Programmer Manual

Tektronix

WCA330 & WCA380
3GHz & 8GHz Wireless Communication Analyzer

070-A794-50

This document applies to firmware version 3.2 and above.

www.tektronix.com

Tektronix Japan, Ltd., 5–9–31 Kitashinagawa, Shinagawa–ku, Tokyo 141–0001 Japan
Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; or c) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

Part I Using Script

Getting Started on Script

Built-in Functions for the Analyzer

Part II Using Command

Getting Started on Command

Syntax and Commands

Status and Events

Programming Examples

Appendices

Glossary and Index

List of Figures

List of Tables

Preface

This programmer manual is for the WCA330 and WCA380 Wireless Communication Analyzers. This manual provides information on using the script language and operating the instrument over the GPIB and TCP/IP interfaces.

This manual consists of two parts:

Part I: Using Script

- *Getting Started on Script* describes the introduction to the script language and several scripting examples.
- *Built-in Functions for the Analyzer* describes each function in the analyzer function set.

Part II: Using Command

- *Getting Started on Command* describes how to connect and set up the analyzer for remote operation.
- *Syntax and Commands* defines the command syntax and processing conventions and describes each command in the analyzer command set.
- *Status and Events* explains the status information and event messages reported by the analyzer.
- *Programming Examples* shows some example analyzer programs.
- *Appendices* contain tables of character charts and factory initialization settings.

Related Manuals

Other documentation for the analyzer includes:

- The *WCA330 and WCA380 User Manual* (Tektronix part number 070-A792-XX) describes the installation and operation of the instrument.
- The *WCA330 and WCA380 Service Manual* (Tektronix part number 070-A795-XX) provides information for maintaining and servicing the instrument.

Difference between WCA330 and WCA380

WCA330 and WCA380 have the same functions except for their measurement frequency ranges:

WCA330 . . . . DC to 3 GHz
WCA380 . . . . DC to 8 GHz

Unless otherwise noted, descriptions in this manual apply to both.

Part I Using Script

Getting Started on Script

Getting Started on Script

The WCA330 and WCA380 Wireless Communication Analyzer supports the standard script language. You can write script programs using the analyzer and a keyboard to set front panel controls, take measurements, and control external GPIB devices.

For the GPIB commands, refer to *Part II: Using Command.*

Overview

Script programs are the text files that describe specific procedures. You have to connect a keyboard to the analyzer before you can write/edit a script program on the analyzer. After writing/editing a program on the analyzer or a PC, you can execute the program only by loading it into the analyzer memory. Use a floppy disk or LAN to transfer programs from a PC. The programs that you have written can be saved on the hard disk of the analyzer or PC. Compilation or linkage tasks required for high-level language are not necessary for script.

If you only execute a program, neither keyboard nor mouse is necessary; you can execute it with the front panel operation.

For connections of the mouse, keyboard, and LAN, refer to the *User Manual.*

On-line Help

For the general scripting information, refer to the following on-line help on the analyzer:

- C:\Program Files\SONY Tektronix\WCA\opScript\Htrplang.hlp

To access the on-line help, use Windows Explorer to open the file with a mouse and keyboard connected to the analyzer. For accessing Windows 98, refer to the *User Manual.*

This manual provides information on the extended functionality of the WCA330 and WCA380 analyzers. This chapter shows some basic examples. The next chapter describes the syntax and usage for each function.

General Rules

Here are several general rules for scripting.

- A program begins with “procedure main” and ends with “endProc”.

```
procedure main
 ...
 (program)
 ...
endProc
```

- The variable declaration begins with “var” and ends with “endVar”.

```
var
 ...
 (variable declaration)
 ...
endVar
```

  A variable is called “local variable” when defined in the procedure, or “global variable” when defined out of the main procedure.

- You can use a combination of upper and lower case letters for describing functions and parameters. Upper and lower case letters are not distinguished.

  These three instructions have the same meaning.

```
DrvSetValue('Span', '1M')

DRVSETVALUE('SPAN', '1M')

drvsetvalue('span', '1m')
```

  The analyzer automatically interprets the unit “m” or “M” as 10^{-3} for amplitude or time and as 10^{+6} for frequency.

- Comments are expressed with a double-slash (//) for one line and a pair of curved braces ({ }) for two or more lines.

```
// comment

{ comment 1
  comment 2
    ...
  comment N }
```

Preparation

Before you edit your script program, do the following steps. For details on operating the analyzer, refer to the *User Manual*.

Suppose that View A and B show the measurement display controlled by the script and you edit the script in View D.

1. Connect a mouse and keyboard to your analyzer.
2. Set the analyzer to four view display so that you can edit your program on one view (View D in this example) and display data on the other views (View A and B in this example).
3. Press CONFIG:**VIEW** (front panel key) → **View D** (side key) and select **Script**.
4. Press VIEW:**D** key. See Figure 1–1 as an example.

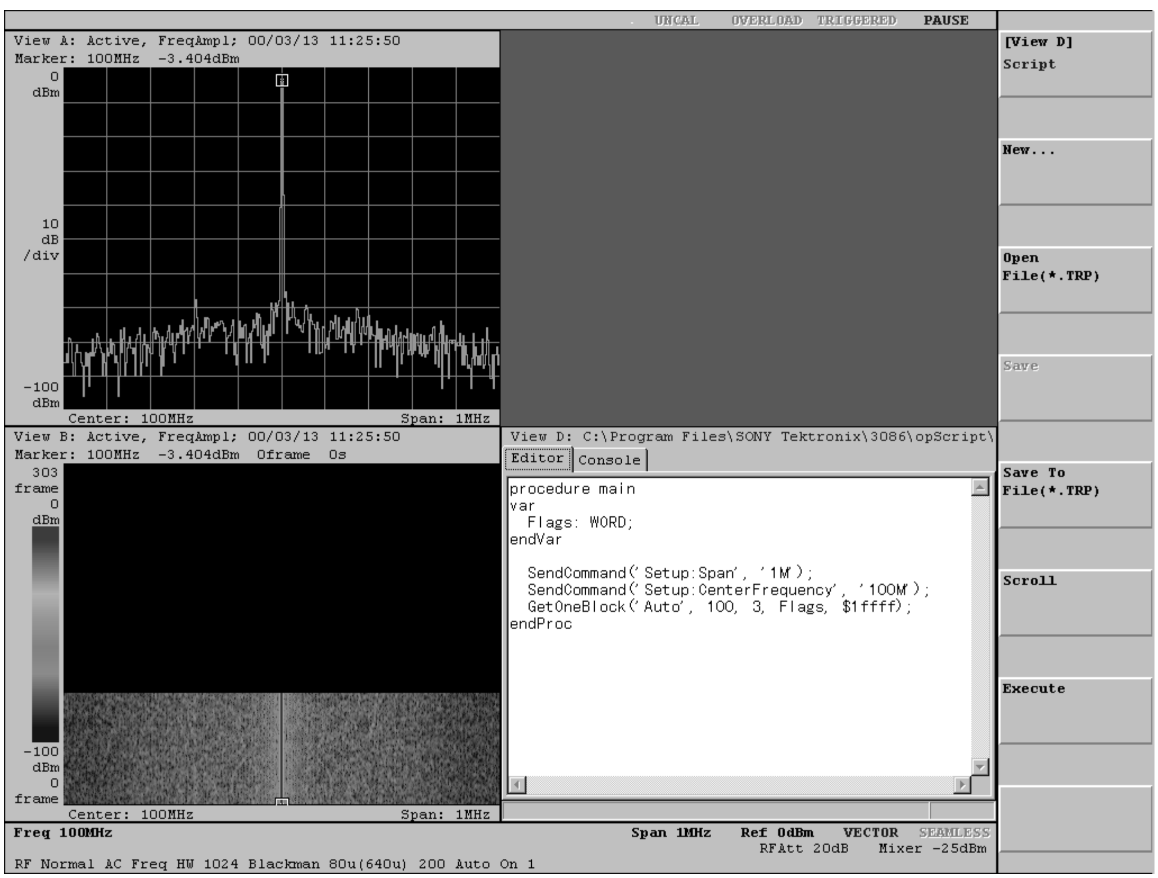


Figure 1–1: Editing a script program on View D

Editing a Script

The side keys used to edit script programs are shown in Figure 1–2.

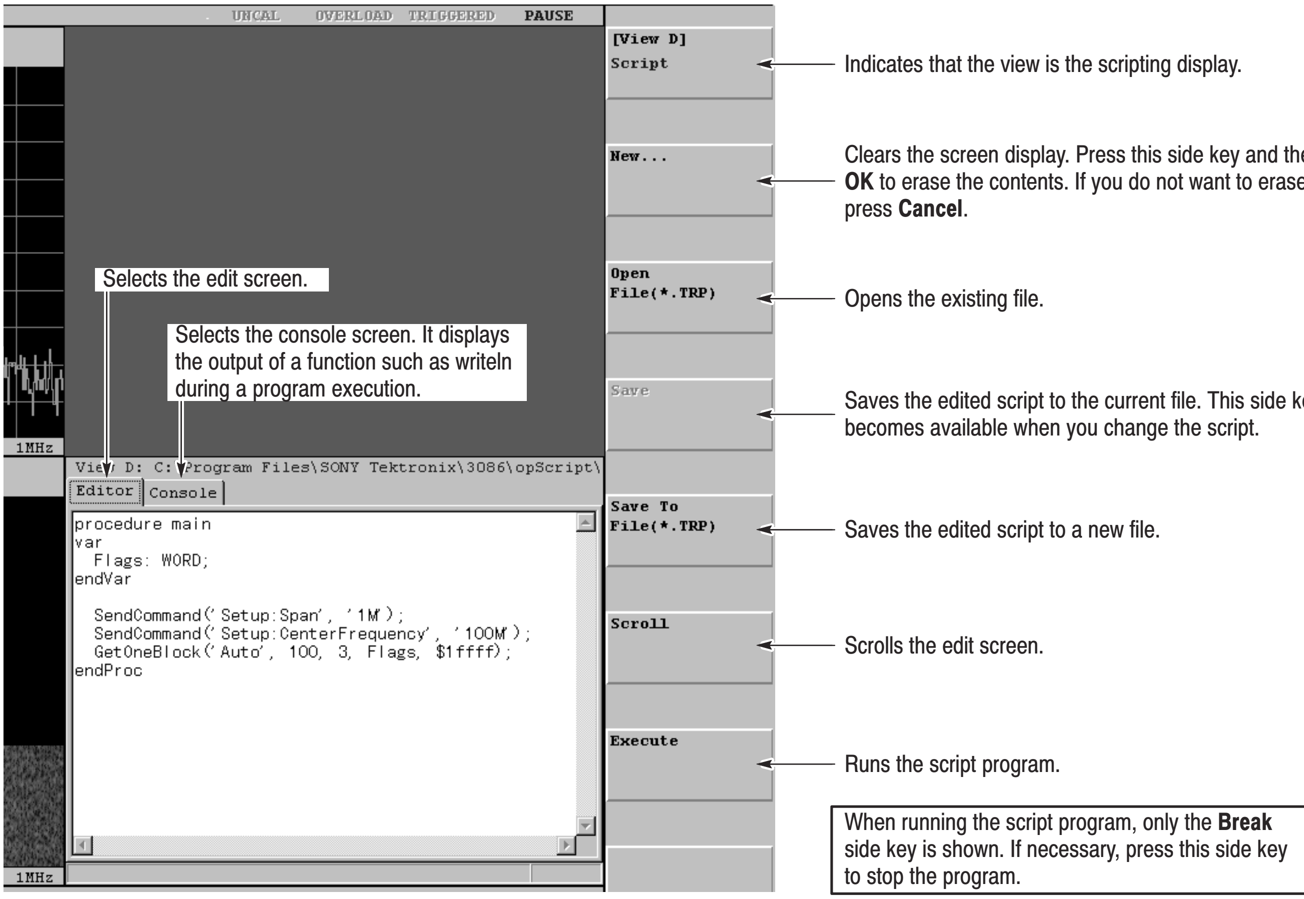


Figure 1–2: Script editing keys

When you press the **Scroll** side key while editing the script, the screen switches from the four-view display to a full display. Press the **Execute** side key to return to the four-view display.

