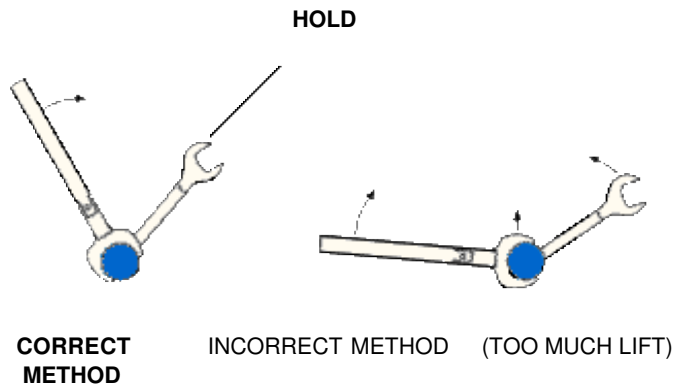


#### To Separate a Connection

1. Support the devices to avoid any twisting, rocking or bending force on either connector.
2. Use an open-end wrench to prevent the device body from turning.
3. Use another open-end wrench to loosen the connector nut.
4. Complete the disconnection by hand, turning only the connector nut.
5. Pull the connectors straight apart.

#### To Use a Torque Wrench

1. Make sure torque wrench is set to the correct torque setting.
2. Position torque wrench and a second wrench (to hold device or cable) within 90° of each other before applying force. Make sure to support the devices to avoid putting stress on the connectors.



3. Hold torque wrench lightly at the end of handle—then apply force perpendicular to the torque wrench handle. Tighten until the "break" point of the torque wrench is reached. Do **not** push beyond initial break point.

#### TORQUING DIRECTION



**STOP WHEN HANDLE BEGINS TO YIELD**

### To Handle and Store Connectors

- Install protective end caps when connectors are not in use.
- Never store connectors, airlines, or calibration standards loose in a box. This is a common cause of connector damage.
- Keep connector temperature the same as analyzer. Holding the connector in your hand or cleaning connector with compressed air can significantly change the temperature. Wait for connector temperature to stabilize before using in calibration or measurements.
- Do not touch mating plane surfaces. Natural skin oils and microscopic particles of dirt are difficult to remove from these surfaces.
- Do not set connectors contact-end down on a hard surface. The plating and mating plane surfaces can be damaged if the interface comes in contact with any hard surface.
- Wear a grounded wrist strap and work on a grounded, conductive table mat. This helps protect the analyzer and devices from electrostatic discharge (ESD).





## Electrostatic Discharge (ESD) Protection

---

Protection against electrostatic discharge (ESD) is essential while removing or connecting cables to the network analyzer. Static electricity can build up on your body and can easily damage sensitive internal circuit elements when discharged. Static discharges too small to be felt can cause permanent damage. To prevent damage to the instrument:

- **Always** have a grounded, conductive table mat in front of your test equipment.
- **Always** wear a grounded wrist strap, connected to a grounded conductive table mat, having a 1 M $\Omega$  resistor in series with it, when making test setup connections.
- **Always** wear a heel strap when working in an area with a conductive floor. If you are uncertain about the conductivity of your floor, wear a heel strap.
- **Always** ground yourself before you clean, inspect, or make a connection to a static-sensitive device or test port. You can, for example, grasp the grounded outer shell of the test port or cable connector briefly.
- **Always** ground the center conductor of a test cable before making a connection to the analyzer test port or other static-sensitive device. This can be done as follows:
  1. Connect a short (from your calibration kit) to one end of the cable to short the center conductor to the outer conductor.
  2. While wearing a grounded wrist strap, grasp the outer shell of the cable connector.
  3. Connect the other end of the cable to the test port and remove the short from the cable.

See [Analyzer Accessories](#) for ESD part numbers.

## Absolute Output Power

---

An absolute output-power measurement displays absolute power versus frequency.

- [What is Absolute Output Power?](#)
- [Why Measure Absolute Output Power?](#)
- [Accuracy Considerations](#)
- [How to Measure Absolute Output Power](#)

[See other Amplifier Parameters topics](#)

### What is Absolute Output Power?

An absolute-output power measurement displays the power present at the analyzer's input port. This power is absolute-it is not referenced (ratioed) to the incident or source power. In the log mag format, values associated with the grid's vertical axis are in units of dBm, which is the power measured in reference to 1 mW.

- 0 dBm = 1 mW
- -10 dBm = 100  $\mu$ W
- +10 dBm = 10 mW

In the linear mag format, values associated with the grid's vertical axis are in units of watts (W).

### Why Measure Absolute Output Power?

Absolute output power is measured when the amplifier's output must be quantified as absolute power rather than a ratioed relative power measurement. For example, during a gain compression measurement, it is typical to also measure absolute output power. This shows the absolute power out of the amplifier where 1-dB compression occurs.

### Accuracy Considerations

The output power of the amplifier should be sufficiently attenuated if necessary. Too much output power could:

- Damage the analyzer receiver
- Exceed the input compression level of the analyzer receiver, resulting in inaccurate measurements.

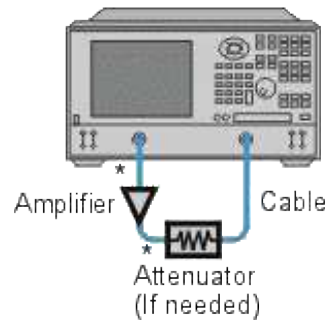
Attenuation of the amplifier's output power can be accomplished using either attenuators or couplers

The amplifier may respond very differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.

### How to Measure Absolute Power

Do the following to measure absolute output power:

1. Preset the analyzer.
2. Select an unratioed power measurement (receiver B).
3. Set the analyzer's source power to 0 dBm.
4. Select an external attenuator (if needed) so the amplifier's output power will be sufficiently attenuated to avoid causing receiver compression or damage to the analyzer's port-2.
5. Connect the amplifier as shown in the following graphic, and provide the dc bias.



\* Direct Connection

6. Select the analyzer settings for your amplifier under test.
7. Remove the amplifier and connect the measurement ports together. Store the data to memory. Be sure to include the attenuator and cables in the test setup if they will be used when measuring the amplifier.
8. Save the instrument state to memory.
9. Reconnect the amplifier.
10. Select the data math function Data/Memory.
11. Scale the displayed measurement for optimum viewing and use a marker to measure the absolute output-power at a desired frequency.
12. Print or save the data to a disk.

## AM-PM Conversion

---

The AM-PM conversion of an amplifier is a measure of the amount of undesired **phase** deviation (PM) that is caused by amplitude variations (AM) inherent in the system.

- [What Is AM-PM Conversion?](#)
- [Why Measure AM-PM Conversion](#)
- [Accuracy Considerations](#)
- [How to Measure AM-PM Conversion](#)

### [Other Tutorials topics](#)

## What Is AM-PM Conversion?

AM-to-PM conversion measures the amount of undesired phase deviation (PM) that is caused by amplitude variations (AM) of the system. For example, unwanted phase deviation (PM) in a communications system can be caused by:

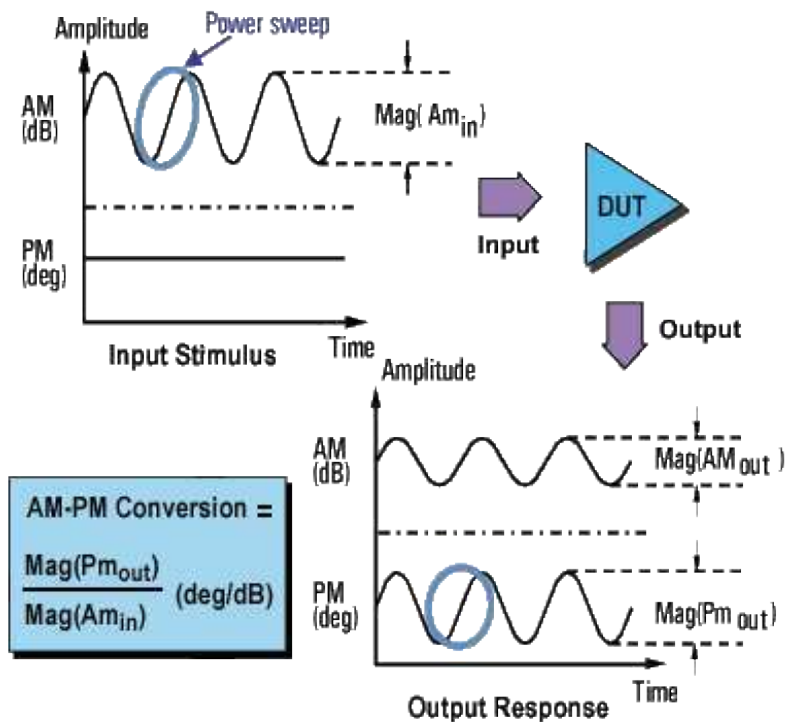
### Unintentional amplitude variations (AM)

- Power supply ripple
- Thermal drift
- Multipath fading

### Intentional modulation of signal amplitude

- QAM
- Burst modulation

AM-to-PM conversion is usually defined as the change in output phase for a 1-dB increment in the power-sweep applied to the amplifier's input (i.e. at the 1 dB gain compression point). It is expressed in degrees-per-dB (°/dB). An ideal amplifier would have no interaction between its phase response and the power level of the input signal.



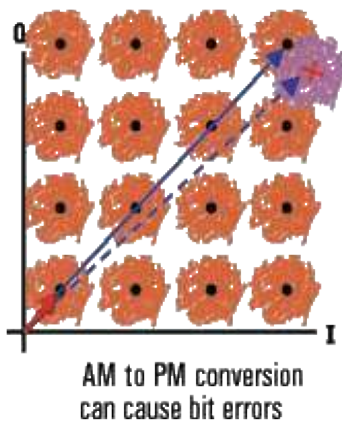
## Why Measure AM-PM Conversion

AM-to-PM conversion is a critical parameter in systems where phase (angular) modulation is used, such as:

- FM
- QPSK
- 16QAM

It is a critical parameter because undesired phase deviation (PM) causes analog signal degradation, or increased bit-error rates (BER) in digital communication systems. While it is easy to measure the BER of a digital communication system, this measurement alone does not help you understand the underlying causes of bit errors. AM-to-PM conversion is one of the fundamental contributors to BER, and therefore it is important to quantify this parameter in communication systems.

Refer to the I/Q diagram below for the following discussion on how AM-to-PM conversion can cause bit errors.



- The desirable state change is from the small solid vector to the large solid vector.
- With AM-to-PM conversion, the large vector may actually end up as shown with the dotted line. This is due to phase shift that results from a change in the input power level.
- For a 64QAM signal as shown (only one quadrant is drawn), we see that the noise circles that surround each state would actually overlap, which means that statistically, some bit errors would occur.

## Accuracy Considerations

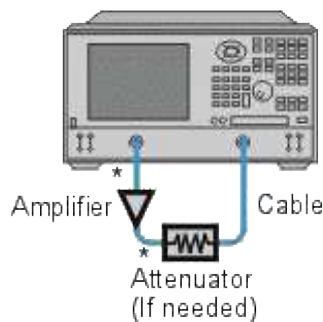
With this method of measuring AM-to-PM conversion, the modulation frequency is approximately the inverse of the sweep time. Even with the fastest power sweep available on most network analyzers, the modulation frequency ends up being fairly low (typically less than 10 Hz). This could cause a slight temperature change as the sweep progresses, especially if the amplifier has low thermal mass, typical of an unpackaged device. Results using this method could differ slightly if the nonlinear behavior of an amplifier is extremely sensitive to thermal changes. (The PNA series analyzers can make power sweeps <1 ms.)

- The amplifier may respond very differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.
- The output power of the amplifier should be sufficiently attenuated if necessary. Too much output power could:
  - damage the analyzer receiver
  - exceed the input compression level of the analyzer receiver, resulting in inaccurate measurements
- Attenuation of the amplifier's output power can be accomplished using:
  - Attenuators
  - Couplers
- The frequency-response effects of the attenuators and couplers must be accounted for during calibration since they are part of the test system. Proper error-correction techniques can reduce these effects.
- The frequency response is the dominant error in an AM-to-PM conversion measurement setup. Performing a

thru-response measurement calibration significantly reduces this error. For greater accuracy, perform a 2-port measurement calibration.

## How to Measure AM-PM Conversion

1. Preset the analyzer.
2. Select an S21 measurement in the power-sweep mode.
3. Enter the start and stop power levels for the analyzer's power sweep. The start power level should be in the linear region of the amplifier's response (typically 10-dB below the 1-dB compression point). The stop power should be in the compression region of the amplifier's response.
4. Select an external attenuator (if needed) so the amplifier's output power will be sufficiently attenuated to avoid causing receiver compression or damage to the analyzer's port 2.
5. Connect the amplifier as shown in the following graphic, and provide the dc bias.



\* Direct Connection

6. Select the analyzer settings for your amplifier under test in order to perform a swept-power gain compression measurement at a chosen frequency. See [Gain Compression](#).
7. Remove the amplifier and perform a measurement calibration. Be sure to include the attenuator and cables in the calibration setup if they will be used when measuring the amplifier.
8. Save the instrument state to memory.
9. Reconnect the amplifier.
10. Use a reference marker to target the amplifier's input power at the 1-dB gain compression point. Select a second marker and adjust its stimulus value until its response is 1-dB below the reference marker.
11. Change the S21 measurement from a log magnitude format to a [phase](#) format (no new calibration is required).
12. Find the [phase](#) change between the markers. The value is the AM-to-PM conversion coefficient at the 1-dB gain compression point.
13. Print the data or save it to a disk.





## Amplifier Parameters Reference

---

- [Gain](#)
- [Gain Flatness](#)
- [Reverse Isolation](#)
- [Gain Drift Versus Time](#)
- [Deviation from Linear Phase](#)
- [Group Delay](#)
- [Return Loss \(SWR,  \$\Gamma\$ \)](#)
- [Complex Impedance](#)
- [Gain Compression](#)
- [AM-to-PM Conversion](#)

### See Also

- [High-Gain Amplifiers](#)
- [High Power with PNA-X](#)

### Gain

$$\tau = \frac{V_{\text{trans}}}{V_{\text{inc}}}$$
$$\text{Gain (dB)} = -20 \log_{10} |\tau|$$
$$\text{Gain (dB)} = P_{\text{out}} \text{ (dBm)} - P_{\text{in}} \text{ (dBm)}$$

The ratio of the amplifier's output power (delivered to a  $Z_0$  [load](#)) to the input power (delivered from a  $Z_0$  source).  $Z_0$  is the [characteristic impedance](#), in this case, 50  $\Omega$ .

For small signal levels, the output power of the amplifier is proportional to the input power. Small signal gain is the gain in this linear region.

As the input power level increases and the amplifier approaches saturation, the output power reaches a limit and the gain drops. Large signal gain is the gain in this nonlinear region. See [Gain Compression](#).

### Gain Flatness

The variation of the gain over the frequency range of the amplifier. See [Small Signal Gain and Flatness](#).

## Reverse Isolation

The measure of transmission from output to input. Similar to the gain measurement except the signal stimulus is applied to the output of the amplifier. See [Reverse Isolation](#).

## Gain Drift versus Time (temperature, bias)

The maximum variation of gain as a function of time, with all other parameters held constant. Gain drift is also observed with respect to other parameter changes such as temperature, humidity or bias voltage.

## Deviation from Linear Phase

The amount of variation from a linear **phase** shift. Ideally, the phase shift through an amplifier is a linear function of frequency. See [Deviation from Linear Phase](#).

## Group Delay

$$\begin{aligned}\tau_g (\text{sec}) &= - \frac{\Delta \theta}{\Delta \omega} \\ &= - \frac{1}{360} * \frac{\Delta \theta}{\Delta f}\end{aligned}$$

The measure of the transit time through the amplifier as a function of frequency. A perfectly linear phase shift would have a constant rate of change with respect to frequency, yielding a constant group delay. See [Group Delay](#).

## Return Loss (SWR, )

$$\begin{aligned}\Gamma &= \frac{V_{\text{refl}}}{V_{\text{inc}}} = \rho \angle \theta \\ \text{Reflection coefficient} &= \rho \\ \text{Return loss (dB)} &= -20 \log_{10} \rho \\ \text{SWR} &= \frac{1+\rho}{1-\rho}\end{aligned}$$

The measure of the reflection mismatch at the input or output of the amplifier relative to the system  $Z_0$  [characteristic impedance](#).

## Complex Impedance

$$\begin{aligned}Z &= \frac{1+\Gamma}{1-\Gamma} * Z_0 \\ &= -R + jX\end{aligned}$$

Complex [impedance](#) (1+G). The amount of reflected energy from an amplifier is directly related to its impedance. Complex impedance consists of both a resistive and a reactive component. It is derived from the characteristic impedance of the system and the reflection coefficient. See [Complex Impedance](#).

## Gain Compression

See [Gain Compression](#) Application.

### AM-to-PM Conversion Coefficient

$$\text{AM/PM} = \frac{\Delta \theta}{\Delta P}$$

The amount of [phase](#) change generated in the output signal of an amplifier as a result of an amplitude change of the input signal.

The AM-to-PM conversion coefficient is expressed in units of degrees/dB at a given power level (usually P<sub>1dB</sub>, which is the 1 dB gain compression point). See [AM-PM Conversion](#).

## Antenna Measurements

---

This topic describes how to setup the PNA to make S21 measurements on an array of antennas. Measurements can be made on up to 100 antenna arrays (Ports) and up to 15 discrete frequencies

### Measurement Sequence

1. The PNA is set to a start frequency.
2. As the antenna moves, the PNA responds to each external trigger signal by measuring an antenna port.
3. When all ports are measured, the PNA increments to the next frequency
4. Again the PNA measures all ports, and so forth until all ports are measured at all frequencies in the forward direction.
5. As the antenna begins moving in the opposite direction, the same sequence occurs, except the PNA decrements in frequency until all ports are measured at all frequencies and the PNA is set back to the original start frequency.