Executing a Script

Press the **Execute** side key in the Script menu (on View D by default) to run the script program that you have made or loaded from a file.

Press the **Break** side key, or the **ROLL**, **BLOCK**, or **CLEAR** key on the front panel if you want to interrupt the program.

Simple Scripting Examples

The example program listed below sets the span to 1 MHz and center frequency to 100 MHz, then acquires 100 frames of data.
(Refer to C:\Program Files\SONY Tektronix\WCA\opScript\First.trp.)

```
procedure main
var
   Flags: WORD;
endVar

SendCommand('Setup:Span', '1M');
   // Set the span to 1 MHz.
SendCommand('Setup:CenterFrequency', '100M');
   // Set the center frequency to 100 MHz.
GetOneBlock('Auto', 100, 3, Flags, $08000);
   // Acquire data with the auto trigger mode and one block = 100 frames.
endProc
```

***NOTE**. For details on the functions, refer to* Built-in Functions for the Analyzer *starting on page 2–1.*

If you want to change hardware settings temporally and then acquire data, use DrvSetValue to manipulate hardware directly. At the end of the task, use DrvSaveSetup and DrvRestoreSetup to restore the hardware settings to those made by the analyzer SETUP program .

```
procedure main
   var
      Flags: WORD;
   endVar

   DrvSaveSetup;          // Save the current settings.

   DrvSetValue('Span', '1M');
      // Set the span to 1 MHz.
   DrvSetValue('CenterFrequency', '100M');
      // Set the center frequency to 100 MHz.
   GetOneBlock('Auto', 100, 3, Flags, $08000);
      { Acquire data with the auto trigger mode and
      one block = 100 frames.}

   DrvRestoreSetup;       // Restore the current settings.
EndProc
```

Performing a Set of Tasks Using a Single Side Key

The following examples show how to display some side keys on a menu, allowing you to perform the specific process when pressing each side key.

Adding Side Keys on a Menu

Every procedure written in script is available after you press the **Execute** side key and until the **Break** side key is pressed. This example adds two side keys to a menu in the main procedure.
(Refer to C:\Program Files\SONY Tektronix\WCA\opScript\Menu.trp.)

```
procedure main
   MenuAppendButton(0, 'Test1|TButton,Test1');
      // Add the side key Test1.
   MenuAppendButton(0, 'Test2|TButton,Test2');
      // Add the side key Test2.
endProc
```

Writing Procedures for a Side Key

This example defines the Test 1 and Test 2 procedures called when pressing the Test1 and Test2 side keys, respectively, added in the above procedure.

```
procedure Test1              // Test1 program
   writeln('Test1');
endProc

procedure Test2              // Test2 program
   writeln('Test2');
endProc
```

Acquiring Data in a Procedure

The GetOneBlock and GetMultiBlock functions allow you to acquire data in a procedure. You must disable the **ROLL** and **BLOCK** keys during a call to the Test1 procedure. In the main procedure, change“Tside key, Test1” to “Tside key, Sleep: Test1”.

```
MenuAppendButton(0, 'Test1|TButton,Sleep:Test1');
    { Add the Test1 side key. Pressing Test1 disables the ROLL and
    BLOCK keys and then executes the procedure Test1.}

procedure Test1
    var
        Flags: WORD;
    endVar

    SendCommand('Setup:Span', '1M');
        // Set the span to 1 MHz.
    SendCommand('Setup:CenterFrequency', '100M');
        // Set the center frequency to 100 MHz.
    GetOneBlock('Auto', 100, 3, Flags, $08000);
        { Acquire data with the auto trigger mode and
        one block = 100 frames. }
EndProc
```

Acquiring Data without Changing Settings Made by SETUP

If you want to change hardware settings temporally and then acquire data, use DrvSetValue to manipulate hardware directly. At the end of the task, use DrvSaveSetup and DrvRestoreSetup to restore the hardware settings to those made by the analyzer SETUP program.

```
procedure Test1
var
    Flags: WORD;
endVar

    DrvSaveSetup;            // Save the current settings.

    DrvSetValue('Span', '1M');
        // Set the span to 1 MHz.
    DrvSetValue('CenterFrequency', '100M');
        // Set the center frequency to 100 MHz.
    GetOneBlock('Auto', 100, 3, Flags, $08000);
        { Acquire data with the auto trigger mode and
        one block = 100 frames. }

    DrvRestoreSetup;         // Restore the current settings.
endProc
```

Performing Tasks Each Time Data Is Acquired

If you want to have a script perform some tasks each time data is acquired using the **ROLL** or **BLOCK** key, write an ActiveEvent procedure. You can also write a MarkerEvent or DregEvent procedure called each time a marker position or register content changes, respectively. The following script calculates RMS whenever a block of data is acquired.
(Refer to C:\Program Files\SONY Tektronix\WCA\opScript\Event.trp.)

```
procedure main
   SendCommand('Setup:Span', '1M');
      // Set the span to 1 MHz.
   SendCommand('Setup:CenterFrequency', '100M');
      // Set the center frequency to 100 MHz.
   SendCommand('Setup:BlockSize', '20');
      // Set the block size to 20 frames.
   SendCommand('Setup:TriggerCount', 'Off');
      // Turn the trigger count function off.

   SendCommand('View1:Source', 'D1');
      // Set the View 1 source to the data register D1.
   SendCommand('View1:Trace2:Source', 'Active');
      // Set the View 1 Trace 2 source to the currently acquired data.
endProc

procedure ActiveEvent
var
   evt: Longint;
endVar

    evt := MsgGetEventFromMessage('Active');
   if (evt = EVT_BLOCK) or (evt = EVT_CHANGEBLOCK) then
      SendCommand('Util3:RMS', 'Active,19,0,D1');
      // Calculate RMS after acquiring one block.
   endIf
endProc
```

NOTE*. You must not call the GetOneBlock and GetMultiBlock functions in the ActiveEvent, MarkerEvent, and DregEvent procedures.*

Manipulating Registers

You can have a script draw graphs on the Waveform view by setting data registers. The following procedure sets a value for each point.
The DregSetDataWithMask procedure allows you to set all the values at a time. (Refer to C:\Program Files\SONY Tektronix\WCA\opScript\Dreg.trp.)

```
procedure main
   var
      n: Integer;
   endVar

// Set the header for the data register D1.
   DregSetHeader('D1', 'XNum', '20');
      // Set the number of data on the horizontal axis to 20.
   DregSetHeader('D1', 'XScale', '20');
      // Set the horizontal axis scale to 20.
   DregSetHeader('D1', 'XStart', '0');
      // Set the minimum value on the horizontal axis to 0.
   DregSetHeader('D1', 'YScale', '100');
      // Set the vertical axis scale to 100.
   DregSetHeader('D1', 'YStart', '-100');
      // Set the minimum value on the vertical axis to –100.
   DregSetHeader('D1', 'ZNum', '1');
      // Set the number of frames to 1.

// Set the data value and mask information for each data point.
    for n := 0 to 19 do
      DregSetPointWithMask('D1', n, -100 + n * 5, 0);
   endFor
endProc
```

Displaying a Grid

The output of functions such as `writeln` is displayed in a window like Notepad. You can display a grid in a part of the window and show calculation results in the cells. (Refer to C:\Program Files\SONY Tektronix\WCA\opScript\Grid.trp.)

```
procedure main
var
   c: Integer;
   r: Integer;
endVar

   SetColCount(4);          // Set the number of columns to 4.
   SetFixedCols(1);         // Set the number of fixed columns to 1.
   SetRowCount(10);         // Set the number of rows to 10.
   SetFixedRows(1);         // Set the number of fixed rows to 1.

// Display "(the column number) – (the row number)" in each cell.
   for c := 0 to GetColCount-1 do
      for r := 0 to GetRowCount-1 do
         SetCell(c, r, IntToStr(c) + '-' + IntToStr(r));
      endFor
   endFor

   AdjustColWidths;         // Adjust the column width.
   SetGridAlign('Top', 80);
      // Specify that 80 % of the upper area of the view is the grid area.

   MenuAppendButton(0, 'Row|TScroll,RowScroll');
      // Add the side key Row. The process is defined in RowScroll.
endProc
```

```
// Moves the currently selected row.
procedure RowScroll
    var
        param: String;
    endVar

    param := GetCommandString;
    if param = '1' then     // When turning the knob cw.
        if GetRow < GetRowCount-1 then
            SetRow(GetRow + 1);
                // Moves the currently selected row to the next downward.
        endIf
    else                    // When turning the knob ccw.
        if GetFixedRows < GetRow then
            SetRow(GetRow - 1);
                // Moves the currently selected row to the next upward.
        endIf
    endIf

    writeln(GetCell(GetCol, GetRow));
        // Writes the contents of the currently selected row.
endProc
```

Adding Side Keys to Select or Enter Values

There are several types of side keys: command side keys which issue a command when pressed; side keys, such as TToggle and TComboBox types, that allow you to make a selection from options, and side keys, such as TEdit type, that allow you to enter numeric values.

TToggle, TComboBox, and TEdit type side keys issue a command without using parameters when updating a display. The strings returned from the SetQueryString procedure is displayed on a menu. When you use a TToggle or TComboBox type side key to select a option or a TEdit type side key to enter a value, these side keys issue a command using it as a parameter. Use the GetCommandString function to obtain the parameter.
(Refer to C:\Program Files\SONY Tektronix\WCA\opScript\Menu2.trp.)

The next example adds the side key “SG Output” that turns the signal generator on or off and “SG Freq” that sets the output frequency as well as the side keys “Test1” and “Test2” described above.

```
var
   CurOutput: String = 'Off';
   CurFreq: String = '100M';
endVar

procedure main
   MenuAppendButton(0, 'SG Output|TToggle,Output|Off,On');
      { Add the side key “SG Output”. The process is defined in the
      procedure Output. }
   MenuAppendButton(0, 'SG Freq|TEdit,Freq|ENG,0,3G|
      ENG,1,10M|1M|Hz');
      { Add the side key “SG Freq”. The process is defined in the
      procedure Freq. }
   MenuAppendButton(0, 'More|TMenu');
         // Add the More side key.
      MenuAppendButton(1, 'Return');
         // Add the Return side key.
      MenuAppendButton(1, 'Test1|TButton,Test1');
         // Add the Test1 side key.
      MenuAppendButton(1, 'Test2|TButton,Test2');
         // Add the Test2 side key.
endProc
```

```
procedure Output // For the SG Output side key.
var
   param: String;
endVar

   param := GetCommandString;
   if param = '' then
      SetQueryString(CurOutput);
         // Set On or Off for re-displaying the menu.
   else
      CurOutput := param;
      writeln('Output ' + CurOutput);
         // Display “Output On” or “Output Off” on the screen.
   endIf
endProc

procedure Freq // For the SG Freq side key.
var
   param: String;
endVar
    param := GetCommandString;
   if param = '' then
      SetQueryString(CurFreq);
         // Set the current frequency for re-displaying the menu.
   else
      CurFreq := param;
      writeln('Freq ' + CurFreq);
         // Display the current frequency on the screen.
   endIf
endProc
```

Manipulating Text Files

The procedures such as ListLoadFromFile and ListSaveToFile allow you to read or write a text file easily. Use ListLoadFromFile to read the entire text file into a string list and ListGetString to obtain a specified row. For strings with the format of “Name=Value”, you can get the value out of the string associated with the name specified by the ListGetValue function. You can handle up to ten string lists with number 0 to 9 concurrently.
(Refer to C:\Program Files\SONY Tektronix\WCA\opScript\List.trp.)

```
procedure main
var
   cnt : Longint;
   n: Longint;
   s: String;
endvar

   ListLoadFromFile(0, 'c:\Program Files\SONY Tektronix\WCA
      \sys\Dacdef.txt');
      // Get strings from the file to the buffer 0.
   cnt := ListGetCount(0);
      // Get the number of strings.
   writeln('Count = ' + IntToStr(cnt));
      // Write the number of strings.

   if cnt = 0 then
      Return;    // Exit when there is no string.
   endIf

// Write the string characters with the string index.
   writeln;
   for n := 0 to cnt-1 do
      s := ListGetString(0, n);
      writeln('[' + IntToStr(n) + '] ' + s);
   endFor

// Write the string characters after “DacB=”.
   writeln;
   writeln('DacB: ' + ListGetValue(0, 'DacB'));

   ListClear(0);     // Clear the buffer.
EndProc
```

Making a Program for Execution Only

When you complete debugging and no longer need editing, use your script program as a View program for execution only. Programs for execution only display neither the **Execute** nor **Break** side key and always stay in the same state as when you press the **Execute** side key. Also, such programs allow you to switch several script files easily on a menu.

1. Register the program name in the analyzer:

a. For example, suppose that the View program for execution only is named “MyScript”. Create the file *opMyScript.ini* that contains the following two lines:

```
[View]
MyScript=Script,MyScript
```

where

`MyScript` on the left side is the View option that you have named.

`Script` on the right side specifies the execution file *Script.exe*. You must *not* use other names.

`MyScript` on the right side must be added to “Script” in the name of the menu definition files as described below (*ScriptMyScript.txo* and *ScriptMyScript.txm*).

It is recommended that `MyScript` on the left and right sides have the same name for simplicity, although you can use different names.

b. Save the file *opMyScript.ini* in the following directory:

C:\Program Files\SONY Tektronix\WCA\Bin

2. Create the menu definition files:

The MyScript program for execution only has the same execution file *Script.exe* as the editable script program. However, it uses menu definition files called *ScriptMyScript.txo* and *ScriptMyScript.txm* instead of *Script.txo* and *Script.txm*.

a. Copy the file *Script.txo* to *ScriptMyScript.txo*, and save the file *ScriptMyScript.txo* in the following directory:

C:\Program Files\SONY Tektronix\WCA\Bin

b. Describe your script file path(s) in the file *ScriptMyScript.txm* as shown in the following example. The path can be a full path or relative path from C:\Program Files\SONY Tektronix\WCA\Bin.

```
Script:Items=Grid,Dreg,Event,Menu
Script=Menu

Source0=..\opScript\Menu.trp
Source1=..\opScript\Event.trp
Source2=..\opScript\Dreg.trp
Source3=..\opScript\Grid.trp
```

where

`Script:Items` specifies the Script menu items when selecting MyScript. Note that the items are listed in reverse order of Source# described below.

`Script=Menu` means that "Menu" is displayed in the menu item selection field initially.

`Source#` specifies your script file(s) (*.trp).

3. Restart the WCA330 or WCA380 system software.

"MyScript" appears on the View option list in the CONFIG:**VIEW** menu.

Breaking the Program Execution

When you run a program for execution only, the **Break** side key is not displayed. Press **ROLL**, **BLOCK**, or **CLEAR** key on the front panel to stop data acquisition (if acquiring data) and set the returned value of the Terminated function to "True". Check the returned value of this function to exit a long running loop.

Built-in Functions for the Analyzer

Function Groups

This section describes the functions in general categories. Functions for the analyzer are divided into the following groups:

- Command-Related Functions
- GPIB Command-Related Functions
- Data Acquisition-Related Functions
- Data Read-Related Functions
- Register-Related Functions
- Event/Marker-Related Functions
- Numerical Calculation-Related Functions
- Grid-Related Functions
- Menu-Related Functions
- String List-Related Functions
- Hardware-Related Functions
- IQ Calculation-Related Functions
- Miscellaneous Functions

Command-Related

Use these functions to send GPIB-like commands. Refer to page 2–13 for details.

Table 2–1: Command-related functions

Function	Description
GetCommandString	Get external command arguments.
SendCommand	Send a GPIB-like command to other system programs.
SendCommandIf	Send a command with query.
SendQuery	Send a GPIB-like query command to other system programs.
SendQueryPas	Send a GPIB-like query command to other system programs.
SetQueryString	Set returned values for a external command.

GPIB Command-Related

Use these functions to control the external GPIB devices.
Refer to page 2–17 for details.

Table 2–2: GPIB command-related functions

Function	Description
GetGPIBRWTimeOut	Query the read/write time-out.
GetGPIBSpr	Query the serial poll status byte.
InitiateGPIB	Make the external GPIB device(s) available.
SendGPIBCommand	Send the command to the external GPIB device.
SendGPIBQuery	Send the query command to the external GPIB device.
SendGPIBQueryPas	Send the query command to the external GPIB device.
SetGPIBRWTimeOut	Set the read/write time-out.
TerminateGPIB	Make GPIB available from Util8 (Remote) system program.

Data Acquisition-Related

Use these functions to acquire data. Refer to page 2–21 for details.

Table 2–3: Data Acquisition-related functions

Function	Description
GetMultiBlock	Get one block for multiple block acquisition.
GetOneBlock	Acquire one block data.
RebuildWindowTable	Reset a window table when restart other system programs.
RestartMultiBlock	Start acquiring the second and subsequent blocks.
StartMultiBlock	Start multiple block acquisition.
StopMultiBlock	Stop multiple block acquisition.

Data Read-Related

Use these functions to read data. Refer to page 2–25 for details.

Table 2–4: Data read-related functions

Function	Description
GetData	Read acquired data.
GetFlatness	Return flatness correction data.
GetFrameHeader	Read the header of the specified frame.
GetHeader	Read the header for the specified source.
GetHeaderPas	Read the header for the specified source.
GetMinMax	Return the number of the frame where max. or min. locates.
GetPointsWithMask	Get data used for displaying on screen.
GetTimeStamp	Get time stamp for the specified frame.

Register-Related

Use these functions to set data registers. Refer to page 2–31 for details.

Table 2–5: Register-related functions

Function	Description
DregScroll	Shift a data register.
DregSetData	Write data to a data register.
DregSetDataWithMask	Write data and mask information to a data register.
DregSetHeader	Set a header for a data register.
DregSetPoint	Modify the value for a specified data point.
DregSetPointWithMask	Modify the value and mask of a specified data point.

Event/Marker-Related

Use these functions to get event codes and to set or query marker values. Refer to page 2–35 for details.

Table 2–6: Event/Marker-related functions

Function	Description
MsgGetCFromMessage	Get the C marker value.
MsgGetCFromMessagePas	Get the C marker value.
MsgGetDeltaXFromMessage	Get the delta-X marker value.
MsgGetDeltaXFromMessage-Pas	Get the delta-X marker value.
MsgGetDeltaZFromMessage	Get the delta-Z marker value.
MsgGetDeltaZFromMessage-Pas	Get the delta-Z marker value.
MsgGetEventFromMessage	Get an event code from the specified source.
MsgGetXFromMessage	Get the X marker value.
MsgGetXFromMessagePas	Get the X marker value.
MsgGetZFromMessage	Get the Z marker value.
MsgGetZFromMessagePas	Get the Z marker value.
MsgSetDeltaX	Set the delta-X marker value.
MsgSetDeltaZ	Set the delta-Z marker value.
MsgSetMarkerC	Set the C marker value.
MsgSetMarkerX	Set the X marker value.
MsgSetMarkerZ	Set the Z marker value.

Numerical Calculation-Related

Use these functions to perform FFT or spline interpolation.
Refer to page 2–41 for details.

Table 2-7: Numerical calculation-related functions

Function	Description
Fft	Perform FFT or IFFT.
MakeSplineTable	Create auxiliary data for spline interpolation.
QuickSort	Sort data.
Spline	Calculate a vertical coordinate with spline interpolation.

Grid-Related

Use these functions to create a grid on screen. Refer to page 2–43 for details.