Once setup, only external trigger signals are sent to the PNA. After each trigger, measurement data is stored in internal PNA memory.

### How to set up the PNA

1. On the **System** menu click **Preset**
2. On the **Sweep** menu point to **Trigger** then click **Trigger**
3. In Trigger Source click **External**
4. In Trigger Scope click **Channel**
5. Click **OK**

### Forward Sweep

1. On the **Trace** menu click **New Trace**
2. Click **S21** then Channel Number **1**
3. On the **Sweep** menu point to **Trigger** then click **Trigger**
4. In Channel Trigger State check **Point Sweep**
5. Click **OK**
6. On the Sweep menu click **Sweep Type:** then **Segment Sweep**
7. Click **OK**

8. On the **View** menu point to **Tables** then click **Segment Table**
9. Do this 15 times - Sweep menu point to **Segment Table** then **Insert Segment**
10. For each Segment in the Segment table:
  1. Click **State**:and select **ON**
  2. Double click both **START** and **STOP** Frequency: (each new segment ascends in frequency)
  3. Double click **Points**: type Number of Ports (elements)

### **Reverse sweep**

Repeat the following steps for each frequency: (up to 15)

- Increment the channel number (**X**) Starting with Channel 2
  - Decrement the frequency (**F**)
1. On the **Trace** menu click **New Trace...**
  2. Click **S21** then Channel Number **X**
  3. When a window contains four traces, check **Create in New Window**.
  4. Click **OK**
  5. On the **Sweep** menu point to **Trigger** then click **Trigger**
  6. In Channel Trigger State check **Point Sweep**
  7. Click **OK**
  8. On the Sweep menu click **Sweep Type**:then **Segment Sweep**
  9. Click **OK**
  10. On the **View** menu point to **Tables** then click **Segment Table**
  11. In the Segment table
    1. Click **State**:and select **ON**
    2. Double click both **START** and **STOP** Frequency **F**
    3. Double click **Points**: type Number of Ports (elements)

---

Last Modified:

8-Apr-2008    Removed reference to antenna macro



## Balanced Measurements

---

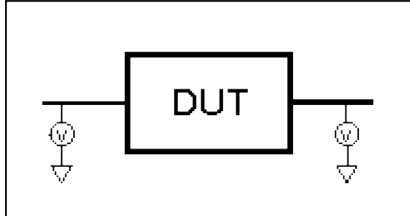
- [What are Balanced Devices?](#)
- [Differential and Common Modes Model](#)
- [Measuring Mixed Mode \(Balanced\) S-Parameters](#)
- [Measuring Imbalance Parameters](#)
- [Measuring CMRR](#)
- [Port Mapping](#)
- [How the PNA makes Balanced Measurements](#)

### Other [Measurement Setup](#) Topics

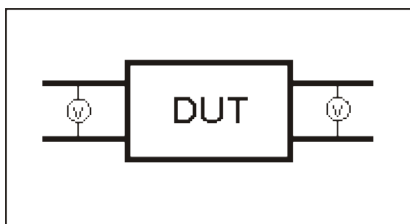
Check out the [Integrated True Mode Stimulus Application \(ITMSA\)](#).

### What are Balanced Devices?

Standard **Single-ended devices** generally have one input port and one output port. Signals on the input and output ports are referenced to ground.



**Balanced devices** have two pins on either the input, the output, or both. The signal of interest is the difference and average of the two input or output lines, not referenced to ground.



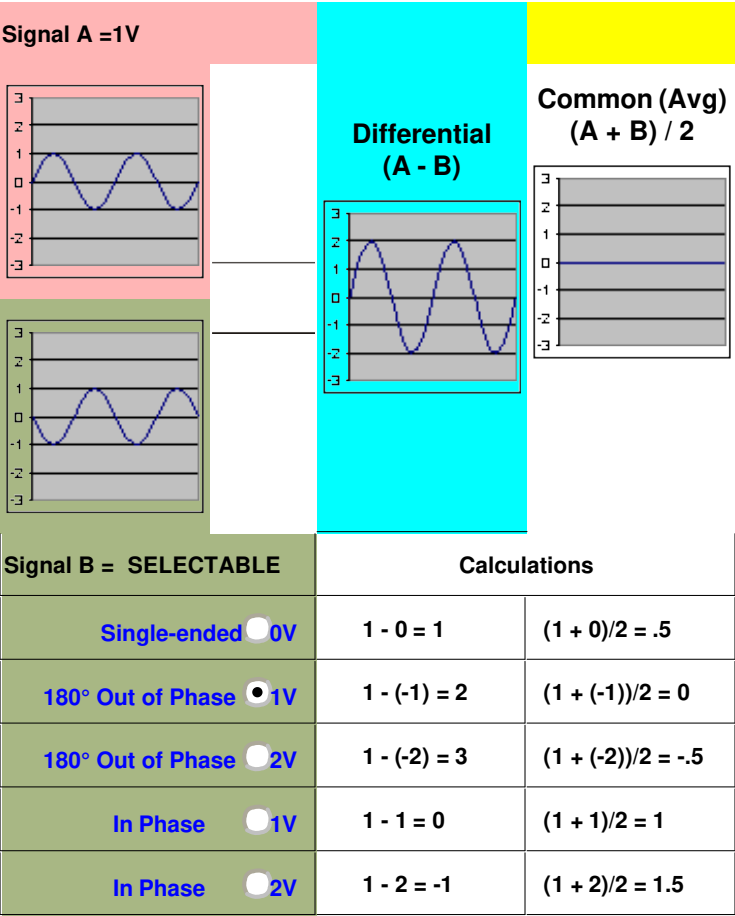
### Differential and Common Modes Model

On balanced devices, the signal of interest is the **difference** and **average** of the two input or output lines. In balanced device terminology, these signals are known as the Differential and Common modes.

The following model shows how two signals (A and B) combine to create Differential and Common mode signals:

- **Signal A** is fixed at 1V peak
- **Signal B** is **selectable**
- **Differential** is calculated as **A minus B**
- **Common** is calculated as the **AVERAGE** of **A and B**

**Note:** Click **Signal B** selections to see various Differential and Common signals.



**Notes:**

- Even when Signal B is 0V, like a Single -ended signal, there is still a unique Differential and Common mode representation of the two individual signals.
- The above model does not show a DUT. The difference and average of two signals can be calculated for both the balanced INPUT and balanced OUTPUT of a device.

**Measuring Mixed Mode (Balanced) S-Parameters**

Mixed mode S-parameters combine traditional S-parameter notation with balanced measurement terminology.

Some balanced devices are designed to amplify the differential component and reject the common component. This allows noise that is common to both inputs to be virtually eliminated from the output. For example, a balanced device may amplify the differential signal by a factor of 5, and attenuate the common signal by a factor of 5. Using



traditional S-parameter notation, an S21 is a ratio measurement of the device **Output** / device **Input**. Mixing this with balanced terminology, we could view the amplifier's Differential Output signal / Differential Input signal. To see this parameter on the PNA, we would select an Sdd21 measurement using the following balanced notation:

### Sabxy -

Where

**a** - device output mode

**b** - device input mode

(choose from the following for both a and b:)

- **d** - differential
- **c** - common
- **s** - single ended

**x** - device output "logical" port number

**y** - device input "logical" port number

### See Also

[Logical port mapping](#)

[Port mapping with External Test Sets](#)

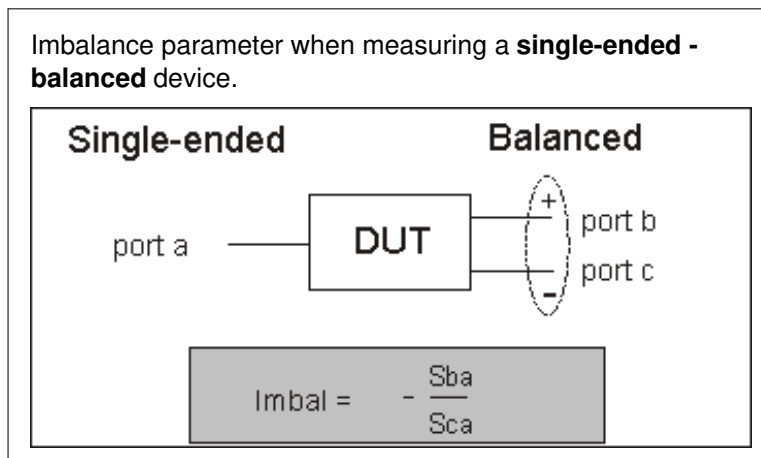
[iTMSA](#)

### Measuring Imbalance Parameters

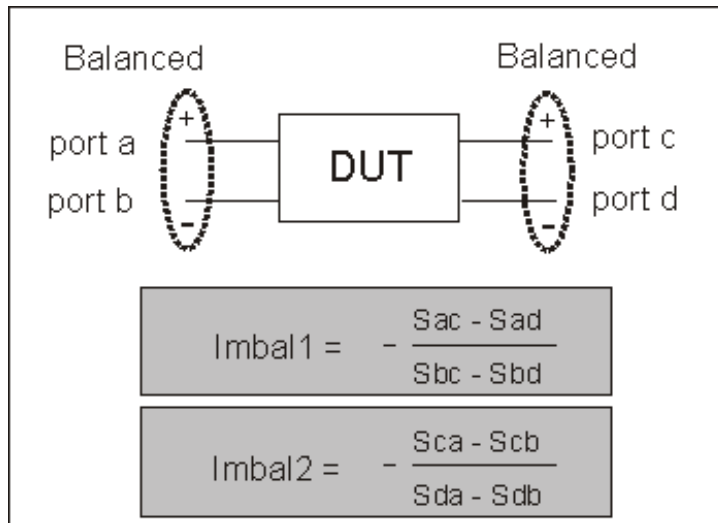
Imbalance is a measure of how well two physical ports that make up a balanced port are matched. With a perfectly balanced port, the same amount of energy flows to both ports and the magnitude of the ratio of these ports is 1.

The notation is similar to traditional S-parameters. In the following diagrams, the letters a, b, c, and d are used because any PNA port can be assigned to any logical port using the [port mapping process](#).

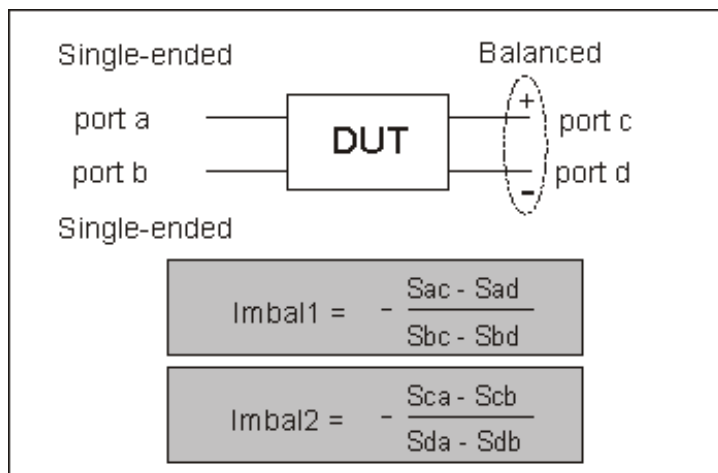
For example, in the following single-ended - balanced formula, **Sba** indicates the device output port is logical port b and the input port is logical port a.



Imbalance1 and Imbalance2 parameters when measuring a **balanced - balanced** device.



Imbalance1 and Imbalance2 parameters when measuring a **single-ended - single-ended - balanced** device.



### Measuring CMRR (Common Mode Rejection Ratio)

CMRR is a ratio of the transmission characteristic in differential mode over the transmission characteristic in the common mode of the balanced port as the measurement parameter. A high value indicates more rejection of common mode, which is desirable in a device that transmits information in the differential portion of the signal. The table below shows the CMRR parameter you can select when measuring each balanced device.

Single-ended - balanced device	$\frac{S_{ds21}}{S_{cs21}}$ and $\frac{S_{sd12}}{S_{sc12}}$
Balanced - balanced device	$\frac{S_{dd21}}{S_{cc21}}$
Single-ended - single-ended - balanced device	$\frac{S_{ds31}}{S_{cs31}}$ and $\frac{S_{ds32}}{S_{cs32}}$

### Device Topology and Port Mapping

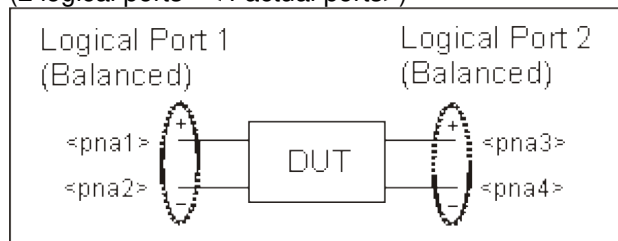
As we have seen on balanced inputs and outputs, the signal of interest is the difference or average of two BALANCED input or BALANCED output lines. It is also possible to have single-ended ports AND balanced ports on the same device. The two balanced input or output lines are referred to as a single "logical" port.

When configuring a balanced measurement on the PNA, select a device 'topology'. Then map each PNA test port to the DUT ports. The PNA assigns "logical ports". [See how to set device topology in the PNA.](#)

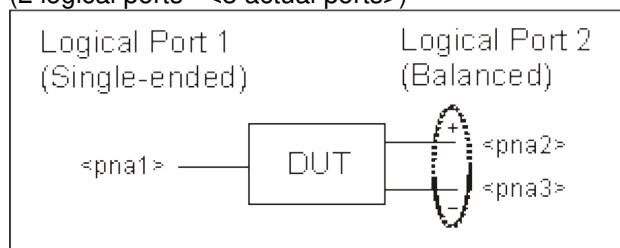
The screenshot shows the PNA configuration interface with the 'Balanced' tab selected. The 'Sdd21' checkbox is checked and circled in red. The 'Topology / Port Mapping' section at the bottom is also circled in red, showing 'BAL1: 1-2' and 'BAL2: 3-4'.

The following device topologies can be measured by a 4-port PNA.

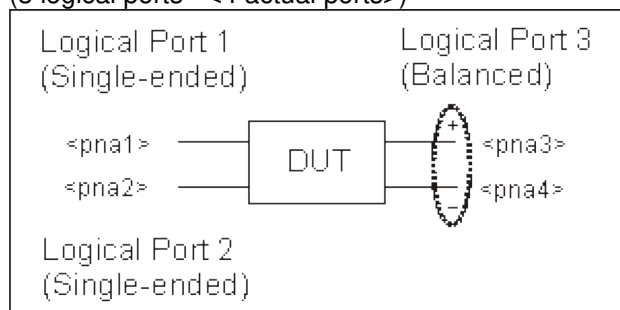
- **Balanced / Balanced**  
(2 logical ports - <4 actual ports>)



- **Single-ended / Balanced**  
(2 logical ports - <3 actual ports>)



- **Single-ended - Single-ended / Balanced**  
(3 logical ports - <4 actual ports>)



These topologies can be used in the reverse (<==) direction to measure:

- **Balanced / Single-ended** topology
- **Balanced / Single-ended - Single-ended** topology

For example, to measure a **Balanced / Single-ended** topology, measure the S12 (reverse direction) of a **Single-ended / Balanced** topology.

### How the PNA makes Balanced Measurements

When using standard Balanced measurements, the PNA does not provide true balanced measurements by stimulating both balanced inputs together and measuring both outputs relative to one another. Instead, the PNA makes only Single-ended measurements. On a Balanced/ Balanced device, it stimulates each input and measures each output individually. From the output data, the PNA calculates the Differential and Common outputs from the DUT using the same math formulas as the above model. However, all measurements and calculations on the PNA are performed in frequency domain using complex (magnitude and phase) data. The Balanced S-parameter display data is then calculated from the Differential and Common inputs and outputs.

In iTMSA mode, the PNA **DOES** stimulate both balanced inputs with true balanced sources. [Learn more.](#)

---

Last Modified:

15-May-2008    Edited for iTMSA

## Complex Impedance

---

When making an  $S_{11}$  or  $S_{22}$  measurement of your device under test, you can view complex-impedance data such as series resistance and reactance as well as [phase](#) and magnitude information. Complex impedance data can be viewed using either the Smith Chart format or the Polar format.

- [What Is Complex Impedance?](#)
- [Accuracy Considerations](#)
- [How to Measure Complex Impedance](#)

### What Is Complex Impedance?

Complex-impedance data is information that can be determined from an  $S_{11}$  or  $S_{22}$  measurement of your device under test, such as:

- Resistance
- Reactance
- Phase
- Magnitude

The amount of power reflected from a device is directly related to the impedances of both the device and the measuring system. For example, the value of the complex reflection coefficient (  $\Gamma$  ) is equal to 0 only when the device impedance and the system impedance are exactly the same (i.e. maximum power is transferred from the source to the [load](#)). Every value for  $\Gamma$  corresponds uniquely to a complex device impedance (as a function of frequency), according to the equation:

$$Z_L = [(1 + \Gamma) / (1 - \Gamma)] Z_0$$

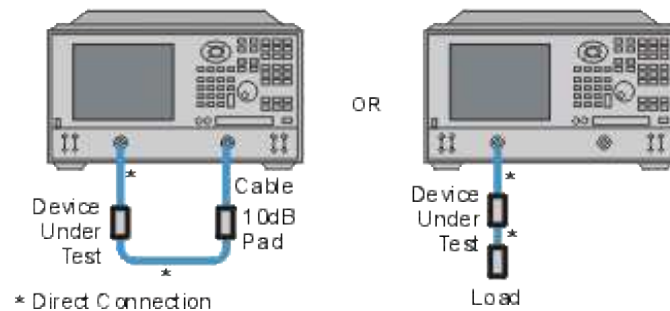
where  $Z_L$  is your test device impedance and  $Z_0$  is the measuring system's [characteristic impedance](#).

Complex Impedance is best viewed using either [Polar](#) or [Smith Chart](#) format.

### Accuracy Considerations

- The Smith chart is most easily understood when used with a full scale value of 1.0.
- For greater accuracy when using markers in the Smith chart or polar formats, activate the discrete marker mode.
- The uncertainty of reflection measurements is affected by:
  - [Directivity](#)
  - Reflection tracking
  - Source match
  - [Load match](#) (with 2-port devices)

With a 2-port [calibration](#), the effects of these factors are reduced. A 1-port calibration provides the same accuracy if the output of the device is well terminated. Refer to the graphic below for the following discussion.



- If you connect the device between both analyzer ports, it is recommended that you use a 10 dB pad on the output of the device to improve measurement accuracy. This is not necessary if you use a 2-port calibration since it corrects for load match.
- If you connect a two-port device to only one analyzer port, it is recommended that you use a high-quality [load](#) (such as a calibration standard) on the output of the device.

## How to Measure Complex Impedance

1. Connect the device as shown in the previous graphic.
2. Preset the analyzer.
3. Set up, calibrate, and perform an S11 or S22 measurement.
4. View impedance data:
  - a. Select the Smith Chart format.
  - b. Scale the displayed measurement for optimum viewing.
  - c. Position the marker to read the resistive and reactive components of the complex impedance at any point along the trace.
  - d. Print the data or save it to a disk.
5. View the magnitude and [phase](#) of the reflection coefficient:
  - a. Select the Smith chart format or the Polar format.
  - b. Select either Lin Marker or Log Marker formats.
  - c. Scale the displayed measurement for optimum viewing.
  - d. Position the marker to read the frequency, magnitude, and phase of the reflection coefficient ( ) at any point along the trace.
  - e. Print the data or save it to a disk.



## Comparing the PNA "Delay" Functions

---

The PNA has three Delay functions which are similar but are used in different ways.

1. **Group Delay format** is used to display the Group Delay of a network. Group Delay is defined as:

$-\frac{d(\phi)}{d(\omega)}$  -- where  $\phi$  is radian angle, and  $\omega$  is radian frequency.

Since it is defined by a derivative, the value must be determined from an analytic function. However, the PNA makes discrete measurements, so we approximate the group delay by taking the finite difference:

$-\frac{(1/360) \cdot \Delta(\phi)}{\Delta(f)}$  -- where  $\phi$  is degree angle and  $f$  is frequency in Hz. The 1/360 does the proper conversion of degrees to radians and Hz frequency to radian frequency.

From this we can see that, if the phase response of a network varies with frequency, then the Group Delay must vary as well. In fact, many filters are specified by the variation of their Group Delay.

If we measure the phase response of a lossless cable, it should be a straight line. But, of course, nothing is perfect. The phase response will have a small amount of noise. This is due to trace noise of the PNA, and the loss with real cables or transmission lines, which causes a small amount of non-linear phase change with frequency. So, if we look at the Group Delay of a cable, we will see a small amount of variation. Also, if the frequency spacing is small enough when you make the measurement, the  $\Delta(f)$  in the denominator becomes very small, so the delay can have wide swings with just a little noise.

To overcome this issue, we sometimes add smoothing to a phase trace, which widens the effective  $\Delta(f)$ , called the aperture, and provides a less noisy Group Delay response. The Group Delay of a device is only valid for a given frequency aperture. [Learn more about Group Delay.](#)

2. **Electrical Delay** function. On many filters, the passband response is specified for a maximum value of "Deviation from Linear Phase". When looking at the passband of a multi-pole filter, one sees the phase changing very rapidly. This makes it difficult to determine the linearity of the phase response. The Electrical Delay function subtracts out a "LINEAR PHASE" equivalent to the delay time value computed as above. When you use this function, you dial in the Linear Delay such that a CONSTANT PHASE SLOPE is removed from the phase trace, until the phase trace is mostly flat. The remaining variation is the deviation from linear phase.

To make this task a little less tedious, the PNA has a marker function called [Marker => Delay](#). This function computes the Group Delay value at the marker position, using a 20% smoothing aperture, then changes the Electrical Delay value to this value. Obviously, if the phase trace is not perfectly linear, moving the marker and recomputing the delay will result in different values. The phase slope added by the electrical delay function applies only to the current measurement. That is, each measurement (S11, S22, S12, S21) can have its own value of electrical delay. [Learn more about Deviation from Linear Phase.](#)

3. **Port Extension** is a function that is similar to calibration. It applies to all the traces in a given channel. It compensates for the phase response change that occurs when the calibration reference plane is not the same as the measurement plane of the device.

Let's look at an example of a DUT that is mounted on a PCB fixture with SMA connectors. We can easily calibrate at the SMA connectors. But if we add the fixture to measure the board-mounted device, the apparent phase of the DUT is changed by the phase of the PCB fixture. We use port extensions to add a LINEAR PHASE (constant delay) to the calibration routines to shift the phase reference plane to that of the DUT. This is ONLY valid if the fixture consists of a transmission line with linear phase response, and this limitation is usually met in practice. The main reason that it is NOT met is that there is mismatch at the SMA-to-PCB interface. This mismatch was not removed with the error correction because it occurs AFTER the SMA connector. Ripple can be seen on the display as signals bounce back and forth between the mismatch and the DUT. If the DUT is well matched, the ripple effect is very small. However, when we use Automatic Port Extension (APE), and we leave the fixture open (the DUT removed), the reflection is large and we see larger ripples. That is why APE uses a curve fitting process to remove the effects of the ripple. For best effect, the wider the IF Bandwidth, the better we can "smooth-out" the ripples with



curve fitting. Still, we are fitting a LINEAR PHASE SLOPE to the phase response, and thus we use only a single Port Extension Delay value to represent the phase slope.

The method used by older VNAs to get this same functionality was to add a mechanical line stretcher to the reference channel, which removed a fixed delay amount from the port. Port extensions give 1x the delay for transmission at each port, and 2x the delay for reflection, so it differs somewhat from Electrical Delay above, in that the math function depends upon the measurement being made. The signal passes twice through the fixture for reflection (out and back), but only once for each port on transmission. For S21, the phase slope added is the sum of the port 1 and port 2 Port Extension Delay values.

The "User Range" APE function is used in cases where a fixture has limited bandwidth, perhaps due to tuning elements or bias elements. In this case, the model of constant delay for the fixture over the whole bandwidth is not valid, so a narrower "User Range" of frequencies can be selected to compute the delay. Since the aperture is smaller, there is more uncertainty in the delay computation for port extension. Also, for those who had been using the [Marker ==> Delay](#) function to estimate the delay, we added the "Active Marker" selection to APE, which works exactly the same as Marker->Delay. [Learn more about Automatic Port Extensions.](#)

## Deviation from Linear Phase

---

Deviation from linear phase is a measure of phase distortion. The electrical delay feature of the analyzer is used to remove the linear portion of the phase shift from the measurement. This results in a high-resolution display of the non-linear portion of the phase shift (deviation from linear phase).

- [What Is Linear Phase Shift?](#)
- [What Is Deviation from Linear Phase?](#)
- [Why Measure Deviation from Linear Phase?](#)
- [Using Electrical Delay](#)
- [Accuracy Considerations](#)

See also [Comparing the PNA Delay Functions](#)

[See other Tutorials](#)

### What Is Linear Phase Shift?

Phase shift occurs because the wavelengths that occupy the electrical length of the device get shorter as the frequency of the incident signal increases. *Linear* phase-shift occurs when the phase response of a device is linearly proportional to frequency. Displayed on the analyzer, the phase-versus-frequency measurement trace of this ideal linear phase shift is a straight line. The slope is proportional to the electrical length of the device. Linear phase shift is necessary (along with a flat magnitude response) for distortionless transmission of signals.

### What Is Deviation from Linear Phase?

In actual practice, many electrical or electronic devices will delay some frequencies more than others, creating non-linear phase-shift (distortion in signals consisting of multiple-frequency components). Measuring deviation from linear phase is a way to quantify this non-linear phase shift.

Since it is only the deviation from linear phase which causes phase distortion, it is desirable to remove the linear portion of the phase response from the measurement. This can be accomplished by using the electrical delay feature of the analyzer to mathematically cancel the electrical length of the device under test. What remains is the deviation from linear phase, or phase distortion.

### Why Measure Deviation from Linear Phase?

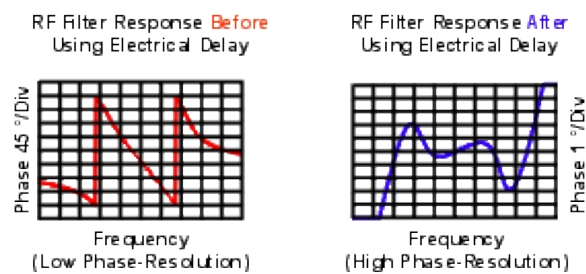
The deviation from linear phase measurement accomplishes the following:

- Presents data in units of phase rather than units of seconds (group delay). For devices that pass modulated signals, units of phase may be most practical.
- Provides a less noisy measurement than a [group delay](#) measurement.

## Using Electrical Delay

The electrical delay feature is the electronic version of the mechanical "line stretcher" of earlier analyzers. This feature does the following:

- Simulates a variable-length lossless transmission line, which is effectively added to or removed from the reference signal path.
- Compensates for the electrical length of the device under test.
- Flattens the measurement trace on the analyzer's display. This allows the trace to be viewed at high resolution in order to see the details of the phase nonlinearity.
- Provides a convenient method to view the deviation from linear phase of the device under test. See the following graphic.



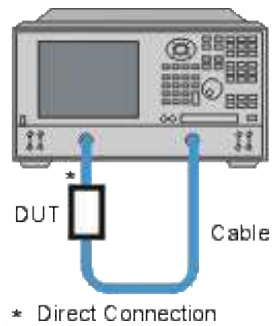
[Learn how to set Electrical Delay.](#)

## Accuracy Considerations

The frequency response of the test setup is the dominant error in a deviation from linear phase measurement. To reduce this error, perform a 2-port measurement calibration.

## How to Measure Deviation from Linear Phase:

1. Preset the analyzer.
2. If your device under test is an amplifier, it may be necessary to adjust the analyzer's source power:
  - Set the analyzer's source power to be in the linear region of the amplifier's output response (typically 10-dB below the 1-dB compression point).
  - Select an external attenuator (if needed) so the amplifier's output power will be sufficiently attenuated to avoid causing receiver compression or damage to the analyzer's port 2.
3. Connect the device under test as shown in the following graphic.



3. Select an S21 measurement.
4. Select the settings for your device under test, including the following:
  - [Format](#): phase
  - [Scale](#): autoscale
5. Remove the device and perform a calibration.
6. Reconnect the device.
7. Scale the displayed measurement for optimum viewing.
8. [Create a marker](#) in the middle of the trace.
9. Press the **>Delay** key to invoke the [Marker to Electrical Delay](#) function. This flattens the phase trace.
10. If desired, on the **Scale** menu, click [Electrical Delay](#) to fine-tune the flatness of the phase trace.
11. Use the markers to measure the maximum peak-to-peak deviation from linear phase.
12. Print the data or save it to a disk.

## Small Signal Gain and Flatness

---

Small signal gain is the gain in the amplifier's linear region of operation. This is typically measured at a constant input power over a swept frequency. Gain **flatness** is the measure of the variation of gain over a specified frequency range.

- [What Is Gain?](#)
- [What Is Flatness?](#)
- [Why Measure Gain and Flatness?](#)
- [Accuracy Considerations](#)
- [How to Measure Gain and Flatness](#)

[See other Amplifier Parameter topics](#)

### What Is Gain?

RF amplifier gain is defined as the difference in power between the amplifier output signal and the input signal. It is assumed that both input and output impedances of the amplifier are the same as the **characteristic impedance** of the system.

- Gain is called  $S_{21}$  using S-parameter terminology
- Gain is expressed in dB—a logarithmic ratio of the output power relative to the input power.
- Gain can be calculated by subtracting the input from the output levels when both are expressed in dBm, which is power relative to 1 milliwatt.
- Amplifier gain is most commonly specified as a minimum value over a specified frequency range. Some amplifiers specify both minimum and maximum gain, to ensure that subsequent stages in a system are not under or over driven.

### What Is Flatness?

**Flatness** specifies how much the amplifier's gain can vary over the specified frequency range. Variations in the flatness of the amplifier's gain can cause distortion of signals passing through the amplifier.

### Why Measure Small-Signal Gain and Flatness?

Deviations in gain over the **bandwidth** of interest will induce distortion in the transmitted signal because frequency components are not amplified equally. Small-signal gain allows you to quantify the amplifier's gain at a particular frequency in a 50-ohm system. Flatness allows you to view the deviations in the amplifier's gain over a specified frequency range in a 50-ohm system.

## Accuracy Considerations

- The amplifier may respond very differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.
- The output power of the amplifier should be sufficiently attenuated if necessary. Too much output power could:
  - damage the analyzer receiver
  - exceed the input compression level of the analyzer receiver, resulting in inaccurate measurements.

Attenuation of the amplifier's output power can be accomplished using:

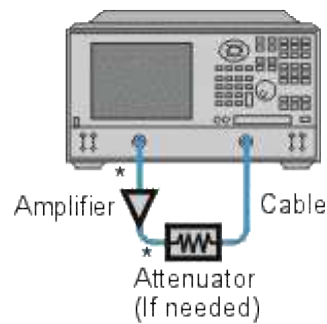
- attenuators
- couplers

The frequency-response effects and mismatches of the attenuators and couplers must be accounted for during calibration since they are part of the test system. Proper error-correction techniques can reduce these effects.

- The frequency response is the dominant error in a small-signal gain and flatness measurement setup. Performing a thru-response measurement calibration significantly reduces this error. For greater accuracy, perform a 2-port measurement calibration.
- Reducing IF bandwidth or using averaging improves measurement dynamic range and accuracy, at the expense of measurement speed.

## How to Measure Gain and Flatness

1. Preset the analyzer.
2. Select an S21 measurement parameter.
3. Set the analyzer's source power to be in the linear region of the amplifier's output response (typically 10-dB below the 1-dB compression point).
4. Select an external attenuator (if needed) so the amplifier's output power will be sufficiently attenuated to avoid causing receiver compression or damage to the analyzer's port-2.



\* Direct Connection

5. Connect the amplifier as shown in the following graphic, and provide the dc bias.
6. Select the analyzer settings for your amplifier under test.
7. Remove the amplifier and perform a measurement calibration. Be sure to include the attenuator and cables in the calibration setup if they will be used when measuring the amplifier.
8. Save the instrument-state to memory.
9. Reconnect the amplifier.
10. Scale the displayed measurement for optimum viewing and use a marker to measure the small signal gain at a desired frequency.
11. Measure the gain flatness over a frequency range by using markers to view the peak-to-peak ripple.
12. Print or save the data to a disk.
13. This type of measurement can be automated.

## Gain Compression

---

Gain compression measures the level of input power applied to an amplifier that will cause a distorted output. The [Gain Compression Application](#) (Opt 086) makes fast and accurate compression measurements.

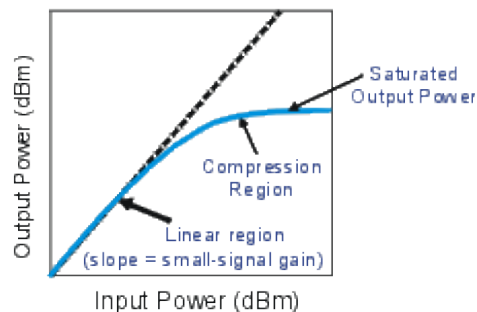
- [What Is Gain Compression?](#)
- [Why Measure Gain Compression?](#)
- [Accuracy Considerations](#)
- [How to Measure Gain Compression](#)

[See other Amplifier Parameter topics](#)

### What Is Gain Compression?

Gain compression occurs when the input power of an amplifier is increased to a level that reduces the gain of the amplifier and causes a nonlinear increase in output power.

The analyzer has the ability to do power sweeps as well as frequency sweeps. Power sweeps help characterize the nonlinear performance of an amplifier. Refer to the graphic below (a plot of an amplifier's output power versus input power at a single frequency) for the following discussion.



- The amplifier has a linear region of operation where gain is constant and independent of power level. The gain in this region is commonly referred to as "small-signal gain."
- As the input power increases, the amplifier gain appears to decrease, and the amplifier goes into compression.
- The most common measurement of amplifier compression is the 1-dB compression point. This is defined as the input power (or sometimes the output power) which results in a 1-dB decrease in amplifier gain (relative to the amplifier's small-signal gain).

### Why Measure Gain Compression?

When driven with a sinusoid, the output of an amplifier is no longer sinusoidal in the compression region. Some of the amplifier output appears in harmonics, rather than occurring only at the fundamental frequency of the input



signal.

As input power is increased even more, the amplifier becomes saturated, and output power remains constant. At this point, further increases in amplifier input power result in no change in output power.

In some cases (such as with TWT amplifiers), output power actually decreases with further increases in input power after saturation, which means the amplifier has negative gain.

Since gain is desired in amplifier operation, it is important to know the limit of input signal that will result in gain compression.