Table 2-8: Grid-Related Functions

Function	Description
AdjustColWidths	Adjust the column width for fixed-length columns.
GetCell	Return a string in a cell.
GetCellAttr	Return an attribute of a cell.
GetColCount, GetRowCount	Return the number of columns and rows, respectively.
GetCol, GetRow	Return the current column and row, respectively.
GetFixedCols, GetFixedRows	Return the number of fixed-length columns and rows.
RefreshCells	Update a grid display.
SetCell	Set a string in a cell.
SetCellAttr	Set an attribute of a cell.
SetColCount, SetRowCount	Set the number of columns and rows, respectively.
SetCol, SetRow	Set the current column and row, respectively.
SetFixedCols, SetFixedRows	Set the number of fixed-length columns and rows, respectively.
SetGridAlign	Determines the location and size of the grid area.

Menu-Related

Use these functions to create new menu side keys. Refer to page 2–47 for details.

Table 2-9: Menu-Related Functions

Function	Description
MenuAppendButton	Add a menu side key.
MenuChangeValue	Change the value displayed on the side key.
MenuModify	Update the whole menu display.
MenuSetButton	Activate the specified side key.
MenuSetEnable	Enable or disable the side key.
MenuSetPage	Display the specified page.
MenuSetVisible	Determine whether or not to display the side key.
MenuUpdate	Update the value displayed on a specified side key.

String List-Related

Use these functions to read or write strings on a list.
Refer to page 2–51 for details.

Table 2-10: String List-Related Functions

Function	Description
ListAdd	Add a new string to a list.
ListClear	Delete all the string from a list.
ListDelete	Delete the specified string.
ListGetCount	Return the number of strings in a list.
ListGetString	Return the string at a given index.
ListGetValue	Return only the value for a specified name.
ListInsert	Insert a string at the specified index.
ListLoadFromConsole	Get a text line on the console into a specified list.
ListSaveToConsole	Display a list string on the console.
ListLoadFromFile	Load a text line in a specified file into a specified list.
ListSaveToFile	Save a list string in a specified file.
ListSetString	Change a string at a given index.
ListSetValue	Set only the value for a specified name.

Hardware-Related

Use these functions to set or query the hardware parameters, such as frequency, span, reference level, and trigger. Refer to page 2–55 for details.

Table 2–11: Hardware–Related Functions

Function	Description
DrvGetState	Return data acquisition state.
DrvGetValue	Return the hardware settings.
DrvGetValuePas	Return the hardware settings.
DrvReadiagResult	Return the version of a subsystem.
DrvRestoreSetup	Restore the hardware settings saved by DrvSaveSetup below.
DrvSaveSetup	Save the hardware settings.
DrvSetValue	Set up the hardware.

IQ Calculation-Related

Perform various calculation with I and Q data.
Refer to page 2–59 for details.

Table 2–12: IQ calculation-related functions

Function	Description
GetAmplData	Calculate amplitude from I/Q data.
GetAmplDataWithMean	Calculate amplitude from I/Q data and returns the mean value.
GetFreqData	Calculate frequency from I/Q data.
GetFreqDataLevelMask	Calculate frequency from I/Q data and mask the result if the amplitude is less than the threshold.
GetFreqDataLevelMaskWith-Mean	Calculate frequency and the mean value from I/Q data, and mask the result if the amplitude is less than the threshold.
GetFreqDataWithMean	Calculate frequency and the mean value from I/Q data.
GetIData	Calculate I data from I/Q data.
GetIDataWithMean	Calculate I data and the mean value from I/Q data.
GetIQDataBandPass	Pass I/Q data through the band-pass filter.
GetIQFFT	Multiply I/Q data by the window function and perform FFT.
GetIQFFTOffset	Subtract offset from I/Q data, multiply it by the window function and perform FFT.
GetIQPData	Calculate I/Q and phase from I/Q data.
GetIQPDataLevelMask	Calculate I/Q and phase from I/Q data and mask the result if the amplitude is less than the threshold.
GetIQPDataLevelMaskWith-Mean	Calculate I/Q, phase, and the mean value from I/Q data and mask the result if the amplitude is less than the threshold.
GetIQPDataWithMean	Calculate I/Q, phase, and the mean value from I/Q data.
GetPhaseData	Calculate phase from I/Q data.
GetPhaseDataLevelMask	Calculate phase from I/Q data and mask the result if the amplitude is less than the threshold.
GetQData	Calculate Q data from I/Q data.
GetQDataWithMean	Calculate Q data and the mean value from I/Q data.
GetWeightTableSize	Return the array size for storing the filter weighting factor.
MakeFFTWindowTable	Return the correction factor from the periodic FFT window.
MakeWeightTable	Make the weighting factor table and return the table size.

Miscellaneous

Miscellaneous functions do not fit into other categories.
Refer to page 2–81 for details.

Table 2–13: Miscellaneous functions

Function	Description
Ceil	Round up a numeric value toward positive infinity.
ChangeFileExt	Change file extension.
CharToInt	Convert the head character in a string to a numeric value.
ClearText	Clear output from functions such as `writeln`.
CompareText	Compare two strings with case insensitive.
CopyLongIntArray	Copy a long integer-type array.
CopyRealArray	Copy a real-type array.
CutStr	Cut a string out of the original string.
EngToFloat	Convert a string including the SI unit to the numeric value.
EngToFloatDef	Convert a string that contains SI units into the numeric value.
ExtractFileName	Return the file name with extension excluding the path.
ExtractFilePath	Return a file path.
FileExists	Check if a specified file exists.
FloatToEng	Convert a numeric value to the string including the SI unit.
FloatToStr	Convert a numeric value to the string.
Floor	Round down a numeric value toward negative infinity.
FormatEng	Convert a numeric value to a string containing SI units.
FormatFloat	Convert a numeric value to a string.
GetExePath	Return a path that includes a script executable file.
GetMinMaxMean	Calculate the maximum, minimum, and mean value of data.
GetMinMaxMeanRegion	Calculate the maximum, minimum, and mean value of data in the specified range.
GetMinMaxMeanRegionWith-Mask	Calculate the maximum, minimum, and mean value of data not masked in the specified range.
GetMinMaxMeanWithMask	Calculate the maximum, minimum, and mean value of data not masked.
GetRunState	Return the run state of the main procedure.
GetSmooth	Smooth data.
GetViewName	Return the name of the view that started a script.
IntToHex	Convert a value to the hexadecimal representation.

Table 2–13: Miscellaneous functions (Cont.)

Function	Description
MakePointsArray	Make an integer array.
ScriptMessageBox	Display a message box.
SetRunState	Set the run state of the main procedure.
SetStatusMessage	Specify a string shown in the status display area.
SetTitle	Set a string that appears at the top of a display of a script.
SetViewTitle	Set a view title displayed at the top of the window.
StrToIntDef	Convert a string to the numeric value.
Terminated	Check if the Break side key has been pressed.
Trim	Delete blank at the head and end of a string.
WaitMilliSecond	Wait in millisecond.

Command-Related Functions

GetCommandString

When a command is sent from an external program to a script, the procedure with the same name as the command is invoked. Use the GetCommandString function in the invoked procedure to obtain the command parameters.

Syntax

```
function GetCommandString: String;
```

SendCommand

Sends a command and argument strings similar to GPIB commands to other system programs such as CONFIG, SETUP, VIEW and UTIL. Unlike GPIB commands, these strings are not preceded by a colon.

Syntax

```
function SendCommand(command: String; argument: String): Longint;
```

Arguments

`command: String` specifies the command string.

`argument: String` specifies the argument for the command.

Returns

The following error codes may be returned. These codes are defined as constants.

ERR_NO_ERROR	= 0;
ERR_COMMAND_ERROR	= –100;
ERR_EXECUTION_ERROR	= –200;
ERR_PARAMETER_ERROR	= –220;
ERR_MASS_STRAGE_ERROR	= –250;
ERR_MEDIA_FULL	= –254;
ERR_FILE_NAME_NOT_FOUND	= –256;
ERR_MEDIA_PROTECTED	= –258;

Examples

```
SendCommand('Setup:Span', '5M');
```
sets the span to 5 MHz.

SendCommandIf

Sends a command string as a query command. When the response is different from the argument, both the command and argument are sent.

Syntax

```
function SendCommandIf(command: String; argument: String):
Longint;
```

Arguments

`command: String` specifies the command string.

`argument: String` specifies the argument for the command.

SendQuery

Sends a query command similar to GPIB commands to other system programs such as CONFIG, SETUP, VIEW and UTIL. Unlike GPIB commands, the query command is not preceded by a colon.

NOTE. *A question mark (?) is required at the end of the query command.*

Syntax

```
function SendQuery(command: String; var value: String): Longint;
```

Arguments

`command: String` specifies the command string.

`argument: String` specifies the argument for the command.

Returns

The error codes same as those returned by SendCommand.

Examples

The following program queries the span.

```
var
   value: query;
endVar

SendQuery('Setup:Span?', query);
```

SendQueryPas

Sends a query command similar to GPIB commands to other programs such as CONFIG, SETUP, VIEW and UTIL. Unlike GPIB commands, the query command is not preceded by a colon.

Syntax `function SendQueryPas(command: String): String;`

Arguments `command: String` specifies the command string.

Returns The response to the query.

Examples The following program queries the span.

```
var
   value: query;
endVar

SendQuery('Setup:Span?');
```

SetQueryString

When a query command is sent from an external program to a script, the procedure with the same name as the query command is invoked. Use the SetQueryString function in the invoked procedure to set the returned value for the query command.

Syntax `procedure SetQueryString(query: String);`

Arguments `query: String` specifies the returned value for the query command.

Examples `SetQueryString('5M');`
sets the returned value to “5M”.

GPIB Command-Related Functions

GetGPIBRWTimeOut

Queries the time-out when writing or reading to/from the external GPIB device.

Syntax `function GetGPIBRWTimeOut: LongInt;`

Returns The value 0 to 17 listed in Table 2–14 on page 2–19 corresponding to the time-out.

Related Functions `SetGPIBRWTimeOut`

GetGPIBSpr

Queries the serial poll status byte to the external GPIB device.

Syntax `function GetGPIBSpr(address: LongInt): LongInt;`

Arguments `address: LongInt` specifies the GPIB address of the external device.

Related Functions `InitiateGPIB`

InitiateGPIB

Makes the external GPIB device available. Execute this procedure always before using the SendGPIBCommand, SendGPIBQuery, SendGPIBQueryPas, and GetGPIBSpr functions.

Syntax `procedure InitiateGPIB;`

Related Functions `SendGPIBCommand`, `SendGPIBQuery`, `SendGPIBQueryPas`, `GetGPIBSpr`

SendGPIBCommand

Sends the command string to the external GPIB device.

Syntax

```
function SendGPIBCommand(address: LongInt; command: String):
Longint;
```

Arguments

`address: LongInt` specifies the primary address of the external GPIB device.

`command: String` specifies the command string sent to the external GPIB device.

Returns

`0` indicates that the command has been sent successfully.