## Accuracy Considerations

The network analyzer must provide sufficient power to drive the amplifier into saturation. If you need a higher input-power level than the source of the analyzer can provide, use a preamplifier to boost the power level prior to the amplifier under test. (See [High-Power Component Measurements](#).) If using a preamplifier, you can increase measurement accuracy in the following ways:

- Use a coupler on the output of the preamplifier so that a portion of the boosted input signal can be used for the analyzer's reference channel. This configuration removes the preamplifier's frequency response and drift errors from the measurement (by ratioing).
- Perform a thru-response calibration including the preamplifier, couplers, and attenuators in the test setup.

The output power of the amplifier should be sufficiently attenuated if necessary. Too much output power could:

- Damage the analyzer receiver
- Exceed the input compression level of the analyzer receiver

Attenuation of the amplifier's output power can be accomplished using:

- Attenuators
- Couplers

The frequency-response effects of the attenuators and couplers must be considered during calibration since they are part of the test system. Proper error-correction techniques can reduce these effects.

- The frequency response is the dominant error in a gain compression measurement setup. Performing a thru-response measurement calibration significantly reduces this error.
- The amplifier may respond very differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.
- Reducing IF bandwidth or using measurement averages improves accuracy, at the expense of measurement speed.

## How to Measure Gain Compression

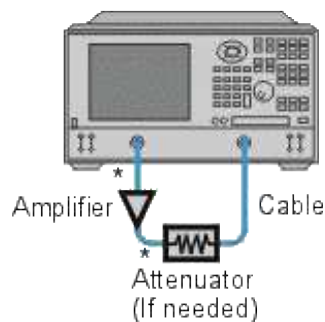
This procedure shows you how to make the following three measurements used to determine amplifier gain compression:

1. A [Swept-Frequency Gain Compression](#) measurement locates the lowest frequency at which the 1-dB gain compression first occurs.
2. A [Swept-Power Gain Compression](#) measurement shows the input power at which a 1-dB drop in gain occurs as a power ramp is applied to the amplifier at a particular frequency point (found in measurement 1).
3. An [Absolute Power](#) measurement shows the absolute power out (in dBm) at compression.

## Swept-Frequency Gain Compression Measurement

A measurement of swept frequency gain compression locates the frequency point where 1-dB compression first occurs.

1. Preset the analyzer.
2. Select an  $S_{21}$  measurement parameter.
3. Set the analyzer's source power to be in the linear region of the amplifier's output response (typically 10-dB below the 1-dB compression point).
4. Select an external attenuator (if needed) so the amplifier's output power will be sufficiently attenuated to avoid causing receiver compression or damage to the analyzer's port-2.
5. Connect the amplifier as shown in the following graphic, and provide the dc bias.
6. Select the analyzer settings for your amplifier under test. To reduce the effects of noise, you may want to specify a narrower IF bandwidth.



\* Direct Connection

7. Remove the amplifier and perform a thru-response calibration. Be sure to include the attenuator and cables in the calibration setup if they will be used when measuring the amplifier.
8. Save the instrument-state to memory.
9. Reconnect the amplifier.
10. Position a marker at approximately mid-span.
11. Adjust the analyzer's scale to 1 dB per division.

12. Store the trace in memory and display Data/Mem.
13. Gradually increase the source power until a 1-dB decrease in gain is observed at the first frequency over some portion of the trace.
14. Use markers to locate the frequency where the 1-dB decrease in gain first occurs. Note this frequency for use in the following measurement.
15. Print the data or save it to a disk.

### **Swept-Power Gain Compression Measurement**

A swept-power gain compression measurement shows the input power resulting in a 1-dB drop in gain as a power ramp at a particular frequency (found in step 13 of the previous measurement) is applied to the amplifier.

1. If not already done, perform the previous measurement of swept-frequency gain compression.
2. Setup an  $S_{21}$  measurement in the power-sweep mode. Include the following settings:
  - Set the CW frequency to the frequency noted in step 14 of the previous measurement of swept-frequency gain compression.
  - Enter the start and stop power levels for the sweep. The start power should be in the linear region of the amplifier's response (typically 10 dB below the 1-dB compression point). The stop power should be in the compression region of the amplifier's response.
3. Adjust the scale to 1-dB per division.
4. Use markers (including reference marker) to find the input power where the 1-dB decrease in gain occurs.
5. Print the data or save it to a disk.

### **Absolute Output Power Measurement**

An absolute-power measurement shows the absolute power-out (in dBm) of the amplifier at compression.

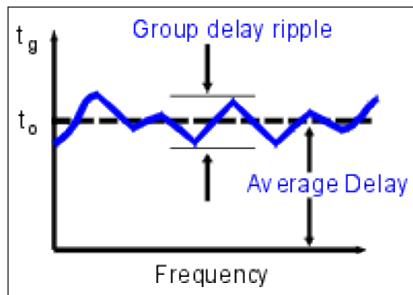
1. Select an unratiod (absolute) power measurement. Choose the B input if using the test setup in the previous graphic.
2. Retain the CW frequency used in the previous measurement of swept-power gain compression.
3. Set a marker to the input power level where the 1-dB decrease in gain occurs (found in step 4 of the previous measurement).
4. Scale the displayed measurement for optimum viewing.
5. Read the marker value to find the absolute output power of the amplifier (in dBm) where the 1-dB decrease in gain occurs.
6. Print the data or save it to a disk.

**Note:** The measurement calibration does not apply to absolute power. Therefore, if there is any attenuation external to the analyzer, you will have to correct for it manually.

---

Linear phase-shift component:	Higher-order phase-shift component:
Represents average signal transit time.	Represents variations in transit time for different frequencies.
Attributed to electrical length of test device.	Source of signal distortion.

Refer to the graphic below for the following discussion:



In a group delay measurement:

- The linear phase shift component is converted to a constant value (representing the average delay).
- The higher order phase shift component is transformed into deviations from constant group delay (or group delay ripple).
- The deviations in group delay cause signal distortion, just as deviations from linear phase cause distortion.
- The measurement trace depicts the amount of time it takes for each frequency to travel through the device under test.

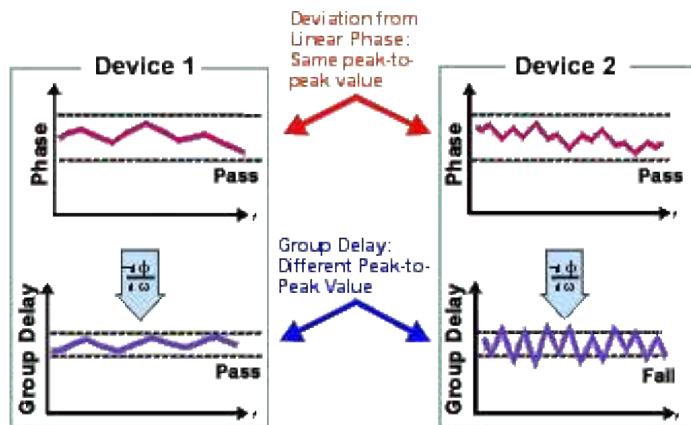
Refer to the following equation for this discussion on how the PNA computes group delay:

$$\begin{aligned}
 \text{Group Delay} = t_g &= \frac{-d\phi}{d\omega} && \begin{array}{l} \phi \text{ in Radians} \\ \omega \text{ in Radians/Sec} \end{array} \\
 &= \frac{-1}{360^\circ} \cdot \frac{d\Theta}{df} && \begin{array}{l} \Theta \text{ in Degrees} \\ f \text{ in Hz } (\omega = 2\pi f) \end{array}
 \end{aligned}$$

- Phase data is used to find the phase change ( $-d$  ).
- A specified frequency aperture is used to find the frequency change ( $d$  ).
- Using the two values above, an approximation is calculated for the rate of change of phase with frequency.
- This approximation represents group delay in seconds (assuming linear phase change over the specified frequency aperture).

## Group Delay versus Deviation from Linear Phase

Group delay is often a more accurate indication of phase distortion than [Deviation from Linear Phase](#).



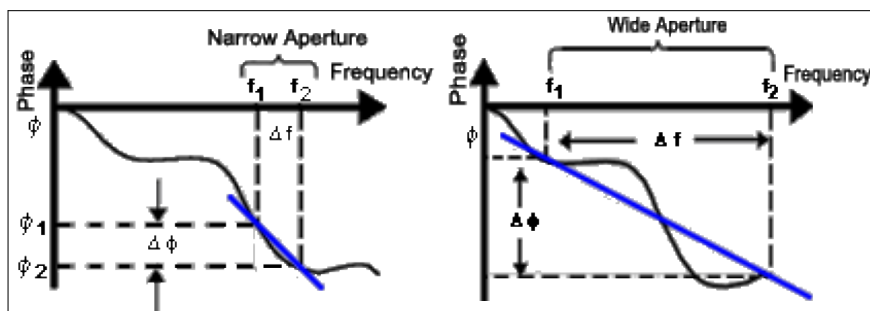
**Deviation from linear phase** results are shown in the upper region of the following graphic: Device 1 and device 2 have the same value, despite different appearances.

**Group Delay** results are shown in the lower region: Device 1 and device 2 have different values of group delay. This is because in determining group delay, the analyzer calculates slope of phase ripple, which is dependent on number of ripples which occur per unit of frequency.

## What Is Aperture?

During a group delay measurement, the PNA measures the phase at two closely spaced frequencies and then computes the phase slope. The frequency interval (frequency delta) between the two phase measurement points is called the aperture. Changing the aperture can result in different values of group delay. The computed slope ( -  $\Delta \phi / \Delta f$  ) varies as the aperture is increased. This is why when you are comparing group delay data, you must know the aperture that was used to make the measurements.

Refer to the graphic below for the following discussion:



Narrow aperture:	Wide aperture:
Provides more detail in phase linearity.	Provides less detail in phase linearity because some phase response averaged-out or not measured.
Makes measurement susceptible to noise (smaller signal-to-noise ratio) and PNA phase detector resolution.	Makes measurement less susceptible to noise (larger signal-to-noise ratio).

Group delay measurements can be made using the following [sweep types](#):

- Linear frequency
- List frequency sweep segment - The group delay aperture varies depending on the frequency spacing and point density. Therefore the aperture is not constant in segment sweep. In segment sweep, extra frequency points can be defined to ensure the desired aperture.

### How to set Group Delay Aperture

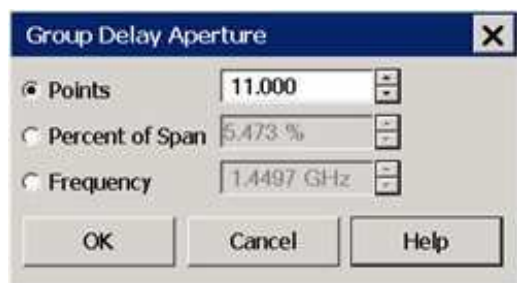
#### Using front-panel HARDKEY [softkey] buttons

1. Press **AvG**
2. then **[More]**
3. then **[Group Delay Aperture]**

#### PNA Menu using a mouse

1. Click **Response**
2. then **AvG**
3. then **Group Delay Aperture**

### Programming Commands





## Group Delay Aperture dialog box help

Although the Group Delay Aperture is defined as the difference in frequency between two data points (see [What Is Aperture?](#)), the group delay calculation can be averaged over many adjacent data points, similar to the PNA smoothing feature. The number of adjacent data points can be set using any of the following methods:

**Note:** You can change the default Group Delay Aperture to two points using a PNA Preference. [Learn how.](#)

**Points** Number of adjacent data points to average. Default setting is 11 points. Choose a value between 2 and the current number of points in the channel.

**Percent of Span** The data points within this percentage of the current frequency span are averaged. Choose a value between (2 points / current number of points) and 100 percent. The span must contain at least two data points.

**Frequency** The data points within this frequency range are averaged. The frequency range must contain at least two data points.

When the frequency span or number of points is reduced so that the current Group Delay Aperture is NOT attainable, the Aperture is adjusted to the new frequency span or number of points.

**OK** Applies setting changes and closes the dialog box.

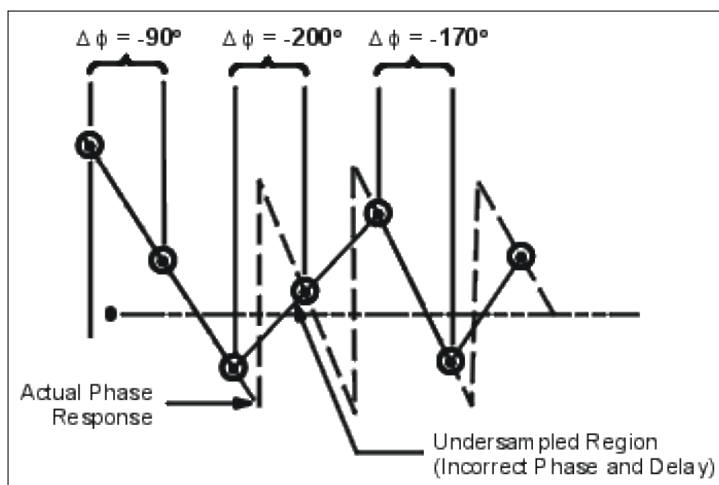
**Cancel** Closes the dialog. Setting changes are NOT applied.

## Accuracy Considerations

It is important to keep the phase difference between two adjacent measurement points less than  $180^\circ$  (see the following graphic). Otherwise, incorrect phase and delay information may result. Undersampling may occur when measuring devices with long electrical length. You can verify that the phase difference measured between two adjacent points is less than  $180^\circ$  by adjusting the following settings until the measurement trace no longer changes:

- Increase the number of points
- Narrow the frequency span

Electrical delay may also be used to compensate for this effect.



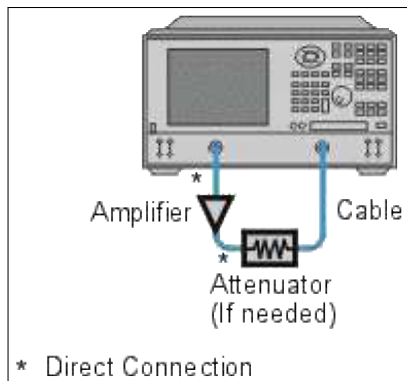
The frequency response is the dominant error in a group delay test setup. Performing a thru-response

measurement calibration significantly reduces this error. For greater accuracy, perform a 2-port measurement calibration.

Particularly for an amplifier, the response may vary differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.

## How to Measure Group Delay

1. Preset the analyzer.
2. If your DUT is an amplifier, it may be necessary to adjust the PNA source power:
  - Set the source power to be in the linear region of the amplifier's output response, typically 10 dB below the 1 dB compression point.
  - If needed, use an external attenuator so the amplifier output power will be sufficiently attenuated to avoid causing receiver compression or damage to the PNA port 2.
3. Connect the DUT as shown in the following graphic.



4. Select an S21 measurement.
5. Select the settings for your DUT:
  - frequency range
  - number of measurement points.
  - format: delay
  - scale: autoscale
6. Remove the DUT and perform a measurement calibration.
7. Reconnect the DUT.
8. Scale the displayed measurement for optimum viewing.
9. Use the Group Delay Aperture setting to increase the aperture, reducing noise on the trace while maintaining

meaningful detail.

10. Use the markers to measure group delay (expressed in seconds) at a particular frequency of interest.
11. Print the data or save it to a disk.

---

Last Modified:

23-Feb-2010    Added new dialog and text

4-Jan-2010    Fixed default aperture formula

## High-Gain Amplifier Measurements

When measuring High-Gain Amplifiers, errors in measuring any of the S-parameters during calibration can result in error in the S21 measurement. This is because all the S-parameters are used in the error correction math.

A particular problem occurs with high gain amplifiers because the source power is set very low. Thus, when making reverse measurements (S22, S12) the signal-to-noise is poor and the raw measurements can be dominated by noise. This noise in the raw measurements will result in a noisy trace appearing for corrected S21 or S11.

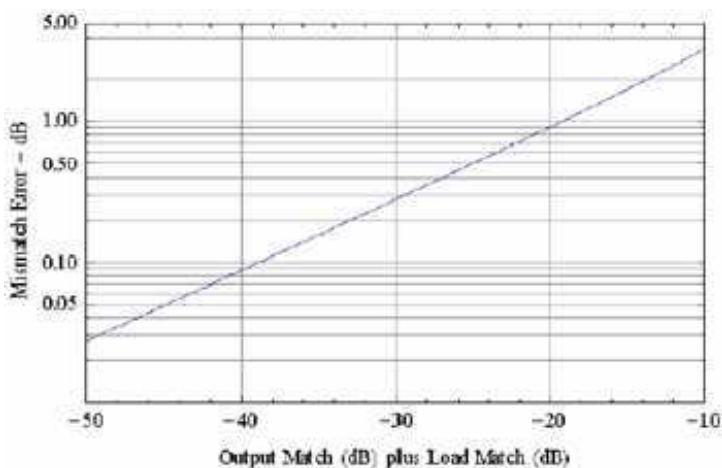
If you are using a large attenuator on port 2 (which improves output match), perform an Enhanced Response Calibration as follows. This corrects for the same errors as the full 2-port correction EXCEPT the interaction between the raw load match and the DUT output match.

1. There is NO need to Uncouple the port powers.
2. Set port powers to an acceptable level. Do NOT overpower the PNA test port.
3. Perform Enhanced Response Cal. [Learn how.](#) (Does not measure or correct for the S12 or S22 / PNA port match).

If you want to do a full correction (for example, when your amplifier output match is poor so the Enhanced Response Cal above is not adequate), then...

1. Uncouple the port powers. [Learn how.](#)
2. Set input (port 1) power to approximately the output power of the amplifier up to 0 dBm
3. Set reverse (port 2) power to the same power (for measuring isolation and S22)
4. Perform a Full 2-port Cal.
5. Re-set the input power (port 1) to a lower power level appropriate for driving the amplifier.

Additional Error due to Mismatch of DUT Output Match and Raw Load Match



---

Last Modified:

23-Sep-2009    MX New topic

## High-Power Amplifier Measurements Using a PNA

---

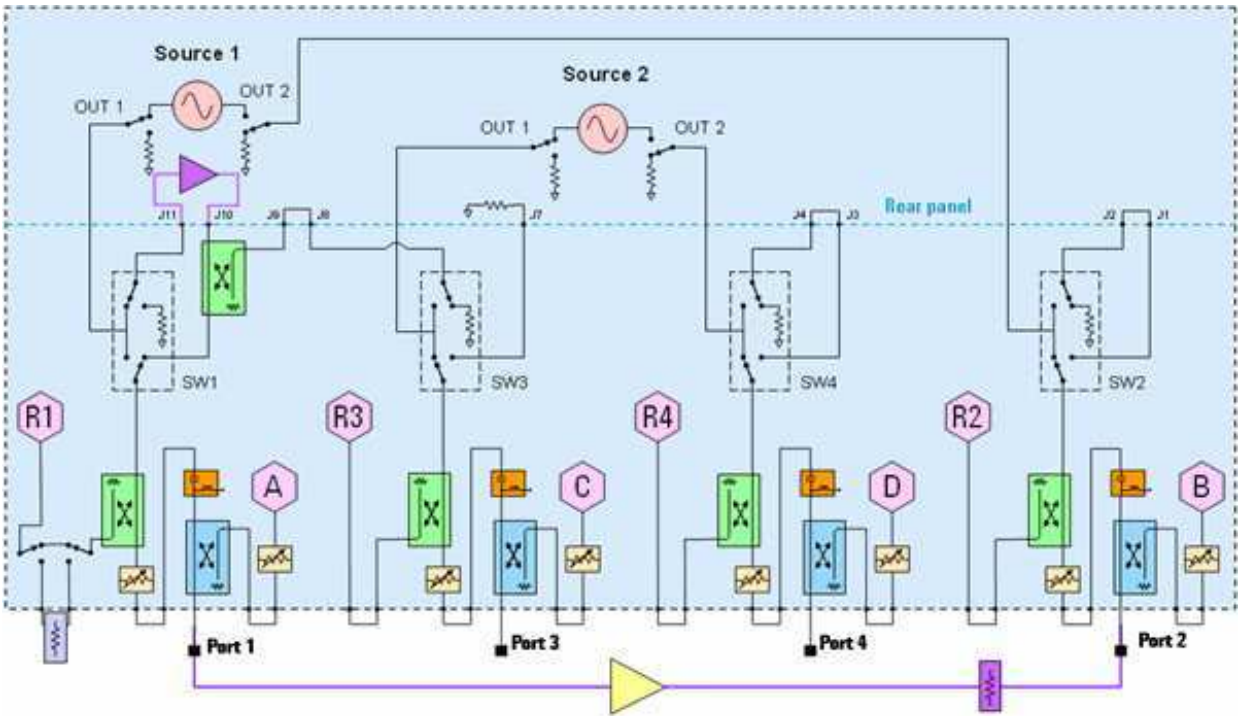
This topic is now covered in detail in Application Note 1408-10, [High-power measurements using the PNA \(5989-1349EN\)](#) at Agilent.com.

### See Also

[High-Power Amplifier Measurements using a PNA-X](#)

# High-Power Amplifier Measurements with the PNA-X

The following is a block diagram of the [PNA-X Opt 423](#). The configuration displayed here is used to make high power amplifier measurements using a preamplifier at the rear panel. The preamplifier can then be switched (SW1) as needed using the [RF Configurator](#).



## Legend

Color	Component	Damage Level
Green	Bridges	+33 dBm
Blue	Couplers	+43 dBm
Orange	Bias-tees	+30 dBm
Purple	User-supplied pre-amp and high-power attenuator	N/A

## Notes

**At J11** (rear-panel), max power is 4 dB to 11 dB higher than Source 1 Out at front panel jumper due to loss of the coupler thru arms, bias-tees, and cables.

**At J10** max power +33 dBm, which is the damage level of the bridge. With +30 dBm into J10, there will be about +15 dBm at R1, assuming 15 dB coupling factor for the R1 bridge. +15 dBm is the damage level of that receiver. Therefore, it may be necessary to add attenuation in place of the R1 loop, not only to protect the receiver, but to bring it out of compression. The 0.1 dB compression level spec for the R1 receiver is between -3 and -18 dBm, depending on the frequency and option configuration.

**At Test Port 2** (DUT output): With the bias-tees (orange), only +30 dBm is allowed into the test port. With Opt H85 (bias-tees removed), +43 dBm is allowed. Add appropriate attenuation to not damage other components.

### About the PNA-X Option H85

A modified version of the standard PNA-X, the option H85 is designed to permit insertion of high power amplifiers and other signal conditioning equipment to allow high power network measurements at RF levels up to 20 Watts (+43 dBm) from 10 MHz to 26.5 GHz. The Option H85 modification supplies extended power range attenuators without bias tees. This is similar to the PNA-X -219 (add extended power range and bias-tees to 2-Port analyzer) or PNA-X -419 (add extended power range and bias-tees to 4-Port analyzer) but deletes the bias tees from the test set.

### See Also

- [High-Gain Amplifier Measurements](#)
- [N5242-H85 Manual](#)
- [RF Path Configurator](#)
- [IF Path Configurator](#)

---

#### Last Modified:

- |             |                            |
|-------------|----------------------------|
| 6-Apr-2009  | Replaced N5242A with PNA-X |
| 14-Aug-2008 | Added link to H85 manual   |
| 10-May-2007 | MX New topic               |



## Phase Measurements

---

Knowledge of both magnitude and phase characteristics is needed for successful higher-level component integration.

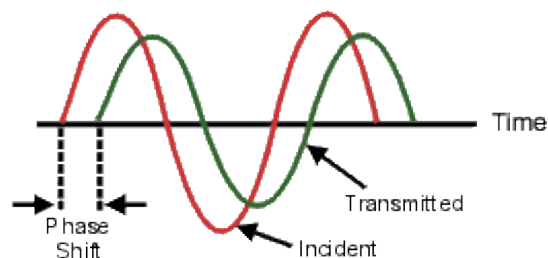
- [What are Phase Measurements?](#)
- [Why Measure Phase?](#)
- [Using the Analyzer's Phase Format](#)
- [Types of Phase Measurements](#)

[See other Tutorials](#)

### What are Phase Measurements?

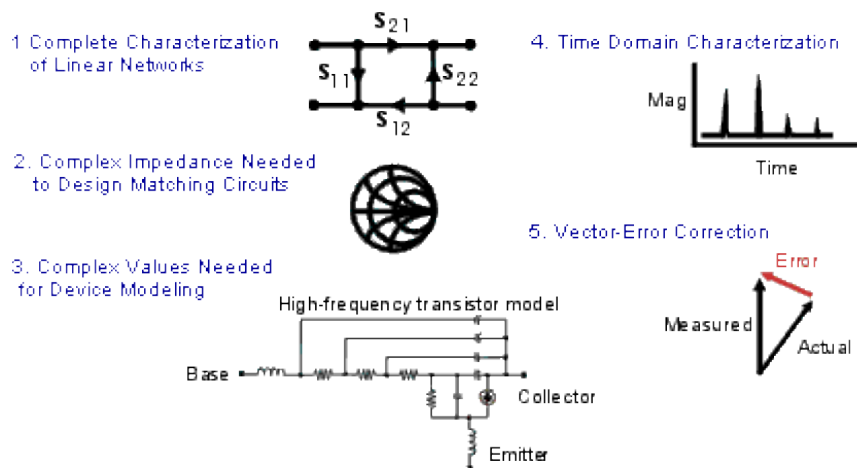
Phase measurements are made using S-parameters, just like amplitude measurements. A phase measurement is a relative (ratio) measurement and not an absolute measurement. Phase measurements compare the phase of the signal going into a device (the incident signal) to the phase of the device's response signal. The response signal can be either reflected or transmitted. Assuming an accurate [calibration](#) has been performed, the difference in phase between the two signals (known as phase shift) is a result of the electrical characteristics of the device under test.

The following graphic shows the phase shift (in time or degrees) between an incident signal and a transmitted signal (as might be seen on an oscilloscope display).



### Why Measure Phase?

Measuring [phase](#) is a critical element of network analysis. The following graphic lists five reasons for measuring both magnitude and phase.



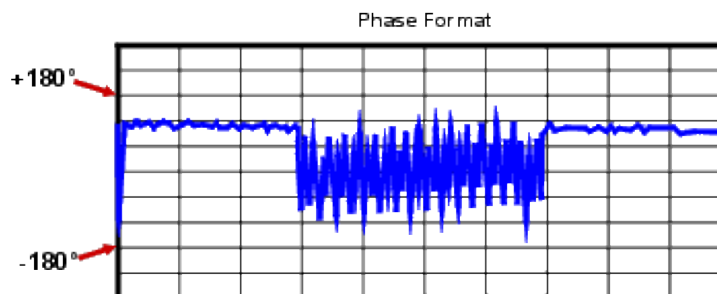
When used in communications systems to pass signals, components or circuits must not cause excessive signal distortion. This distortion can be:

- Linear, where flat magnitude and linear phase shift versus frequency is not maintained over the bandwidth of interest.
- Nonlinear, such as AM-to-PM conversion.

It is important to measure how reflective a component or circuit is, to ensure that it transmits or absorbs energy efficiently. Measuring the complex impedance of an antenna is a good example.

## Using the Analyzer's Phase Format

The analyzer's phase format displays a phase-versus-frequency or phase-versus-power measurement. The analyzer does not display more than  $\pm 180$  degrees phase difference between the reference and test signals. As the phase value varies between  $+180$  degrees and  $-180$  degrees, the analyzer display creates the sawtooth pattern as shown in the following graphic.



The sawtooth pattern does not always reach  $+180$  degrees and  $-180$  degrees. This is because the measurement is made at discrete frequencies, and the data point at  $+180$  degrees and  $-180$  degrees may not be measured for the selected sweep.

## Types of Phase Measurements

Complex impedance data is information such as resistance, reactance, phase, and magnitude that can be

determined from an S11 or S22 measurement. Complex impedance data can be viewed using either the Smith Chart format or the Polar format.

[AM-to-PM conversion](#) is a measure of the amount of undesired phase deviation (PM) that is caused by amplitude variations (AM) of the system. AM-to-PM conversion is usually defined as the change in output phase for a 1-dB increment in the input power to an amplifier (i.e. at the 1 dB gain compression point). This is expressed in degrees-per-dB (°/dB).

[Deviation from linear phase](#) is a measure of phase distortion caused by a device. Ideally, the phase shift through a device is a linear function of frequency. The amount of variation from this theoretical phase shift is known as its deviation from linear phase (also called phase linearity).

[Group delay](#) is another way to look at phase distortion caused by a device. Group delay is a measure of transit time through a device at a particular frequency. The analyzer computes group delay from the derivative of the measured phase response.

## Deviation from Linear Phase Versus Group Delay

Although deviation from linear phase and group delay are similar measurements, they each have their purpose.

The following are the advantages of deviation from linear phase measurements:

- Less noisy than group delay.
- Able to characterize devices that pass phase modulated signals, and show units of phase rather than units of seconds.

The following are the advantages of group delay measurements:

- More easily interpreted indication of phase distortion than deviation from linear phase.
- Able to most accurately characterize a device under test. This is because in determining group delay, the analyzer calculates the slope of the phase ripple, which is dependent on the number of ripples which occur per unit of frequency. Comparing two phase responses with equal peak-to-peak phase ripple, the response with the larger phase slope results in:
  - More group delay variation.
  - More signal distortion.

See also [Comparing the PNA Delay Functions](#).

## Reverse Isolation

---

Reverse isolation is a measure of amplifier reverse transmission response- from output to input.

- [What is Reverse Isolation](#)
- [Why Measure Reverse Isolation?](#)
- [Accuracy Considerations](#)
- [How to Measure Reverse Isolation](#)

[See other Tutorials](#)

### What is Reverse Isolation?

Reverse isolation is a measure of how well a signal applied to the device output is "isolated" from its input.

The measurement of reverse isolation is similar to that of forward gain, except:

- The stimulus signal is applied to the amplifier's output port.
- The response is measured at the amplifier's input port.

The equivalent S-parameter is S12.

### Why Measure Reverse Isolation?

An ideal amplifier would have infinite reverse isolation-no signal would be transmitted from the output back to the input. However, reflected signals can pass through the amplifier in the reverse direction. This unwanted reverse transmission can cause the reflected signals to interfere with the desired fundamental signal flowing in the forward direction. Therefore, reverse isolation is important to quantify.

### Accuracy Considerations

Since amplifiers often exhibit high loss in the reverse direction, generally there is no need for any [attenuation](#) that may have been used to protect the port 2 receiver during forward transmission measurements. Removing the attenuation will:

- Increase the [dynamic range](#), resulting in improved measurement accuracy.
- Require a new [calibration](#) for maximum accuracy.

The RF source power can be increased to provide more dynamic range and accuracy.

**Note:** With the attenuation removed and the RF source power increased, a forward sweep could damage the analyzer's port 2 receiver. Do not perform a forward sweep or use 2-port calibration unless the forward power is set low enough to avoid causing port 2 receiver compression or damage.

If the isolation of the amplifier under test is very large, the transmitted signal level may be near the noise floor or crosstalk level of the receiver. To lower the noise floor:

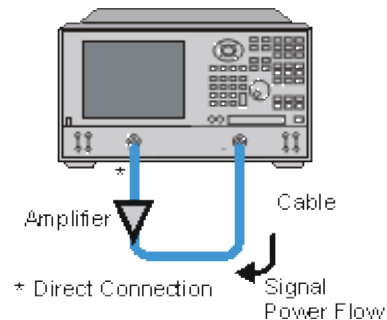
- Use or increase measurement averages.
- Reduce the IF bandwidth of the analyzer.

**Note:** Reducing IF bandwidth or using averaging improves measurement dynamic range and accuracy, at the expense of reduced measurement speed.

- When crosstalk levels affect the measurement accuracy, reduce the crosstalk error term by performing a response and isolation calibration. When performing the isolation part of the calibration it is important to use the same average factor and IF bandwidth during the calibration and measurement.
- The frequency response of the test setup is the dominant error in a reverse isolation measurement. Performing a thru-response measurement calibration significantly reduces this error. This calibration can be done as part of the response and isolation calibration.
- The amplifier may respond very differently at various temperatures. The tests should be done when the amplifier is at the desired operating temperature.

## How to Measure Reverse Isolation

1. Connect the amplifier as shown in the following graphic.



2. Preset the analyzer.
3. Select an S12 measurement.
4. Select the settings for your amplifier under test.
5. Remove the amplifier and perform a thru-response calibration or a response and isolation calibration.
6. Scale the displayed measurement for optimum viewing and use a marker to measure the reverse isolation at a desired frequency.
7. Print or save the data to a disk.



## Reflection Measurements

---

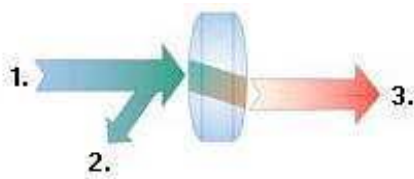
Reflection measurements are an important part of network analysis.

- [What are Reflection Measurements?](#)
- [Why Make Reflection Measurements?](#)
- [Expressing Reflected Waves](#)
  - [Return Loss](#)
  - [VSWR](#)
  - [Reflection Coefficient](#)
  - [Impedance](#)
  - [Summary of Expressions](#)

[See other Tutorials](#)

### What are Reflection Measurements?

To understand reflection measurements, it is helpful to think of traveling waves along a transmission line in terms of a lightwave analogy. We can imagine incident light striking some optical component like a clear lens. Some of the light is reflected off the surface of the lens, but most of the light continues on through the lens. If the lens had mirrored surfaces, then most of the light would be reflected and little or none would be transmitted.



1. Incident    2. Reflected    3. Transmitted

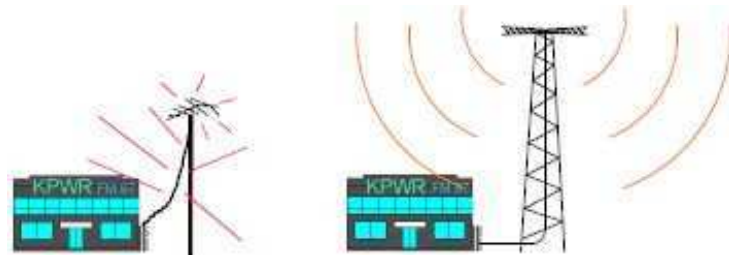
With RF energy, reflections occur when the impedance of two mated devices are not the same. A reflection measurement is the ratio of the reflected signal to the incident signal. Network analyzers measure the incident wave with the R (for reference) channel and the reflected wave with the A channel. Therefore, reflection is often shown as the ratio of A over R (A/R). We can completely quantify the reflection characteristics of our device under test (DUT) with the amplitude and phase information available at both the A and R channel. In S-parameter terminology, S11 is a reflection measurement of port1 of the device (the input port); S22 is a reflection measurement of the port 2 (the output port)

## Why Make Reflection Measurements?

One reason we make reflection measurements to assure efficient transfer of RF power. We do this because:

1. RF energy is not cheap. When energy is reflected, that means less energy is transmitted to where it is intended to go.
2. If the reflected energy is large, it can damage components, like amplifiers.

For example, in the following graphic, the radio station on the left is not operating at peak efficiency. The amplifier impedance is not the same as the transmission line, and the transmission line impedance is not the same as the antenna. Both of these conditions cause high reflected power. This condition results in less transmitted power, and the high reflected power could damage the amplifier.