`–1` indicates that the command has been sent unsuccessfully.

Related Functions

`InitiateGPIB`

SendGPIBQuery

Sends the query command string to the external GPIB device and indicates the query command status, either successfull (0) or unsuccessfull (-1).

Syntax

```
function SendGPIBQuery(address: LongInt; command: String;
var value: String): Longint;
```

Arguments

`address: LongInt` specifies the primary address of the external GPIB device.

`command: String` specifies the query command string sent to the external GPIB device.

Returns

`0` indicates that the query command has been sent successfully.

`–1` indicates that the query command has been sent unsuccessfully.

Related Functions

`InitiateGPIB`

SendGPIBQueryPas

Send the query command string to the external GPIB device and returns the response for the query.

Syntax

```
function SendGPIBQueryPas(address: LongInt; command: String):
String;
```

Arguments

`address: LongInt` specifies the primary address of the external GPIB device.

`command: String` specifies the query command string sent to the external GPIB device.

Returns

The response for the query.

Related Functions

`InitiateGPIB`

SetGPIBRWTimeOut

Sets the time-out when writing or reading to/from the external GPIB device.

Syntax

```
procedure SetGPIBRWTimeOut(tmo: LongInt);
```

Arguments

`tmo: LongInt` specifies the number 0 to 17 corresponding to the time-out as listed in Table 2–14.

Table 2–14: Time-out setting

Argument	Time-out	Argument	Time-out	Argument	Time-out
0	0	6	3 ms	12	3 s
1	10 μs	7	10 ms	13	10 s
2	30 μs	8	30 ms	14	30 s
3	100 μs	9	100 ms	15	100 s
4	300 μs	10	300 ms	16	300 s
5	1 ms	11	1 s	17	1000 s

Related Functions

`GetGPIBRWTimeOut`

TerminateGPIB

Enables the Util8:GPIB (Remote) system program, which has been disabled by executing the InitiateGPIB procedure.

Syntax

```
procedure TerminateGPIB;
```

Related Functions

`InitiateGPIB`

Data Acquisition-Related Functions

GetMultiBlock

When acquisition of one block of data is completed, GetMultiBlock stops the acquisition immediately. Otherwise, it waits for the maximum time-out seconds before it stops the acquisition.

When several blocks are acquired, calling StartMultiBlock, GetMultiBlock, RestartMultiBlock, and StopMultiBlock in combination is more efficient than calling GetOneBlock several times. You can also run the process that generates a trigger event after StartMultiBlock or RestartMultiBlock if necessary.

Syntax

```
function GetMultiBlock(Timeout: Longint; var Flags: WORD;
Events: Longint): Boolean;
```

Arguments

Refer to `GetOneBlock` below.

GetOneBlock

Acquires one block of data.

Syntax

```
function GetOneBlock(Trigger: String; BlockSize: Longint;
Timeout: Longint; var Flags: WORD; Events: Longint): Boolean;
```

Arguments

`Trigger: String` specifies the trigger mode for Setup:Trigger.
If space characters are specified, the current Setup:Trigger value is used.

`BlockSize: Longint` specifies the block size for Setup:BlockSize.
If space characters are specified, the current Setup:BlockSize value is used.

`Timeout: Longint` forcefully terminates acquisition and returns False, if data acquisition is not completed within the Timeout seconds.

`var Flags: WORD` returns the appropriate information per bit. These values are defined as constants. The meanings are also shown below:

FRAME_INVALID	= $1;	The frame has no valid data.
FRAME_LASTFRAME	= $2;	The frame is the last one.
FRAME_OVERLOAD	= $8;	The analyzer is overloaded.
FRAME_TRIGGERED	= $10;	The analyzer is triggered.

`Events: Longint` sends an event that has occurred during data acquisition to the program(s) corresponding to the bit(s) turned on, as listed below:

SETUP	= $10000;
VIEW1	= $08000;
VIEW2	= $04000;
VIEW3	= $02000;
VIEW4	= $01000;
VIEW5	= $00800;
VIEW6	= $00400;
VIEW7	= $00200;
VIEW8	= $00100;
UTIL1	= $00080;
UTIL2	= $00040;
UTIL3	= $00020;
UTIL4	= $00010;
UTIL5	= $00008;
UTIL6	= $00004;
UTIL7	= $00002;
UTIL8	= $00001

Returns

`True` indicates successful completion.

`False` indicates forceful termination due to a time-out.

Examples

The following program acquires 100 frames of data in the auto trigger mode. Only the VIEW1 program updates its display.

```
var
   Flags: WORD;
endVar

GetOneBlock('Auto', 100, 10, Flags, $08000);
```

RebuildWindowTable

RebuildWindowTable is called when other system programs such as SETUP, VIEW, and UTIL are restarted so that events are sent correctly on data acquisition.

Syntax

```
procedure RebuildWindowTable;
```

RestartMultiBlock

Restarts acquiring the second and subsequent blocks when you have started the acquisition with the procedure StartMultiBlock.

When several blocks are acquired, calling StartMultiBlock, GetMultiBlock, RestartMultiBlock, and StopMultiBlock in combination is more efficient than calling GetOneBlock several times. You can also run the process that generates a trigger event after StartMultiBlock or RestartMultiBlock if necessary.

Syntax

```
procedure RestartMultiBlock(Events: Longint);
```

Arguments

`Events: Longint` (refer to page 2–22)

Related Functions

`StartMultiBlock`

StartMultiBlock

Starts acquiring the first block.

When several blocks are acquired, calling StartMultiBlock, GetMultiBlock, RestartMultiBlock, and StopMultiBlock in combination is more efficient than calling GetOneBlock several times. You can also run the process that generates a trigger event after StartMultiBlock or RestartMultiBlock if necessary.

Syntax

```
procedure StartMultiBlock(Trigger: String; BlockSize: Longint;
Events: Longint);
```

Arguments

Refer to GetOneBlock on page 2–21.

Examples

The following program acquires three blocks (1 block = 100 frames) of data in the auto trigger mode. Only the VIEW1 program updates its display.

```
var
   Flags: WORD;
endVar

StartMultiBlock('Auto', 100, $08000);
GetMultiBlock(10, Flags, $08000);
RestartMultiBlock($08000);
GetMultiBlock(10, Flags, $08000);
RestartMultiBlock($08000);
GetMultiBlock(10, Flags, $08000);
StopMultiBlock($08000);
```

Related Functions

RestartMultiBlock

StopMultiBlock

Stops data acquisition started with the procedure StartMultiBlock and restores the parameters Trigger and BlockSize.

When several blocks are acquired, calling StartMultiBlock, GetMultiBlock, RestartMultiBlock, and StopMultiBlock in combination is more efficient than calling GetOneBlock several times. You can also run the process that generates a trigger event after StartMultiBlock or RestartMultiBlock if necessary.

Syntax

```
procedure StopMultiBlock(Events: Longint);
```

Data Read-Related Functions

GetData

Reads FFTPoints data items without any corrections. In case of frequency domain, center frequency position data is returned at the array index 0. Neither gain nor phase correction is done to the returned data.

Syntax

```
function GetData(SourceName: String; FormatName: String;
Z: Longint; var Data: Real; var TimeStamp: Real): Longint;
```

Arguments

`SourceName: String` (Refer to GetHeader on page 2–27).

`FormatName: String` (Refer to GetHeader on page 2–27).

`Z: Longint` (Refer to GetFrameHeader on page 2–26).

`var Data: Real` returns data. Prepare an array with FFTPoints elements.

`var TimeStamp: Real` returns the elapse time from frame 0 (the current frame).

Returns

Refer to GetFrameHeader on page 2–26.

GetFlatness

Returns the Gain Correction [dB] and Phase Correction [deg] Tables. Data is corrected by subtracting data before a correction from the value on the table. These tables have FFTPoints data items with center frequency position data stored at the table index 0.

Syntax

```
procedure GetFlatness(SourceName: String; var Ampl: Real;
var Phase: Real);
```

Arguments

`SourceName: String` (Refer to GetHeader on page 2–27).

`var Ampl: Real` returns the Gain Correction Table [dB].

`var Phase: Real` returns the Gain Correction Table [deg].

GetFrameHeader

Reads the header information on a specified frame.

Syntax

```
function GetFrameHeader(SourceName: String; Z: Longint;
var Ticks: Longint; var DateTime: String): Longint;
```

Arguments

`SourceName: String` (Refer to GetHeader on page 2–27).

`Z: Longint` specifies a frame between 0 and ZNum –1. `0` indicates the current frame.

`var Ticks: Longint` returns the counter value since data acquisition began. Uses the TickToSecond() function to calculate the time between two frames.

`var DateTime: String` returns the date and time when the specified frame is acquired.

Returns

The appropriate information per bit is returned.

FRAME_INVALID	= $1;
FRAME_LASTFRAME	= $2;
FRAME_OVERLOAD	= $8;
FRAME_TRIGGERED	= $10

GetHeader

Reads header information from the specified source.

Syntax

```
procedure GetHeader(SourceName: String; FormatName: String;
Name: String; var Value: String);
```

Arguments

`SourceName: String` specifies one of the following:

Active	Data read into memory
Zoom	Data in memory created by the zoom function
D1 to D8	Data in a data register
D1D2 to D7D8	Data in a pair register
File path	Data in a file

`FormatName: String` specifies one of the following formats:

FreqAmpl	Amplitude in frequency domain [dBm]
FreqPhase	Phase in frequency domain [deg]
FreqI, FreqQ	I and Q components in frequency domain [V]
TimeAmpl	Amplitude in time domain [dBm]
TimePhase	Phase in time domain [deg]
TimeI, TimeQ	I and Q components in time domain [V]

`Name: String` specifies a header.

XNum	The number of data items on the X axis (e.g., ‘801’, ‘1024’)
XScale	The range of the X axis (e.g., ‘5.00625M’, ‘80u’)
XStart	The minimum value on the X axis (e.g., ‘2.5M’, ‘0’)
XUnit	The unit for the X axis (e.g., ‘Hz’, ‘s’)
XLeftLabel	The label of the X axis in the left direction (e.g., ‘Center’, ‘Start’)
XRightLabel	The label of the X axis in the right direction (e.g., ‘Span’, ‘Stop’)
YScale	The range of the Y axis (e.g., ‘100’, ‘100m’)
YStart	The minimum value on the Y axis (e.g., ‘–100’, ‘–50m’)
YUnit	The unit for the Y axis (e.g., ‘dBm’, ‘V’)
YMiddleUnit	The unit for the Y axis (e.g., ‘dB’, ‘V’)
ZNum	The number of frames (e.g., ‘200’)
ZUnit	The unit of frames (e.g., ‘frame’)

`var Value: String` returns the value of the header specified in Name.