The radio station on the right installed properly "matched" transmission line and antenna. Very little of the transmitted signal is reflected, resulting in increased broadcast power, more listeners, more advertising revenue, and more profit. The amplifier, transmission, and antenna all need to be measured to ensure that reflected power is minimized.

## Expressing Reflected Waves

After making a reflection measurement, the reflection data can be expressed in a number of ways, depending on what you are trying to learn. The various expressions are all calculated by the analyzer from the same reflection measurement data. Each method of expressing reflection data can be graphically displayed in one or more formats. For more information, see display formats.

### Return Loss

The easiest way to convey reflection data is return loss. Return loss is expressed in dB, and is a scalar (amplitude only) quantity. Return loss can be thought of as the absolute value or dB that the reflected signal is below the incident signal. Return loss varies between infinity for a perfect impedance match and 0 dB for an open or short circuit, or a lossless reactance. For example, using the log magnitude format on the analyzer, the measured reflection value on the screen may be -18dB. The minus sign is ignored when expressing return loss, so the component is said to have 18dB of return loss.

### VSWR

Two waves traveling in opposite directions on the same transmission line cause a "standing wave". This condition can be measured in terms of the voltage standing wave ratio (VSWR or SWR for short). VSWR is defined as the maximum reflected voltage over the minimum reflected voltage at a given frequency. VSWR is a scalar (amplitude only) quantity. VSWR varies between one for a perfect match, and infinity for an open or short circuit or lossless reactance.

### Reflection Coefficient

Another way of expressing reflection measurements is reflection coefficient gamma (  $\Gamma$  ). Gamma includes both



magnitude and phase.

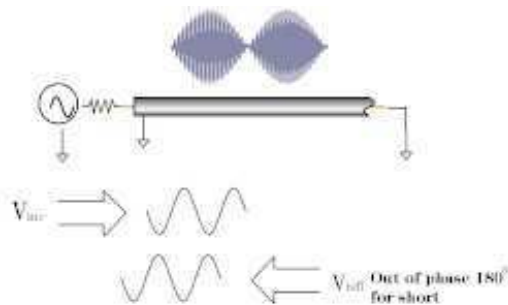
The magnitude portion of gamma is called rho (  $\rho$  ). Reflection coefficient is the ratio of the reflected signal voltage to the incident signal voltage. The range of possible values for  $\rho$  is between zero and one. A transmission line terminated in its characteristic impedance will have all energy transferred to the load; zero energy will be reflected and  $\rho = 0$ . When a transmission line terminated in a short or open circuit, all energy is reflected and  $\rho = 1$ . The value of rho is unitless.

Now for the phase information. At high frequencies, where the wavelength of the signal is smaller than the length of conductors, reflections are best thought of as waves moving in the opposite direction of the incident waves. The incident and reflected waves combine to produce a single "standing" wave with voltage that varies with position along the transmission line.

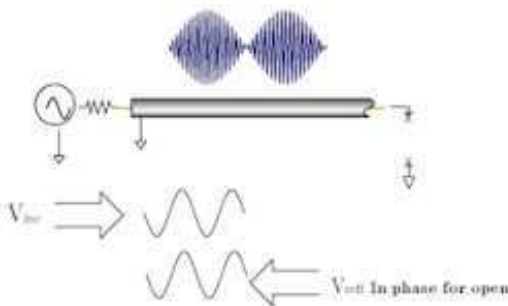
When a transmission line is terminated in its characteristic impedance ( $Z_0$ ) there is no reflected signal. All of the incident signal is transferred to the load, as shown in the following graphic. There is energy flowing in one direction along the transmission line.



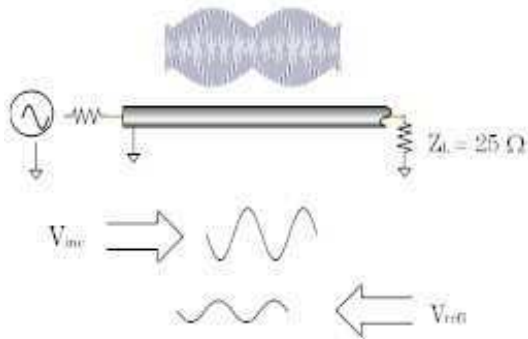
When a transmission line is terminated in a short circuit termination, all of the energy is reflected back to the source. The reflected wave is equal in magnitude to the incident wave ( $\rho = 1$ ). The voltage across any short circuit is zero volts. Therefore, the voltage of the reflected wave will be 180 degrees out of phase with the incident wave, canceling the voltage at the load.



When a transmission line is terminated in an open circuit termination, all of the energy is reflected back to the source. The reflected wave is equal in magnitude to the incident wave ( $\rho = 1$ ). However, no current can flow in an open circuit. Therefore, the voltage of the reflected wave will be in phase with the voltage of the incident wave.



When a transmission line is terminated in a 25 ohm resistor, some but not all of the incident energy will be absorbed, and some will be reflected back towards the source. The reflected wave will have an amplitude 1/3 that of the incident wave and the voltage of the two waves will be out of phase by 180 degrees at the load. The phase relationship will change as a function of distance along the transmission line from the load. The valleys of the standing wave pattern will no longer go to zero, and the peaks will be less than that of the open / short circuit.

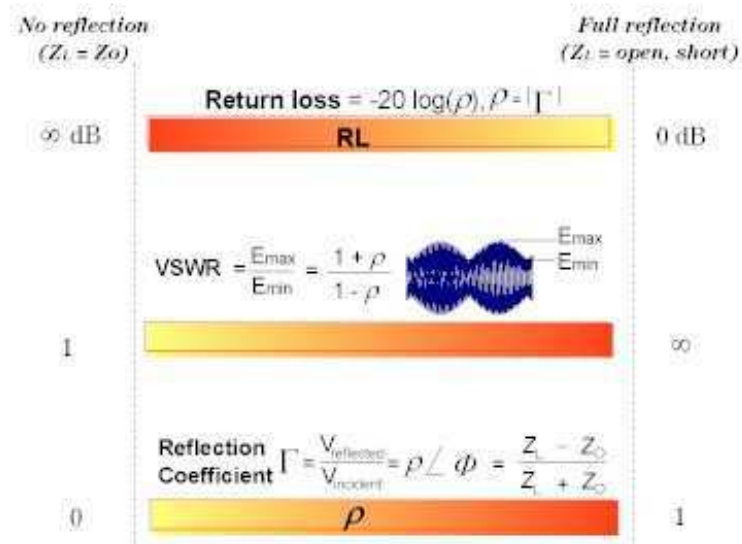


For more information, see [Phase Measurements](#).

## Impedance

Impedance is another way of expressing reflection data. For more information on Impedance, see [Smith Charts](#).

## Summary of the Expressions of Reflection Measurements:



## Select the PNA Model

Please select the PNA model of the specifications you would like to see.

Click the Internet link to be sure you are viewing the most recent version.

Embedded in PNA Help	Internet
<a href="#">E8361A/C</a>	<a href="#">E8361-90007</a>
<a href="#">E8362A/B/C, E8363A/B/C, E8364A/B/C</a>	<a href="#">E8364-90031</a>
<a href="#">N5230A/C 2-Port</a>	<a href="#">N5230-90016</a>
<a href="#">N5230A/C 4-Port</a>	<a href="#">N5230-90020</a>
<a href="#">N5241A and N5242A</a> (PNA-X)	<a href="#">N5242-90007</a>
<a href="#">N5244A and N5245A</a> (PNA-X)	<a href="#">N5245-90008</a>
<a href="#">N5264A (Measurement Receiver)</a>	<a href="#">N5264-90003</a>
<a href="#">Option H29</a> (N5244A / N5245A)	

See the [equations that are used to generate uncertainty curves](#).

The specifications for the following discontinued models are available ONLY on the Internet:

- [E8356A, E8357A, E8358A](#)
- [E8801A, E8802A, E8803A](#)
- [N3381A, N3382A, N3383A](#)
- [E8362A, E8363A, E8364A](#)

## Glossary

[A](#)   [B](#)   [C](#)   [D](#)   [E](#)   [F](#)   [G](#)   [H](#)   [I](#)   [J](#)   [K](#)   [L](#)   [M](#)  
[N](#)   [O](#)   [P](#)   [Q](#)   [R](#)   [S](#)   [T](#)   [U](#)   [V](#)   [W](#)   [X](#)   [Y](#)   [Z](#)

---

**12-Term Error Correction** See [Error Correction, 12-Term](#).

**1-Port Device** A device with a single connector or path to the device's circuitry. Examples include an oscillator and a load.

**2-Port Calibration, Full** See [Error Correction, 12-Term](#).

**2-Port Device** A device with two connectors or other paths to the device's circuitry. Examples include filters, SAW devices, attenuators, matching pads, and amplifiers.

**3-Term Error Correction** See [Error Correction, 3-Term](#).

---

## A

**Active Channel** The highlighted channel affected by front panel functions.

**Active Function Readout** The area of a display screen where the active function and its state are displayed. The active function is the one that was completed by the last key selection or remote programming command.

**Active Marker** The marker on a trace that can be repositioned either by front panel controls or by programming commands.

**Active Trace** A trace that is being swept (updated) with incoming signal information.

**ADC** Analog to Digital Converter

**Address** The identification (represented by a name, label, or number) for a register, location in storage, or any other data source or destination. Examples are the location of a station in a communications network, or a device on the GP-IB.

**ADM** Add-Drop Multiplexer

**Admittance (Y)** The inverse of an impedance (i.e. the ratio of current to voltage). Complex admittances take the form  $Y = G + jB(t)$ .

**ALC** Automatic Level Control. See [Automatic Gain Control](#).

**AM** Amplitude Modulation

**AM Group Delay** A technique for the measurement of group delay through a device which utilizes an amplitude modulated (AM) source. Note: The actual delay of the modulation envelope is measured directly with an external scalar detector. Devices that distort the amplitude of a signal cannot be measured. These include amplifiers with automatic gain control (AGC) and devices subject to saturation or power limiting.

**Amplitude Modulation** The process, or result of the process, of varying the amplitude of a carrier signal. The resulting modulated carrier contains information that can be recovered by demodulation. See also [Modulation](#).

**Analog** The general class of devices or circuits in which the output varies as a continuous function of the input.

**Annotation** The labeling of specific information on the display (such as frequency or power).

**ANSI** American National Standards Institute: A national membership organization (open to manufacturers, organizations, users, and communications carriers) that approves standards, accredits standards development groups and certificate programs, and represents and coordinates US interests in non-treaty and non-government

standards bodies.

**Aperture** The frequency span of the network analyzer used for calculating group delay. The narrower the aperture, the finer the resolution of the group delay variations, but noise is reduced by increasing the aperture.

**Array** A set of numbers or characters that represents any given function.

**ASCII** American Standard Code for Information Interchange

**Attenuation** Denotes a reduction in signal amplitude. The difference between transmitted and received power due to loss through equipment, lines, or other transmission devices; usually expressed in decibels.

**Attenuator** An RF or microwave device used to reduce the power level of a signal by precise, incremental amounts over its entire frequency range.

**Automatic Calibration System** AutoCal: Feature offered on Rohde&Schwarz network analyzers.

**Automatic Gain Control (AGC)** A circuit used in amplifiers and other active devices to keep its RF power level constant as other parameters change, such as frequency. Synonym: Automatic Leveling Control (ALC)

**Autoscale** An analyzer feature that evaluates waveforms and adjusts controls to stable and enhance the display.

**AUX** Auxiliary; refers to rear-panel input connector.

**Averaging** A noise reduction technique that computes each data point based on consecutive sweeps and weighted by a user-specified averaging factor. Each new sweep is averaged into the trace until the total number of sweeps is equal to the averaging factor.

---

## B

**B/R** The ratio of data sampled at B to the data sampled at R.

**Band Pass** A range of frequencies that are passed through a device, such as a filter. Frequencies not within the band pass are limited or attenuated. See also [Cutoff Frequency](#).

**Bandwidth (BW)** The difference between the frequencies of a continuous frequency band within which performance of a device falls within specifications.

**Bandwidth Limit** The condition prevailing when the system bandwidth is exceeded and signal distortion occurs beyond specifications.

**Bandwidth Selectivity** A measure of a filter's ability to resolve signals unequal in amplitude. It is the ratio of the 60 dB bandwidth to the 3 dB bandwidth for a given resolution filter (IF). Bandwidth selectivity tells us how steep the filter skirts are. Bandwidth selectivity is sometimes called shape factor.

**Binary** A method of representing numbers in a scale of two (on or off, high-level or low-level, one or zero). A compact, fast format used to transfer information to and from the analyzer.

**BMP** Bit-Mapped

**Brightness** See [Color Brightness](#).

**Broadband Device** A device that operates over a very wide frequency range and exhibits only small variations in response over that range.

**Buffer** A storage device used when transmitting information to compensate for a difference in the rate of flow of information between two devices.

**Burst Carrier** A carrier that is periodically turned off and on. A burst carrier may or may not be modulated.

**BUS** Basic Utility System

**Bus** One or more conductors used as a path to deliver transmitted information from any of several sources to any of several destinations.

**BW** Bandwidth

**Byte** Eight bits of data representing one character processed as a unit.

---

## C

**CAD** Computer Aided Design

**CAE** Computer Aided Engineering

**Calibration** In HP instrumentation, the process of periodically (usually annually) verifying an instrument is performing to specifications. A calibration certificate is awarded after verification.

In network analyzers, the process of removing systematic errors from measurements. See [Error Correction](#).

**Calibration Kit** Hardware and software required to perform error correction on a network analyzer for a specific measurement and/or test set.

**Calibration, 2-Port** See [Error Correction, 12-Term](#).

**Calibration, Blackburn** Calibrations of transmission path with corrected source match involving 15 calibration terms. Synonym: 15-term error correction

**Calibration, Frequency Response** The simplest error correction procedure to perform, but only corrects for a few of the twelve possible systematic error terms. Frequency response corrections can be made for reflection measurements, transmission measurements, and isolation measurements.

**Calibration, Interpolation** A user selectable network analyzer feature that calculates (interpolates) new error correction terms from existing terms when there is a change in network analyzer parameters, such as IF bandwidth, power, or sweep time. The resulting error correction is not as accurate as completing a full 2-port calibration.

**Calibration, Port Extension** See [Port Extension](#).

**Calibration, Reference Plane** See [Reference Plane](#).

**Calibration, Set Z** Sets the system impedance, usually 50 or 75 ohms.

**Calibration, SOLT** A calibration using four known standards: Short-Open-Load-Through. Also known as a full two-port calibration and 12-term error correction. See also [Error Correction](#).

**Calibration, TRL and LRM** A calibration used in environments where the DUT cannot be connected directly to the network analyzer ports, (MMIC, microstrip, beam-lead diodes etc.). Thru-Reflect-Line (TRL) and M (Match) standards are fabricated and used because known high-quality standards are not readily available. The requirements for characterizing these standards are less stringent, but the calibration is not as accurate as the traditional full two-port calibration using S-O-L-T standards. The terms are used interchangeably (TRL, LRL, LRM etc.) but they all refer to the same basic calibration method.

**Characteristic Impedance** The impedance looking into the end of an infinitely long lossless transmission line.

**Color Brightness** A measure of the intensity (brightness) of a color.

**Command** A set of instructions that are translated into instrument actions. The actions are usually made up of individual steps that together can execute an operation.

**Continuous Sweep Mode** The analyzer condition where traces are automatically updated each time trigger conditions are met.

**Controller** A device capable of specifying the talker and listeners for an information transfer. An external computer connected to an instrument to control its operation.

**Corrected** Measurements made after performing error correction.

**Coupler** See [Directional Coupler](#).

**CPU** Central Processing Unit

**Crosstalk** The occurrence of a signal at one port of a device being affected by a signal in any other path. Isolation is the measurement of crosstalk.

**Cursor** An electronically generated pointer that moves across the display to manipulate controls.

**Cutoff Frequency** In filters, the frequency at which attenuation is 3dB below the band pass signal level, known as the 3dB points.

**CW** Continuous wave: A single frequency (rather than a swept frequency).

---

## D

**DAC** Digital to Analog Converter

**dB** Decibel: a relative unit of measure. The ratio in dB is given by:  $10 \log_{10} (P_1/P_2)$  where  $P_1$  and  $P_2$  are the measured powers. The dB is preferred instead of arithmetic ratios or percentages because when components are connected in series, their effect on power, expressed in dB, may be arithmetically added and subtracted. For example, if a 3dB attenuator is connected to a 10dB amplifier, the net gain of the two components is (-3dB + 10dB = +7dB).

**dBm** Absolute unit of measure in decibels: 0dBm = 1 mW. The conventions of the dB (adding and subtracting) continue to apply.

**DBMS** Database Management System

**DC** Direct Current

**Default** A known set of conditions used in the absence of user-defined conditions.

**Delay** See [Group Delay](#).

**Demodulation** The process of recovering from a modulated carrier, information in the form of a signal having essentially the same characteristics as the original modulating signal. Recovery of the modulating signal accomplished by signal detection.

**Detection** The process of demodulating signal carriers. There are two basic ways of providing signal detection in network analyzers: Diode detectors (used in broadband applications) and heterodyning, (used in narrowband applications).

**Detector, Diode** A device used to convert a RF signal to a proportional DC level. If the signal is amplitude modulated, the diode strips the RF carrier signal from the modulation. Many sources used with scalar analyzers are amplitude modulated with a 27.778 kHz signal and then detected in the network analyzer. Phase information on the signal carrier is lost in diode detection.

**Deviation from Linear Phase** Linear phase refers to the nature of the phase shift of a signal through a device. The phase is linear if a plot of phase shift versus frequency is a straight line using linear scales. Deviation from linear phase causes signal distortion.