GetHeaderPas

Reads header information from the specified source.

Syntax

```
procedure GetHeaderPas(SourceName: String; FormatName: String;
Name: String): String;
```

Arguments Refer to the arguments of the GetHeader function on page 2–27.

GetMinMax

Returns the minimum and maximum values within the range specified by n0 and n1 to Min and Max, respectively. Also it returns their positions to nMin and nMax, respectively. For FreqAmpl, gain corrections are made. For FreqPhase, phase corrections are made.

Syntax

```
procedure GetMinMax(SourceName: String; FormatName: String;
Z: Longint; n0: Longint; n1: Longint; var nMin: Longint;
var nMax: Longint; var Min: Real; var Max: Real);
```

Arguments

`SourceName: String` (Refer to GetHeader on page 2–27).

`FormatName: String` (Refer to GetHeader on page 2–27).

`Z: Longint` (Refer to GetFrameHeader on page 2–26).

`n0: Longint, n1: Longint` specifies the range in which the minimum and maximum values are searched. The range includes n0 and n1.

`var nMin: Longint, var nMax: Longint` returns the positions of the minimum and maximum values.

`var Min: Real, var Max: Real` returns the minimum and maximum values.

GetPointsWithMask

Returns data after thinning or compressing it (to facilitate display), according to the content of the Points array. This function can be used when one trace consists of several frames like 3 GHz span. Use GetData() to read data without any corrections. For FreqAmpl, gain corrections are made. For FreqPhase, phase corrections are made.

Syntax

```
function GetPointsWithMask(SourceName: String;
FormatName: String; Z: Longint; p0: Longint; p1: Longint;
MinMax: Longint; PointNum: Longint; var Points: Longint;
var DataMin: Real; var DataMax: Real; var Mask: Longint;
var TimeStamp: Real): Longint;
```

Arguments

`SourceName: String` (Refer to GetHeader on page 2–27).

`FormatName: String` (Refer to GetHeader on page 2–27).

`Z: Longint` (Refer to GetFrameHeade on page 2–26).

`p0: Longint, p1: Longint` specifies the range of the Points array used for this function. The range includes p0 and p1.

`MinMax: Longint`

> 0 means that the data at the positions indicated by Points array elements is written into the DataMax array.
>
> –1 means that the minimum and maximum values between the data at the position indicated by the Points array (n) and the data at the position immediately before the position indicated by the Points array (n+1) is written into the DataMin and DataMax arrays, respectively.

`PointNum: Longint` specifies the sizes of the DataMin, DataMax, Mask arrays

`var Points: Longint` is a point array retaining positions of data written to the DataMin, DataMax and Mask arrays. You must store the position of the last data +1 in the Points array (PointNum +1).

`var DataMin: Real` and `var DataMax: Real` are arrays that return the minimum and maximum values, respectively.

var Mask: Longint is an Array that returns mask information

> 0 indicates valid data.
> –1 indicates invalid data.

`var TimeStamp: Real` (Refer to GetData on page 2–25).

Returns

Refer to GetFrameHeader on page 2–26.

GetTimeStamp

Reads the time stamp of a specified frame.

Syntax

```
function GetTimeStamp(SourceName: String; Z: Longint;
var TimeStamp: Real): Longint;
```

Arguments

SourceName: String (Refer to GetHeader on page 2–27).

`Longint` (Refer to GetHeader on page 2–27).

`var TimeStamp: Real` (Refer to GetData on page 2–25)

Returns

Refer to GetFrameHeader on page 2–26.

Register-Related Functions

DregScroll

Shifts a data register right or left by a specified number of data points. After the shift, vacant area is filled with zero (see Figure 2–1).

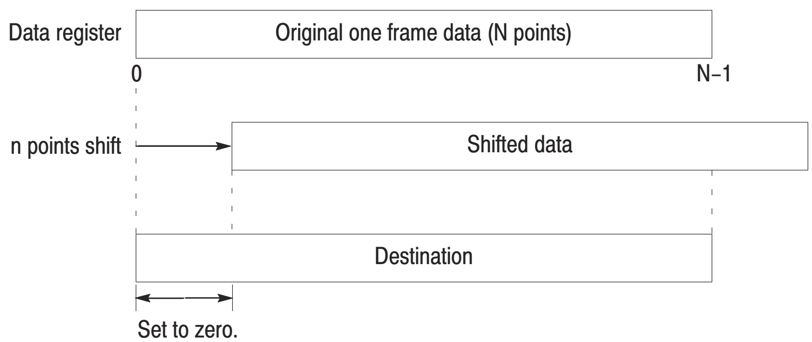


Figure 2–1: DregScroll functioning

Syntax

`procedure DregScroll(SourceName: String; n: Longint);`

Arguments

`SourceName: String` (refer to DregSetHeader on page 2–33)

`n: Longint` specifies the number of data points to shift.

n < 0	Shifts left.
n > 0	Shifts right.

DregSetData

Writes data into a data register. If XNum is not specified by DregSetHeader (refer to page 2–33), data cannot be written.

Syntax

```
procedure DregSetData(SourceName: String; var Data: Real);
```

Arguments

`SourceName: String` (refer to DregSetHeader on page 2–33)

`var Data: Real` writes data into a data register. Prepare data with a length that equals or exceeds the value specified for Xnum of the header.

DregSetDataWithMask

Writes data and mask information into a data register. If XNum is not specified by DregSetHeader (refer to page 2–33), data cannot be written.

Syntax

```
procedure DregSetDataWithMask(SourceName: String; var Data: Real;
var Mask: Longint);
```

Arguments

`SourceName: String` (Refer to DregSetHeader on page 2–33)

`var Data: Real` indicates data to be written into a data register. Prepare data with a length that equals or exceeds the value specified for Xnum of the header.

`var Mask: Longint` indicates the mask information written into a data register. Prepare data with a length that equals or exceeds the value specified for Xnum of the header.

`0` indicates valid data.
`-1` indicates invalid data.

DregSetHeader

Sets the header values of a data register. Changing the Xnum value will reset other header values in the data register. Begin with setting the Xnum value.

Syntax

```
procedure DregSetHeader(SourceName: String; Name: String;
Value: String);
```

Arguments

SourceName: String specifies one of the following registers:

‘D1’ to ‘D8’	Data registers
‘D1D2’ to ‘D7D8’	Pair registers

`Name: String` specifies a header.

XNum	The number of data items on the X axis (e.g., ‘801’, ‘1024’)
XScale	The range of the X axis (e.g., ‘5.00625M’, ‘80u’)
XStart	The minimum value on the X axis (e.g., ‘2.5M’, ‘0’)
XUnit	The unit for the X axis (e.g., ‘Hz’, ‘s’)
XLeftLabel	The label of the X axis in the left direction (e.g., ‘Center’, ‘Start’)
XRightLabel	The label of the X axis in the right direction (e.g., ‘Span’, ‘Stop’)
YScale	The range of the Y axis (e.g., ‘100’, ‘100m’)
YStart	The minimum value on the Y axis (e.g., ‘–100’, ‘–50m’)
YUnit	The unit for the Y axis (e.g., ‘dBm’, ‘V’)
YMiddleUnit	The unit for the Y axis (e.g., ‘dB’, ‘V’)
ZNum	The number of frames (e.g., ‘200’)
ZUnit	The unit for frames (e.g., ‘frame’)

`Value: String` specifies the value of the header specified in Name.

DregSetPoint

Replaces a value at a specified position of a data register.

Syntax

```
procedure DregSetPoint(SourceName: String; n: Longint;
Value: Real);
```

Arguments

`SourceName: String` (Refer to DregSetHeader on page 2–33)

`n: Longint` specifies the data point to be changed.

`Value: Real` specifies a new value.

DregSetPointWithMask

Changes a value or mask information at a specified position of a data register.

Syntax

```
procedure DregSetPointWithMask(SourceName: String; n: Longint;
Value: Real; Mask: Longint);
```

Arguments

`SourceName: String` (Refer to DregSetHeader on page 2–33)

`n: Longint` specifies the data point to be changed.

`Value: Real` specifies a new value.

`Mask: Longint` specifies new mask information.

Event/Marker-Related Functions

MsgGetCFromMessage

Used in the MarkerEvents function to track changes of the carrier frequency on digital demodulation (C-markers).

Syntax `procedure MsgGetCFromMessage(var Marker: String);`

Arguments `var Marker: String` returns the value of the C marker.

MsgGetCFromMessagePas

Used in the MarkerEvents function to track changes of the carrier frequency on digital demodulation (C-markers).

Syntax `procedure MsgGetCFromMessagePas: String;`

Returns The value of the C marker.

MsgGetEventFromMessage

Used in the ActiveEvent() function called when acquiring data, or in the DregEvent() function called when updating registers.

Syntax

```
function MsgGetEventFromMessage(SourceName: String): Longint;
```

Arguments

`SourceName: String` specifies “Active” or a register name.

Returns

If this function is called with SourceName set to Active in the ActiveEvent(), one of the following event numbers is returned. These values are defined as constants.

EVT_IDLE	= 0;
EVT_START	= 1;
EVT_STOP	= 2;
EVT_ROLL	= 3;
EVT_CHANGEROLL	= 4;
EVT_BLOCK	= 5;
EVT_CHANGEBLOCK	= 6;
EVT_POWEROFF	= 7;
EVT_RESTART	= 8;
EVT_ZOOMSTART	= 9;
EVT_ZOOMBLOCK	= 10;
EVT_ZOOMSTOP	= 11;
EVT_PREPARE	= 12;
EVT_ENDROLL	= 13;
EVT_QUICK	= 14;
EVT_CHANGEENDROLL	= 15;
EVT_MONITOR	= 16;
EVT_CHANGEMONITOR	= 17

If this function is called with SourceName set to a value such as D1 in the DregEvent() function, one of the following event numbers is returned. These values are defined as constants.

VT_ROLL	= 3;
EVT_CHANGEROLL	= 4;

MsgGetXFromMessage, MsgGetDeltaXFromMessage

Used in the MarkerEvent function called when updating markers to obtain the values of the X and DeltaX markers, respectively.

Syntax

```
procedure MsgGetXFromMessage(XUnit: String; var Marker: String);
```

```
procedure MsgGetDeltaXFromMessage(XUnit: String;
var Marker: String);
```

Arguments

`XUnit: String` specifies one of the following message types:

Hz	Frequency domain
s	Time domain

`var Marker: String` returns the value of the X or DeltaX marker.

MsgGetXFromMessagePas, MsgGetDeltaXFromMessagePas

Used in the MarkerEvent function called when updating markers to obtain the values of the X and Delta-X markers, respectively.