**Digital** Pertaining to the class of devices or circuits in which the output varies in discrete steps.

**Digital Demodulation** Describes a technique of extracting the information used to modulate a signal. Digital signal processing algorithms are used on the signal after it has been converted from an analog to a digital form (digitized).

**Dimension** To specify the size of an array. The number of array rows or columns.

**Directivity** In a 3-port directional coupler, the ratio of the power present at the auxiliary port when the signal is traveling in the forward direction to the power present at the auxiliary port when the same signal is traveling in the reverse direction.

**Directional Coupler** A 3-port device typically used for separately sampling the backward (reflected) wave in a

transmission line.

**Disk** A circular, magnetic storage medium.

**Display** Noun: See [Screen](#).

Verb: To show annotation and measurement data on the display.

**Display Detector Mode** The manner in which analog, video information is processed prior to being digitized and stored in memory.

**Display Dynamic Accuracy** The amplitude uncertainty, usually in dB, over the display dynamic range.

**Display Dynamic Range** The amplitude range, in dB, over which the display dynamic accuracy applies.

**Display Formats** Graphical formats for displaying measurement data. These include single channel, overlay (multiple traces on one graticule), split (each trace on separate graticules).

**Display Modes** The ways in which measurement data can be presented graphically. On a network analyzer, the choices are Cartesian/rectilinear (XY plot with log or linear magnitude, phase, group delay, SWR, real and imaginary, and dBV, dBmV and dBuV), polar (magnitude and angle), magnitude and phase, and Smith chart. Not all display modes are available on all network analyzers. In addition, displays can present this information in various combinations of traces. Common modes are dual, (the ability to display more than one trace, usually over the same frequency range), and alternate, (the ability to display more than one trace, each with different frequency range and type).

**Display Phase Dynamic Accuracy** The phase measurement uncertainty, usually in degrees, for measurements whose units are in degrees.

**Display Points** The total number of measurement points made in a single measurement. The points can be in units of frequency, power, or time. The number of points often dictates measurement speed, resolution, and aperture.

**Display Trace Noise, Magnitude** The amplitude uncertainty of the trace, in dB, due to random noise in the test system.

**Display Trace Noise, Phase** The phase uncertainty of the trace, in degrees, due to random noise in the test system.

**Display Type** The type of display screen built into the analyzer. Data can be displayed as a raster drawing (a computer-like dot map) or as a vector drawing (lines drawn on the display). Color and display standard can also be specified as monochrome (single color), or color (two or more colors). The format standard may also be specified, such as VGA or SVGA, for IBM-compatible personal computers.

**Distortion** Deterioration of a signal's quality due to the nonlinear characteristics of a device or system transfer function. Distortion is measured as a combination of the changes in amplitude, frequency and phase of signal at the output of a device or system as compared to the signal at the input.

**Drift** The slow change in signal frequency.

**DSP** Digital Signal Processing

**DUT** Device Under Test

**DVM** Digital Volt Meter

**Dynamic Range** In a receiver, the range of signal levels, from minimum to maximum, that can be reliably measured simultaneously. Dynamic range allows small signals to be measured in the presence of large signals. Source power and receiver compression usually limits the maximum boundary to dynamic range. Receiver residual responses and noise floor usually limit the minimum power boundary.

---



# E

**ECal** See [Electronic Calibration](#).

**Electrical Delay** A simulated variable length of lossless transmission line, added to or subtracted from a receiver input, to compensate for interconnecting cables. The firmware equivalent of mechanical or analog "line stretchers" in other network analyzers.

**Electronic Calibration (ECal)** A calibration system for electronic calibration of RF and microwave vector network analyzers. The electronic calibration system creates a twelve-term, two-port error model and then provides a confidence check of the calibration. The Ecal system consists of a repeatable, variable-impedance, solid-state calibration standard and a mainframe control unit which interfaces with the 8510, 8720 series, and the 8753 network analyzers or a USB module which interfaces with the PNA series network analyzers.

**EMC** Electro-Magnetic Compatibility

**EMI** Electro-Magnetic Interference: Unintentional interfering signals generated within or external to electronic equipment. Typical sources could be power-line transients, noise from switching-type power supplies and/or spurious radiation from oscillators. EMI is suppressed with power-line filtering, shielding, etc.

**Engage** To activate a function.

**Enter** The process of inputting information.

**EPROM** Electronically Programmable, Read-Only Memory

**Error Correction** In network analyzers, a process that removes or reduces systematic (repeatable) measurement errors by measuring known standards from a calibration kit. Synonym: measurement calibration

**Error Correction, 3-Term** Used to remove systematic measurement errors on a device with one port, such as a load.

**Error Correction, 12-Term** Correction for a two port device using six parameters:

Directivity

Source match

Load match

Reflection frequency response

Transmission frequency response

Isolation

To completely characterize a two-port device, these six parameters must be characterized in the forward and reverse directions, making a total of 12 terms. The user usually has the option of omitting isolation from the correction process. Synonym: Full two-port error correction

**Error Correction, 1-Port** Corrects a test set for port 1 or port 2 directivity, frequency response, and source match errors. The process requires three known standard terminations, for example, open, short, and load.

**Error Message** A message on a display that indicates an error condition. Missing or failed hardware, improper user operation, or other conditions that require additional attention can cause an error condition. Generally, the requested action or operation cannot be completed until the condition is resolved.

**ESD** Electro Static Discharge

**Ethernet** A network that adheres to the IEEE 802.3 Local Area Network standard.

**Ethernet address** A hexadecimal number which is used to identify a machine on a network. Each analyzer is assigned a unique Ethernet address at the factory and it is stored in the analyzer's ROM.

**External trigger signal** A TTL signal that is input to an analyzer and initiates a measurement sweep or similar event, making the measurements synchronous with the external triggering source.

## F

**Filter** A passive device that allows some frequencies to pass and attenuates others, depending on the type and specifications. A high-pass filter passes frequencies above the cutoff frequency, a low-pass filter passes frequencies below the cutoff frequency, and a band-pass filter passes frequencies between two specific frequencies.

**Firmware** An assembly made up of hardware and instruction code. The hardware and instruction code is integrated and forms a functional set that cannot be altered during normal operation. The instruction code, permanently installed in the circuitry of the instrument, is classified as ROM (read only memory). The firmware determines the operating characteristics of the instrument or equipment.

**Flatness** The amplitude and phase response of a device under test (DUT), a signal source, a receiver, or a combination of these. See also [Frequency Response](#).

**FM** Frequency Modulation

**Frequency** The number of periodic oscillations, vibrations, or waves per unit of time, usually expressed in cycles per second, or Hertz (Hz).

**Frequency Accuracy** The uncertainty with which the frequency of a signal or spectral component is indicated, either in an absolute sense or relative to another signal or spectral component. Absolute and relative frequency accuracies are specified independently.

**Frequency Range** The range of frequencies over which a device or instrument performance is specified.

**Frequency Resolution** The ability of a network analyzer to measure device characteristics at closely spaced frequencies and display them separately. Resolution of equal amplitude responses is determined by IF bandwidth. Resolution of unequal amplitude responses is determined by IF bandwidth and bandwidth selectivity.

**Frequency Response** The peak-to-peak variation in the displayed amplitude response over a specified center frequency range. Frequency response is typically specified in terms of dB, relative to the value midway between the extremes.

**Frequency Span** The magnitude of the displayed frequency component. Span is represented by the horizontal axis of the display. Generally, frequency span is given as the total span across the full display. Some analyzers represent frequency span (scan width) as a per-division value.

**Frequency Stability** The ability of a frequency component to remain unchanged in frequency or amplitude over short and long-term periods of time. Stability refers to an oscillator's ability to remain fixed at a particular frequency over time.

**Front Panel Key** Keys that are located on the front panel of an instrument. The key labels identify the function the key activities. Numeric keys and step keys are two examples of front panel keys.

**Full 2-Port Calibration** See [Error Correction, 12-Term](#).

**Function** The action or purpose that a specific item is intended to perform or serve. The network analyzer contains functions that can be executed via front panel key selections, or through programming commands. The characteristics of these functions are determined by the firmware in the instrument. In some cases, a DLP (downloadable program) execution of a function allows you to execute the function from front panel key selections.

**Fundamental Frequency** In any waveform, the lowest frequency component; all other components are harmonics. A pure sinusoid has only one component, the fundamental.

---

## G

**Gb** Gigabit

**GB** Gigabyte

**GHz** Gigahertz

**GIF** Graphics Interchange Format - Standard graphic format to store bitmapped graphics files.

**Giga** Prefix for one billion.

**GP I/O** General Purpose Input / Output; a connector usually on the back of an instrument that allows communication with other test equipment, external test sets, switches, and computers that enable the instrument to be triggered or to trigger external equipment. An example is a foot switch that continues or cycles a measurement, allowing the operator to use both hands on the test hardware.

**GPIB** General Purpose Interface Bus - IEEE 488 bus is interconnect bus and protocol, allows linking of instruments and computer.

**Graticule (or Grid)** Enclosed area where waveform is displayed on instrument. Tick marks, on frame or axis, are a scaling aid for making visual measurements.

**Group Delay** A measure of the transit time of a signal through a DUT versus frequency. Group delay can be calculated by differentiating the DUT's insertion-phase response with respect to frequency. See also [AM Group Delay](#) and [Deviation from Linear Phase](#).

**GUI** Graphical User Interface

---

## H

**Hardcopy** Paper copy of data.

**Hardkey** A front-panel key, which engages a single analyzer function or presents a single menu of softkeys.

**Horizontal Reference** See [Reference Level](#).

**Horizontal Resolution** The analyzer's ability to take closely spaced horizontal data points over the full sweep.

**Host Computer** A computer or device on a network that provides end users with services such as computation and database access and that usually performs network control functions.

**Host Name** A unique name that is used to identify each host machine on a network. The host name is directly linked to, and can usually be used in place of, the IP address. The user or the system administrator usually creates the host name.

**HP** Hewlett-Packard Company

**HPGL** Hewlett-Packard Graphics Language

**HP-IB** Hewlett-Packard Interface Bus. A parallel interface that allows "daisy chaining" of more than one device to a port on a computer or instrument. Interface protocol is defined in IEEE 488.2; equivalent to the industry standard GPIB.

**HTTP** HyperText Transfer Protocol: Used to carry World Wide Web (WWW) traffic.

**Hue** The dimension of color referred to a scale of perceptions ranging from red through yellow, green, and blue, and back to red. A particular gradation of color, tint, shade.

---

## I

**I/O** Input/Output

**I/O Path** Input/Output Path

**IEEE** Institute of Electrical and Electronic Engineers

**IF** Intermediate Frequency: the frequency at which a signal is processed after mixing.

**Impedance** The ratio of voltage to current at a port of a circuit, expressed in ohms.

**Initialize** The process that assigns information locations to a disk to prepare the magnetic media to accept files.

**Input** A path intended for putting a signal into an instrument.

Most network analyzers have either 3 (labeled A, B, and R) or 4 inputs (labeled A, B, R1, and R2). Inputs are not the same as channels.

**Input Attenuator** An attenuator between the input connector and the first mixer of a spectrum analyzer (also called an RF attenuator). The input attenuator is used to adjust the signal level incident to the first mixer, and to prevent gain compression due to high-level or broadband signals. It is also used to set the dynamic range by controlling the degree of internally-generated distortion. For some analyzers, changing the input attenuator settings changes the vertical position of the signal on the display, which then changes the reference level accordingly. In Agilent microprocessor-controlled analyzers, the IF gain is changed to compensate for changes in input attenuator settings. Because of this, the signals remain stationary on the display, and the reference level is not changed.

**Insertion Loss** The difference between the power measured before and after the insertion of a device. The attenuation between the input and output of a device.

**Intensity** Brightness; emitting or reflecting light; luminosity.

**Interface** A connection that allows a common communication link between two or more instruments.

**Intermodulation Distortion** Undesired frequency components resulting from the interaction of two or more spectral components passing through a device having nonlinear behavior, such as a mixer or an amplifier. The undesired components are related to the fundamental components by sums and differences of the fundamentals and various harmonics. The algorithm is:  $f_1 \pm f_2$ ,  $2f_1 \pm f_2$ ,  $2f_2 \pm f_1$ ,  $3f_1 \pm 2f_2$ , and so on.

**Internet** The connection of two or more distinct networks. Often a gateway or router is used to make the connection.

**Interpolate** To determine a value of a signal between two adjacent points by a procedure or algorithm.

**IP** Internet Protocol

**IP Address** Internet protocol address: a unique number that is assigned to each device which is to be connected to a TCP/IP network. Before using an analyzer on a network, your network administrator will need to assign an IP address. An IP address consists of a 32-bit value presented in decimal dot notation: 4 octets (bytes) separated by a dot.

**ISDN** Integrated Services Digital Network: A standard digital service capability that features one or more circuit-switched communication channels capable of carrying digital voice, data, or image signals, a packet-switched channel for out-of-band signaling and control. In addition, ISDN provides a collection of standard and optional features that support information productivity for the user, providing higher-speed Internet access than analog systems.

**ISO** International Standards Organization

**Isolation** A specification or measure of the immunity that one signal has to being affected by another adjacent signal. The occurrence is known as crosstalk.

**Isolator** An RF device used for providing isolation between paths and components. Made from a 3-port circulator, the third port being terminated in a 50ohm load.

---

## J

---

## K

**Kilo** Prefix for one thousand.

**KB** Kilobyte

**Kb/s** Kilobytes per second

---

## L

**LAN** Local Area Network

**LANS** Local Area Network System

**LCD** Liquid Crystal Display

**LED** Light Emitting Diode

**LIF** Logical Interchange Format (used for older HP disk drives/computers)

**Limit Lines** Lines input by the user that overlay the analyzer's measurement data to allow automatic detection of data that is out of the acceptable range. Pass/Fail annotation, audio alarms, or electronic output can be triggered to notify the operator or on-line computer program of the over-limit condition.

**Limit-Line File** The user-memory file that contains the limit-line table entries.

**Limit-Line Table** The line segments of a limit line are stored in the limit-line table. The table can be recalled to edit the line segments, then restored in the limit-line file.

**Linear Device** A device in which the output is continuously proportional to the input.

**LO** Local Oscillator. In a superheterodyne system, the LO is mixed with the received signal to produce a sum or difference equal to the intermediate frequency (IF) of the receiver.

**LO Feedthrough** The response that in a superheterodyne system when the first local oscillator frequency is equal to the first IF.

**Load** A one port microwave device used to terminate a path in its characteristic impedance.

**Load Match** A measure of how close the device's terminating load impedance is to the ideal transmission line impedance. Match is usually measured as return loss or standing wave ratio (SWR) of the load.

**Local Lock Out** A condition or command that prevents analyzer front-panel entries (and disables the Local key).

**Local Operation** To operate manually from the front panel.

**Log** Logarithm

**Log Display** The display mode in which vertical deflection is a logarithmic function of the input signal amplitude. Log display is also called logarithmic display. The display calibration is set by selecting the value of the reference level position and scale factor in dB per division.

**LRM** Line-Reflect-Match. See [Calibration, TRL, and LRM](#).

---

## M

**Magnitude** The amplitude of a signal measured in its characteristic impedance without regard to phase. See also [Scalar](#).

**Marker** A graphical symbol along a display trace that is annotated with measurement characteristics of that specific data point.

**Marker Functions** Mathematical or statistical computation on the data of one or more markers to provide the operator more information. For example, the marker delta function calculates and displays the difference between two markers.

**Maximum Input Level** The maximum signal power that may be safely applied to the input of an analyzer. The maximum input level is typically 1 W (+30 dBm) for Agilent spectrum analyzers.

**MB** Megabyte

**Measurement Uncertainty** The quantified amount of error in a measurement situation. Calibrations are intended to reduce the amount of uncertainty. The following are sources of measurement errors that lead to uncertainty:

- Systematic errors (imperfections in calibration standards, connectors, cables, and instrumentation)
- Random errors (noise, connector repeatability)
- Drift (source and instrumentation)

**Mega** Prefix for one million.

**Memory** A storage medium, device, or recording medium into which data can be stored and held until some later time, and from which the entire original data may be retrieved.

**Memory Card** A small memory device shaped like a credit card that can store data or programs.

**Menu** The analyzer functions that appear on the display and are selected by pressing front panel keys. These selections may invoke a series of other related functions that establish groups called menus.

**MHz** Megahertz

**milli** Prefix for one-thousandth.

**Modem** Modulator/Demodulator

**Modulation** The process, or the result of the process, of varying a characteristic of a carrier signal with an information-bearing signal, causing the carrier to contain the information. See [AM](#) and [FM](#).

**Monitor** Any external display.

**Monochrome** Having only one color (chromaticity).

**ms** Millisecond

**mW** Milliwatt: one thousandth of a watt

**Multisync** A type of monitor that can synchronize its horizontal sweep to various frequencies within a specified range.

---

## N

**Narrowband** In network analysis, the frequency resolution of the analyzer's receiver that is sufficiently narrow to resolve the magnitude and phase characteristics of narrowband devices. The reduced receiver bandwidth usually decreases the noise floor of the receiver, providing more measurement amplitude range.

**Narrowband Device** A device whose transfer characteristics are intended to operate over a very narrow frequency range and are designed to provide well-defined amplitude responses in that range, such as a band pass filter.

**Network Analysis** The characterization of a device, circuit, or system derived by comparing a signal input going into the device to a signal or signals coming out from the device.

**NIST** National Institute of Standards and Technology

**Nit** The unit of luminance (photometric brightness) equal to one candela per square meter.

**Noise** Random variations of unwanted or disturbing energy in a communications system from man-made and natural sources that affects or distorts the information carried by the signal. See also [Signal-to-Noise Ratio](#).