Syntax

```
procedure MsgGetXFromMessagePas(XUnit: String): String;
```

```
procedure MsgGetDeltaXFromMessagePas(XUnit: String): String;
```

Arguments

`XUnit: String` specifies one of the following message types:

Hz	Frequency domain
s	Time domain

Returns

The value of the X or Delta-X marker.

MsgGetZFromMessage, MsgGetDeltaZFromMessage

Used in the MarkerEvent function called when updating markers to obtain the values of the Z and DeltaZ markers, respectively.

Syntax

```
procedure MsgGetZFromMessage(SourceName: String;
var Marker: String);
```

```
procedure MsgGetDeltaZFromMessage(SourceName: String;
var Marker: String);
```

Arguments

`SourceName: String` specifies one of the following message types:

Active
Zoom
File path

`var Marker: String` returns the value of the Z or DeltaZ marker.

MsgGetZFromMessagePas, MsgGetDeltaZFromMessagePas

Used in the MarkerEvent function called when updating markers to obtain the values of the Z and Delta-Z markers, respectively.

Syntax

```
procedure MsgGetZFromMessagePas(SourceName: String): String;
```

```
procedure MsgGetDeltaZFromMessagePas(SourceName: String): String;
```

Arguments

`SourceName: String` specifies one of the following message types:

Active
Zoom
File path

Returns

The value of the Z or Delta-Z marker.

MsgSetMarkerC

This function is used to set the value of the C marker used in the Polar View program and track changes of the carrier frequency on digital demodulation. The CONFIG program notifies the SETUP, VIEW, and UTIL programs of this value.

Syntax

```
procedure MsgSetMarkerC(Marker: String);
```

Arguments

`Marker: String` specifies the value of the C Marker.

MsgSetMarkerX, MsgSetDeltaX

Set the values of the X and DeltaX markers, respectively. The CONFIG program notifies the SETUP, VIEW, and UTIL programs of these values.

Syntax

```
procedure MsgSetMarkerX(XUnit: String; Marker: String);
```

```
procedure MsgSetDeltaX(XUnit: String; Marker: String)
```

Arguments

`XUnit: String` specifies one of the following message types:

Hz	Frequency domain
s	Time domain

`Marker: String` specifies the value of the X or DeltaX marker.

MsgSetMarkerZ, MsgSetDeltaZ

Set the values of the Z and DeltaZ markers, respectively. The CONFIG program notifies the SETUP, VIEW, and UTIL programs of these values.

Syntax

```
procedure MsgSetMarkerZ(SourceName: String; Marker: String);
```

```
procedure MsgSetDeltaZ(SourceName: String; Marker: String)
```

Arguments

`SourceName: String` specifies one of the following message types:

Active
Zoom
File path

`Marker: String` specifies the value of the Z or DeltaZ marker.

Numerical Calculation-Related Functions

Fft

Performs FFT or IFFT.

Syntax

```
function Fft(n: Longint; var x: Real; var y: Real): Longint;
```

Arguments

`n: Longint` specifies the number of FFT points. Specifying 0 releases the allocated area.

n < 0	IFFT
n > 0	FFT

`var x: Real, var y: Real` specifies arrays of real numbers and imaginary numbers., where calculation results are returned.

Returns

0 indicates successful completion.

1 indicates abnormal termination due to memory shortage.

MakeSplineTable

Creates auxiliary data for spline interpolation. Refer to Spline on page 2–42.

Syntax

```
procedure MakeSplineTable(n: Longint; var x: Real; var y: Real;
var z: Real);
```

Arguments

`n: Longint` specifies the number of data items.

`var x: Real, var y: Real` contains original data.

`var z: Real` contains auxiliary data.

QuickSort

Performs quick sort.

Syntax

```
procedure quicksort(n: Longint; var a: Real);
```

Arguments

`n: Longint` specifies the number of data items.

`var a: Real` contains input data. Results are overwritten.

Spline

Calculates the vertical coordinate corresponding to any horizontal coordinate with spline interpolation using auxiliary data created by MakeSplineTable.

Syntax

```
procedure Spline(n: Longint; t: Real; var ret: Real; var x: Real;
var y: Real; var z: Real);
```

Arguments

`n: Longint` specifies the number of data items.

`t: Real` specifies a horizontal coordinate.

`var ret: Real` returns a vertical coordinate.

`var x: Real, var y: Real` contains original data.

`var z: Real` contains auxiliary data. Refer to MakeSplineTable on page 2–41.

Grid-Related Functions

AdjustColWidths

Adjusts the column width so that strings in fixed–length columns can be seen in full. This function must be called after setting strings, using the SetCell() function, in all the fixed-length columns.

Syntax

```
procedure AdjustColWidths;
```

GetCell

Returns a string in a cell.

Syntax

```
function GetCell(col: Longint; row: Longint): String;
```

GetCellAttr

Returns an attribute of a cell.

Syntax

```
function GetCellAttr(c: Longint; r: Longint): Longint;
```

GetColCount, GetRowCount

Return the number of columns and rows, respectively.

Syntax

```
function GetColCount: Longint;

function GetRowCount: Longint;
```

GetCol, GetRow

Return the current column and row, respectively.

Syntax

```
function GetCol: Longint;

function GetRow: Longint;
```

GetFixedCols, GetFixedRows

Return the number of fixed-length columns and rows, respectively.

Syntax

```
function GetFixedCols: Longint;

function GetFixedRows: Longint;
```

RefreshCells

Updates a grid display.

Syntax

```
procedure RefreshCells;
```

SetCell

Sets a string in a cell.

Syntax

```
procedure SetCell(col: Longint; row: Longint; s: String);
```

SetCellAttr

Sets an attribute of a cell.

Syntax

```
procedure SetCellAttr(col: Longint; row: Longint; attr: Longint;
color: Longint; selectedColor: Longint);
```

Arguments

`attr: Longint` specifies monochrome or color display.

`0` specifies that a cell is displayed in black.

Other than `0` specifies that a cell is displayed in the color specified in color and selectedColor.

`color: Longint` specifies the color of unselected cells.
The following values are defined as constants:

clBlack	= $000000;
clMaroon	= $000080;
clGreen	= $008000;
clOlive	= $008080;
clNavy	= $800000;
clPurple	= $800080;
clTeal	= $808000;
clGray	= $808080;
clSilver	= $C0C0C0;
clRed	= $0000FF;
clLime	= $00FF00;
clYellow	= $00FFFF;
clBlue	= $FF0000;
clFuchsia	= $FF00FF;
clAqua	= $FFFF00;
clLtGray	= $C0C0C0;
clDkGray	= $808080;
clWhite	= $FFFFFF;

`selectedColor: Longint` specifies the color of selected cells.

SetColCount, SetRowCount

Set the number of columns and rows, respectively.

Syntax

```
procedure SetColCount(n: Longint);
```

```
procedure SetRowCount(n: Longint);
```

SetCol, SetRow

Set the current column and row, respectively.

Syntax

```
procedure SetCol(n: Longint);

procedure SetRow(n: Longint);
```

SetFixedCols, SetFixedRows

Set the number of fixed-length columns and rows, respectively.

Syntax

```
procedure SetFixedCols(n: Longint);

procedure SetFixedRows(n: Longint);
```

SetGridAlign

Determines the location and size of the grid area.

Note that the window displayed by a script can be divided into two parts:

Text area:	Displays characters using functions such as writeln().
Grid area:	Displays tables. The area is set by this SetGridAlign function.

Syntax

```
procedure SetGridAlign(Align: String; percent: Longint);
```

Arguments

`Align: String` specifies the location of the grid area.
Valid values are as follows:

`None`	Text area only
`Top`	
`Bottom`	
`Left`	
`Right`	
`Client`	Grid area only

`percent: Longint` specifies the percentage of the grid area.

Examples

`SetGridAlign('Left', 50);`
sets the left half of the window as a grid area.

Menu-Related Functions

MenuAppendButton

Adds a new menu side key.

Syntax

```
procedure MenuAppendButton(level: Longint; s: String);
```

Arguments

`level: Longint` specifies the level of menu to which side keys are added. 0 indicates the top level of the hierarchy.

`s: String` specifies the string that combines a side key label and a side key type with “ | ”. Note that the Unit at the end of the string is optional.

TButton (command side key)
Issues a command.
Format: Label|TButton|Unit

TComboBox (combo box)
Allows you to make a selection from two or more options.
Format: Label|TComboBox,Command|Options|Unit

TEdit (edit)
Allows you to enter a numeric value.
Format: Label|TEdit,Command|Range|Step Range|Step Initial Value|Unit
The possible values for Range and Step Range are:
`INT` – Displays Integers.
`FLT` – Displays real numbers without using M, k and so on.
`ENG` – Displays real numbers using M, k and so on.
`HEX` – Displays hex numbers.

TMenu (submenu)
Takes you to the menu one level below the current level.
(For example, the “More...” side key.)
Add a “Return” side key to return to the original menu level.
Format: Label|TMenu|Unit

TScroll (scroll)
Issues a command with an argument of +1 or –1 depending on scroll button operation.
Format: Label| TScroll,Command|Unit

TToggle (toggle)
Allows you to make a selection from two options like On/Off.
Format: Label | TToggle, Command | Candidate | Unit

Label Only
Adds a side key that only displays a label.
Note the following two labels have special meanings:
`Return` – Returns to the menu one level above the current level.
`Blank` – Blank side key

Examples

```
MenuAppendButton(0, 'Position | TComboBox, Position |
First, Second, Third');
```

adds the "Position" side key with the selections of First, Second, and Third.

```
MenuAppendButton(0, 'SG Freq | TEdit,Freq | ENG,0,3G |
ENG0.1,100M | 1M | Hz');
```

adds the "SG Freq" side key that enters the frequency ranging from 0 Hz to 3 GHz with the step from 0.1 Hz to 100 MHz and the step initial value of 1 MHz.

```
MenuAppendButton(0, 'More | TMenu');
```

adds a "More" side key that takes you to the menu one level below the current level.

```
MenuAppendButton(1, 'Return');
```

adds a "Return" side key to return to the original menu level.

```
MenuAppendButton(0, 'SG Output | TToggle, Output | Off, On');
```

adds a "SG Output" side key to turn the output on or off.

MenuChangeValue

Changes the set value displayed on the side key.

Syntax

```
procedure MenuChangeValue(Name: String; Value: String);
```

Arguments

`Name: String` specifies the name of the side key.

`Value: String` sets the value displayed on the side key.

MenuModify

Updates the entire menu display after adding all the side keys using MenuAppendButton. MenuModify can be omitted in the main procedure.

Syntax

```
procedure MenuModify;
```

MenuSetButton

Activates the specified side key.

Syntax

```
procedure MenuSetButton(Button: LongInt);
```

Arguments

`Button: LongInt` specifies the side key with 1 to 8.