**Noise Figure (F):** For a two-port device, a measure of how the noise generated inside the device degrades the signal-to-noise ratio of a signal passing through the device at 290 degrees, usually expressed in dB.

**Noise Floor** The analyzer's internal displayed noise. The noise level often limits how small a signal magnitude can be measured. In network analysis, noise floor is measured with the test ports terminated in loads, full two-port error correction, 10 Hz IF bandwidth, maximum test port power, and no averaging during the test.

**Non-Insertable Devices** In measurement calibration, a device that cannot be substituted for a [Zero-Length Through Path](#). It has the same type and sex connectors on each port, or a different type of connector on each port.

**Nonvolatile Memory** Memory data that is retained in the absence of an ac power source. This memory is typically retained with a battery. Refer also to battery-backed RAM.

**Normalize** To subtract one trace from another to eliminate calibration data errors or to obtain relative information.

---

## O

**Offset** To move or set off a determined amount. Used in instruments for offsetting frequencies, limits, delay, loss, impedance, etc.

**Output Attenuation** The ability to attenuate the signal, the source, in order to control its power level.

---

## P

**PC** Personal Computer

**PDF** Portable Document Format (used on the Web)

**Parser, Command** Reads program messages from the input queue of a device in the order they were received from the controller. The parser determines what actions the analyzer should take. One of the most important functions of the command parser is to determine the position of a program message in the analyzer SCPI command tree. When the command parser is reset, the next element it receives is expected to arise from the base of the analyzer command tree.

**Peak Search** A function on an analyzer that searches for the largest response and places a marker on it.

**Phase** The fractional part of a cycle through which an oscillation has advanced, measured from an arbitrary starting point; usually measured in radians or degrees. In network analysis, the phase response of the device under test is the change in phase as a function of frequency between the input stimulus and the measured response.

**Port** The physical input or output connection of an instrument or device.

**Port Extension** Redefining the reference plane to other than that established at calibration. A new reference plane is defined in seconds of delay from the test set port.

**Positive Peak** The maximum, instantaneous value of an incoming signal.

**Postscript (.ps files)** Stores bitmapped graphics files in an encapsulated format for direct use by postscript printers.

**Power, Max Input** The upper limit to input power for which the specifications apply. Some specifications may have different levels of maximum inputs. For example, compression power maximum is usually higher than the harmonic distortion maximum.

**Power, Safe Input** The input power, usually in dBm, allowed without damaging the instrument.

**Preset** A pre-defined instrument state (that also runs an analyzer self-test). The action of pushing the Preset key.

**Protocol** A set of conventions that specify how information will be formatted and transmitted on a network, and how machines on a network will communicate.

---



## Q

**Q or Q Factor** The ratio of energy stored to energy lost in a resonant circuit. High Q indicates a sharp resonance response over frequency.

**Query** Any analyzer programming command having the distinct function of returning a response. These commands may end with a question mark (?). Queried commands return information to the computer.

---

## R

**$r + jx$**  Expression for complex impedance, where r represents the resistive portion and x represents the reactive portion.

**R Channel** Reference Channel

**RAM** Random Access Memory, or read-write memory: A storage area allowing access to any of its storage locations. Data can be written to or retrieved from RAM, but data storage is only temporary. When the power is removed, the information disappears. User-generated information appearing on a display is RAM data.

**ROM** Read Only Memory: A storage area that can be read only; it cannot be written to or altered by the user. In instruments, the storage area that contains the "brains" or operational programming; the firmware.

**Receiver** A circuit or system designed for the reception and/or measurement of signals in a specified frequency spectrum.

**Receiver Dynamic Range** See [Dynamic Range](#).

**Reference Level** An instrument function that allows the user to set the amplitude value at the reference position. On network analyzers, the reference position is also selectable. On some spectrum analyzers, the reference position is fixed at the top of the display.

**Reference Plane** The electrical location at which a network analyzer assumes the system connectors and fixturing ends and the DUT begins. The reference plane is set by using calibration standards with known electrical length. The closer the reference plane is to the device under test (DUT), the better the characterization of the device because of the elimination of test system uncertainties.

**Reference Receiver** In a network analyzer, the receiver that measures signals as they come out of the source, before they are incident on the test port and DUT. Typically, these signals are used to compare with the signal at the Test Port Receiver, to determine the affect that the DUT has on the signal. In a 2-port network analyzer, these are typically named 'R1' (port 1) and 'R2' (port 2). [See a block diagram](#) of the receivers in your PNA.

**Reflection** The phenomenon in which a traveling wave strikes a discontinuity and returns to the original medium.

**Reflection Coefficient** The ratio of the reflected voltage to the incident voltage into a transmission line or circuit. If a transmission line is terminated in its characteristic impedance, the reflection coefficient is zero. If the line is shorted or open the coefficient is 1. See also [Return Loss](#) and [SWR](#).

**Reflection Measurements** Measurements that characterize the input and /or output behavior of the device under test (DUT). Measured as the ratio of the reflected signal to the incident signal as a function of frequency. Parameters are called return loss, reflection coefficient, impedance, and standing wave ratio (SWR), all as a function of frequency. See also [S-Parameters](#).

**Remote** A mode of operation where another device (or computer) controls an instrument via the HP-IB. In this mode, the instrument front panel keys are disabled. Front panel operation is called local operation.

**Remote Programming** The automatic operation of an instrument by a computer, usually through a HP-IB, LAN, or RS-232 link.

**Resolution** The ability of a receiver to resolve two signals.



**Resolution Bandwidth** The ability of a spectrum analyzer to display adjacent responses discretely (Hertz, Hertz decibel down). This term is used to identify the width of the resolution bandwidth filter of a spectrum analyzer at some level below the minimum insertion loss point (maximum deflection' point on the display). Typically, it is the 3 dB resolution bandwidth that is specified, but in some cases the 6 dB resolution bandwidth is specified.

**Return Loss** The amount of dB that the reflected signal is below the incident signal. If zero signal is reflected, the impedance of the device is equal to the characteristic impedance of the transmission system, and return loss is infinite. If the entire incident signal is reflected, the return loss is zero. See also [S-Parameters](#), [Reflection Coefficient](#), and [SWR](#).

**Reverse Measurement** The measurement of a device from output to input.

**RF** Radio Frequency (from approximately 50 kHz to approximately 3 GHz). Usually referred to whenever a signal is radiated through the air.

**ROM** Read Only Memory

---

## S

**S/N** Signal-to-Noise Ratio

**Sampler** An electronic component that captures the signal level and phase across a known impedance at a uniform rate. In Network Analyzers, this sampling rate must be sufficiently high and precisely timed to make accurate measurements. Network analyzers typically have three or four samplers or mixers.

**Sampler Bounce** The leakage or crosstalk between a network analyzer's samplers. Delay in this crosstalk caused by leakage transmission propagation, give the interference its "bounce" appearance. Sampler bounce causes an increase in the noise level of the affected channel, reducing the sensitivity of the analyzer.

**Saturation** The degree of color purity, on a scale from white to pure color.

**Scalar** A quantity that has magnitude but no phase. A network analyzer capable of measuring only magnitude.

**Scale Factor** The display vertical axis calibration in terms of units per division.

**SCPI** Standard Commands for Programmable Instruments

**Screen** The physical surface of the CRT or flat panel upon which the measurement results, setup information, softkey definitions, and other instrument communication is presented.

**Self-Test** A group of tests performed at power-up (or at preset) that verify proper instrument operation.

**Sensitivity** The minimum input signal required to produce a specified output signal having a specified signal-to-noise ratio, or other specified criteria.

On a spectrum analyzer, the level of the smallest sinusoid that can be observed, usually under optimized conditions of minimum resolution bandwidth, 0 dB input attenuation, and minimum video bandwidth.

The normalized change in YIG component's center frequency resulting from a change in tuning coil current, specified in MHz/mA.

**Serial Prefix** The five-character prefix that begins an instrument serial number; used to represent versions of firmware or hardware changes that have occurred.

**Server** A device that is configured to provide a service to other devices on a network, such as shared access to a file system or printer.

**Signal-to-Noise Ratio** SNR: The ratio of the amplitude of the desired signal to the amplitude of noise signals, usually expressed in dB and in terms of peak values for impulse noise and root-mean-square values for random noise.

**Single Sweep Mode** The spectrum analyzer sweeps once when trigger conditions are met. Each sweep is initiated

by pressing an appropriate front panel key, or by sending a programming command.

**Small Signal Gain Compression** A situation when the input signal's measured amplitude is less than its actual level due to overloading of the network analyzer's input mixer; the analyzer is operating nonlinearly. For broadband analyzer detectors, a signal other than the one under test can put the analyzer into this gain compressed mode, thereby making even lower level signals appear at a lower level than actual. The broadband mode measures all the power incident to the analyzer, not just the signals at the frequency of interest.

**Smith Chart** A graphical mapping of the complex reflection coefficient into normalized complex impedance. Circles on the chart represent constant resistance and radiating lines orthogonal to the circles represent constant reactance. The center of the chart represents the characteristic impedance of the transmission system. Any point on the chart defines a single complex impedance. A line on the chart represents changing impedance over frequency.

**SOLT** Short-Open-Load-Through calibration. See also [Calibration, SOLT](#).

**Source** A device that supplies signal power; a sweep oscillator or synthesized sweeper.

**Source Amplitude Accuracy** The amplitude uncertainty, in dB, of the source power readout.

**Source Amplitude Flatness** The amplitude flatness, in dB, of the source power over the frequency range specified.

**Source Frequency Resolution** The smallest unit of frequency which can be set and/or measured, in Hz.

**Source Frequency Time Base Accuracy** A measure of the analyzer's frequency stability measured in parts per million (ppm, or 1 part in 10E6). For example, a stability of  $\pm 5.0$  ppm means that an analyzer will measure 1 MHz to an accuracy of  $\pm 5 \times 10^{-6} \times 10^6 \text{ Hz} = \pm 5 \text{ Hz}$ .

**Source Frequency Time Base Stability** A measure of the analyzer's time base accuracy over time and temperature. Typically the time base accuracy will be specified for 1 year. A typical temperature frequency stability is  $\pm 10$  ppm for  $25^\circ\text{C} \pm 50^\circ\text{C}$ .

**Source Harmonics** The level of harmonics generated by the analyzer's signal source, in dBc from the fundamental.

**Source Match** A measure of how close the signal source impedance is to the ideal transmission line impedance of the test system. Match is usually measured as return loss or standing wave ratio (SWR) of the source.

**Span** The stop frequency minus the start frequency. The span setting determines the horizontal-axis scale of the analyzer display.

**Span Accuracy** The uncertainty of the indicated frequency separation of any two signals on the display.

**S-Parameters (Scattering Parameters)** A convention used to characterize the way a device modifies signal flow using a network analyzer. A two port device has four S-parameters: forward transmission (S21), reverse transmission (S12), forward reflection (S11), and reverse reflection (S22).

**Stop/Start Frequency** Terms used in association with the stop and start points of the frequency measurement range. Together they determine the span of the measurement range.

**Storage States** The number of settings, programs, traces, and other parameters available to be saved, cataloged, and recalled at any one time.

**Storage, Disk** An internal or external digital storage disk for saving test data, instrument settings, IBASIC programs, and other measurement parameters. Storage formats include MS-DOS (R) and HPs standard LIF with binary, PCX, HP-GL, or ASCII data formats.

**Structural Return Loss** Poor return loss in cable due to a periodic fault such as a periodic dent caused by dropping the cable spool or by the cable pulling process during manufacture.

**Supplemental Characteristics** Typical but non-warranted performance parameters, denoted as "typical", "nominal" or "approximate".

**Sweep** The ability of the source to provide a specified signal level over a specified frequency range in a specified time period. Also see [Sweep Mode](#) and [Sweep Type](#).

In data processing mode, a series of consecutive data point measurements, taken over a sequence of stimulus values.

**Sweep Mode** The way in which a sweep is initiated or selected, e.g., single, continuous, alternate, or chopped.

**Sweep Type** The method of sweeping the source, e.g., linear, log, or frequency step.

**Sweeper** A signal source that outputs a signal that varies continuously in frequency.

**SWR** Standing Wave Ratio, calculated as  $(1 + \Gamma) / (1 - \Gamma)$  where  $\Gamma$  is the reflection coefficient.

**Sync** Synchronization, or Synchronized

**Syntax** The grammar rules that specify how commands must be structured for an operating system, programming language, or applications.

**System Dynamic Range** The difference between the maximum receiver input level and the receiver's noise floor. System dynamic range applies to transmission measurements only, since reflection measurements are limited by directivity.

---

## T

**T/R** See [Transmission/Reflection](#).

**Termination** A load connected to a transmission line or other device.

**Test Limit** The acceptable result levels for any given measurement.

**Test Port** See [Port](#).

**Test Port Receiver** In a network analyzer, the receiver directly behind the test ports, used to measure the signal as it is reflected off, or transmitted through, the DUT. This signal is typically compared with the signal at the [Reference Receiver](#) to determine how the DUT affects a signal. In a 2-port network analyzer, these are typically named 'A' (port 1) and 'B' (port 2). [See a block diagram](#) of the receivers in your PNA.

**Test Set** The arrangement of hardware (switches, couplers, connectors and cables) that connect a test device input and output to the network analyzer's source and receiver to make s-parameter measurements.

**Third Order Intercept** TOI: The power input to a non-linear device that would cause third order distortion at the same power level. TOI is a measurement to determine the distortion characteristics of a mixer or receiver. The higher the value, the more immune the receiver to internal distortion.

**Thru** Through line: A calibration standard. See [Calibration, SOLT](#).

**Tint** A shade of color; hue.

**Toggle** To switch states, usually to change a function from on to off, or off to on.

**TOM** Thru-Open-Match: A Rohde&Schwarz term to describe a calibration method.

**Trace** A series of data points containing frequency and response information. The series of data points is often called an array. The number of traces is specific to the instrument.

**Tracking** The ability of the analyzer's receiver to tune to the source frequency over the measurement frequency range. Poor tracking results in amplitude and phase errors due to the receiver IF circuits attenuating and delaying the device under test output.

**Transfer Function** The ratio of the output signal to the stimulus signal, both as a function of frequency.

**Transmission** See [Transmission Measurements](#).

**Transmission Intermodulation Spurious** A measure of the capability of the transmitter to inhibit the generation of intermodulation distortion products. Intermodulation spurious is sometimes called intermodulation attenuation.

**Transmission Measurements** The characterization of the transfer function of a device, that is, the ratio of the output signal to the incident signal. Most common measurements include gain, insertion loss, transmission coefficient, insertion phase, and group delay, all measured over frequency. See also [S-Parameters](#).

**Transmission/Reflection (T/R)** Refers to the suite of measurements made by a scalar or vector network analyzer to characterize a device's behavior over frequency. See also [S-Parameters](#).

**Transparent** Something that is not visible to the user. Usually a procedure that occurs without the user's initiation or knowledge.

**Trigger** A signal that causes the instrument to make a measurement. The user can select several options for triggering, such as manual, continuous, or external (for synchronizing measurements to an external source).

**TRL** Through-Reflect-Line. See [Calibration, TRL and LRM](#).

**TTL** Transistor-Transistor Logic

**Two-Port Error Correction** See [Error Correction, 12-Term](#).

---

## U

**Uncorrected** Measurements made without performing error correction.

**Uncoupled Channels** Stimulus or receiver settings allowed to be set independently for each channel.

**UNI** User-Network Interface: The point at which users connect to the network.

**Units** Dimensions on the measured quantities. Units usually refer to amplitude quantities because they can be changed. In analyzers with microprocessors, available units are dBm (dB relative to 1 mW dissipated in the nominal input impedance), dBmV (dB relative to 1 mV), dBW (dB relative to 1 W), V (volts), W (watts).

---

## V

**Variable** A symbol, the value of which changes either from one iteration of a program to the next, or within each iteration of a program.

**Vector** A quantity that has both magnitude and phase.

A network analyzer capable of measuring both magnitude and phase.

**VEE** Visual Engineering Environment (Agilent software product)

**Velocity Factor** A numerical value related the speed of energy through transmission lines with different dielectrics (.66 for polyethylene). Used in making time domain measurements.

**Vertical Resolution** The degree to which an instrument can differentiate amplitude between two signals.

**Video** An electrical signal containing timing, intensity, and often color information that, when displayed, gives a visual image.

**Video Bandwidth** In spectrum analyzers, the cutoff frequency (3 dB point) of an adjustable low-pass filter in the video circuit. When the video bandwidth is equal to or less than the resolution bandwidth, the video circuit cannot fully respond to the more rapid fluctuations of the output of the envelope detector. The result is a smoothing of the trace, or a reduction in the peak-to-peak excursion, of broadband signals such as noise and pulsed RF when viewed in broadband mode. The degree of averaging or smoothing is a function of the ratio of the video bandwidth to the resolution bandwidth.

**Video Filter** In spectrum analyzers, a post-detection, low-pass filter that determines the bandwidth of the video

amplifier. It is used to average or smooth a trace. Refer also to [Video Bandwidth](#).

**VNA** Vector Network Analyzer

---

## W

**Waveform** A representation of a signal plotting amplitude versus time.

**Wireless** A term that refers to a broad range of technologies that provide mobile communications for home or office, and "in-building wireless" for extended mobility around the work area, campus, or business complex. It is also used to mean "cellular" for in-or out-of-building mobility services.

**WWW** World Wide Web

---

## X

---

## Y

---

## Z

**Zero-Length Through Path** In a measurement calibration, when the two test cables mate together directly without using adapters or a thru-line. See also [Non-Insertable Devices](#).