Examples

`MenuSetButton(1);`
activates the first side key.

MenuSetEnable

Enable or disable the side key.

Syntax

```
procedure MenuSetEnable(Path: String; Flag: Boolean);
```

Arguments

`Path: String` specifies the path of the side key. Express the tree structure with colon(s) (:) from the top level. For example, `Top:Level1:Level2:Button1`.

`Flag: Boolean` sets `True` when enabling the side key or `False` when disabling the side key.

MenuSetPage

Displays the specified page. (One page displays menu items on one screen.)

Syntax

```
procedure MenuSetPage(Page: String);
```

Arguments

`Page: String` specifies the page. Express the tree structure with colon(s) (:) from the top level. For example, `Top:Level1:Level2`.

Examples

`MenuSetPage('TOP');`
displays the first page.

MenuSetVisible

Determines whether or not to display the side key.

Syntax `procedure MenuSetPage(Page: String);`

Arguments `Path: String` specifies the path of the side key. Express the tree structure with colon(s) (:) from the top level. For example, `Top:Level1:Level2:Button1`.

`Flag: Boolean` sets `True` when displaying the side key or `False` when not displaying the side key.

MenuUpdate

Updates only the value displayed on a side key.

Syntax `procedure UpdateMenu;`

String List-Related Functions

ListAdd

Adds a new string to a list.

Syntax `procedure ListAdd(id: Longint; s: String);`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

`s: String` contains a new string.

ListClear

Deletes all the string from a list.

Syntax `procedure ListClear(id: Longint);`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

ListDelete

Deletes the string specified in Index.

Syntax `procedure ListDelete(id: Longint; Index: Longint);`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

`Index: Longint` specifies the index of the string.
0 indicates the first string; 1 the second, and so on.

ListGetCount

Returns the number of strings in a list.

Syntax `function ListGetCount(id: Longint): Longint;`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

ListGetString

Returns the string at a given index.

Syntax `function ListGetString(id: Longint; Index: Longint): String;`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

`Index: Longint` specifies the index of the string.
0 indicates the first string; 1 the second, and so on.

Returns The string at the specified index.

ListGetValue

Returns only the value of the string associated with a specified name for strings with the format of "name = value".

Syntax `function ListGetValue(id: Longint; Name: String): String;`

Arguments `id: Longint` specifies a list. The range is 0 to 9.

`Name: String` specifies a name used as identifier.

Returns Only the value of the string associated with a specified name. If the appropriate string with the format of "name = value" cannot be found in a list, or if no strings match the specified Name, space characters are returned.

ListInsert

Inserts a string at the specified index.

Syntax
```
procedure ListInsert(id: Longint; Index: Longint; s: String);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

`Index: Longint` specifies the index at which a new string is inserted. 0 indicates the first string; 1 the second, and so on.

`s: String` contains a new string.

ListLoadFromConsole

Gets a text line on the console into a specified list. For the console, see Figure 1–2 on page 1–4.

Syntax
```
procedure ListLoadFromConsole(id: Longint);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

ListLoadFromFile

Loads a text line in a specified file into a specified list.

Syntax
```
procedure ListLoadFromFile(id: Longint; FileName: String);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

`FileName: String` specifies a file.

ListSaveToConsole

Displays a list string on the console. For the console, see Figure 1–2 on page 1–4.

Syntax
```
procedure ListSaveToConsole(id: Longint);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

ListSaveToFile

Saves a list string in a specified file.

Syntax

```
procedure ListSaveToFile(id: Longint; FileName: String);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

`FileName: String` specifies a file.

ListSetString

Changes a string at a given index.

Syntax

```
procedure ListSetString(id: Longint; Index: Longint; s: String);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

`Index: Longint` specifies the index of the string.
0 indicates the first string; 1 the second, and so on.

`s: String` contains a new string.

ListSetValue

Sets only the value of the string associated with a specified name for strings with the format of "name = value".

Syntax

```
procedure ListSetValue(id: Longint; Name: String; Value: String);
```

Arguments

`id: Longint` specifies a list. The range is 0 to 9.

`Name: String` specifies a name used as an identifier.

`Value: String` specifies a value to set.

Hardware-Related Functions

DrvGetState

Returns the data acquisition state. When data acquisition is not carried out, STT_IDLE or STT_POWEROFF is returned.

Syntax

```
function DrvGetState: Longint;
```

Returns

The following values are defined as constants.

STT_IDLE	= 0;
STT_POWEROFF	= 1;
STT_RUNNING	= 2;
STT_PRE_RUNNING	= 3;
STT_PRE_IDLE	= 4;
STT_DRAWING	= 5;
STT_ZOOM_RUNNING	= 6;
STT_ZOOM_PRE_IDLE	= 7;
STT_PREPARING	= 8;
STT_RESAMPLING	= 9;

DrvGetValue

Returns hardware settings without using the SETUP program.

Syntax

```
function DrvGetValue(Name: String; var Value: String): Longint;
```

Arguments

`Name: String` (refer to DrvSetValue on page 2–58). If you specify ‘Hardware’, the instrument name (‘WCA330’ or ‘WCA380’) is returned to Value.

`Value: String` (refer to DrvSetValue on page 2–58)

Returns

Refer to DrvSetValue on page 2–58.

Examples

The following program queries the span setting.

```
var
   Value: String;
endVar

DrvGetValue('Span', Value);
```

DrvGetValuePas

Returns hardware settings without using the SETUP program.

Syntax

```
function DrvGetValue(Name: String): String;
```

Arguments

`Name: String` (refer to DrvSetValue on page 2–58). If you specify ‘Hardware’, the instrument name (‘WCA330’ or ‘WCA380’) is returned.

Returns

The hardware settings.

Examples

The following program queries the span setting.

```
var
   Value: String;
endVar

Value := DrvGetValuePas('Span');
```

DrvReadDiagResult

Returns the version of a subsystem.

Syntax `function DrvReadDiagResult(Name: String): Longint;`

Arguments `Name: String` must be 'Version'.

Returns Other than `-1` indicates the version number. Above the decimal point returns to the upper eight bits and the value below to the lower eight bits.

`-1` indicates the software is the PC version (SL7PCW3).

DrvRestoreSetup

Restores hardware settings saved by DrvSaveSetup below.

Syntax `procedure DrvRestoreSetup;`

DrvSaveSetup

Saves hardware settings. Use DrvRestoreSetup to restore the settings.

Syntax `procedure DrvSaveSetup;`

DrvSetValue

Sets up hardware without using the SETUP program.

Syntax

```
function DrvSetValue(Name: String; Value: String): Longint;
```

Arguments

`Name: String` specifies the standard commands provided by the SETUP program except Load, MarkerToFreq, MaxSpan, Save, Version and Zoom:*.

`Value: String` is the same parameter that the standard commands in the SETUP program use.

Returns

One of the following error codes:

ERR_NO_ERROR	= 0;
ERR_COMMAND_ERROR	= –100;
ERR_PARAMETER_ERROR	= –220;

Examples

`DrvSetValue('Span', '5M');`
sets the span to 5 MHz.

IQ Calculation-Related Functions

GetAmplData

Calculates amplitude from time-domain I and Q data with the specified unit.

Syntax

```
procedure GetAmplData(AUnit: String; Num: LongInt;
DataI: array of Real; DataQ: array of Real; Data: array of Real;
Mask: array of LongInt);
```

Arguments

`AUnit: String` specifies the unit of amplitude: W, V, dBm, dBuV, dBmV, or dBV.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`Data: array of Real` specifies the array that will store the resulting amplitude data.

`Mask: array of LongInt` specifies the array that will store the mask data for the amplitude data. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetAmplDataWithMean

Calculates amplitude from time-domain I and Q data with the specified unit, and returns the mean value. The mean value is calculated for the power if the specified unit is W or dBm, or for the voltage if any other unit is specified.

Syntax

```
procedure GetAmplDataWithMean(AUnit: String; Num: LongInt;
DataI: array of Real; DataQ: array of Real; var Mean: Real;
Data: array of Real; Mask: array of LongInt);
```

Arguments

`AUnit: String` specifies the unit of amplitude: W, V, dBm, dBuV, dBmV, or dBV.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`var Mean: Real` returns the mean value of power or voltage.

`Data: array of Real` specifies the array that will store the resulting amplitude data.

`Mask: array of LongInt` specifies the array that will store the mask data for the amplitude data. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetFreqData

Calculates frequency from time-domain I and Q data.

Syntax

```
procedure GetFreqData (Center: Real; DeltaT: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; Data: array of Real;
Mask: array of longInt);
```

Arguments

`Center: Real` sets the center frequency.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`Data: array of Real` specifies the array that will store the resulting frequency data.

`Mask: array of LongInt` specifies the array that will store the mask data for the frequency data. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetFreqDataLevelMask

Calculates frequency from time-domain I and Q data and masks the result if the amplitude is less than the specified threshold.

Syntax

```
procedure GetFreqDataLevelMask (Center: Real; DeltaT: Real;
Threshold: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; Data: array of Real;
Mask: array of longInt);
```

Arguments

`Center: Real` sets the center frequency.

`DeltaT: Real` specifies the time interval of the data.

`Threshold: Real` specifies the amplitude threshold in dBm for masking the result.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`Data: array of Real` specifies the array that will store the resulting frequency data.

`Mask: array of LongInt` specifies the array that will store the mask data for the frequency data. The mask has two values: 0 if the amplitude is greater than the threshold and –1 if less than or equal to the threshold.

GetFreqDataLevelMaskWithMean

Calculates frequency and the mean value from time-domain I and Q data and masks the result if the amplitude is less than the specified threshold.

Syntax

```
procedure GetFreqDataLevelMaskWithMean(Center: Real;
DeltaT: Real; Threshold: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; var Mean: Real;
Data: array of Real; Mask: array of longInt);
```

Arguments

`Center: Real` sets the center frequency.

`DeltaT: Real` specifies the time interval of the data.

`Threshold: Real` specifies the amplitude threshold in dBm for masking the result.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`var Mean: Real` returns the mean value of the frequency data.

`Data: array of Real` specifies the array that will store the resulting frequency data.

`Mask: array of LongInt` specifies the array that will store the mask data for the frequency data. The mask has two values: 0 if the amplitude is greater than the threshold and –1 if less than or equal to the threshold.

GetFreqDataWithMean

Calculates frequency and the mean value from time-domain I and Q data.

Syntax

```
procedure GetFreqDataWithMean(Center: Real; DeltaT: Real;
Num: LongInt; DataI: array of Real; DataQ: array of Real;
var Mean: Real; Data: array of Real; Mask: array of longInt);
```

Arguments

`Center: Real` sets the center frequency.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`var Mean: Real` returns the mean value of the frequency data.

`Data: array of Real` specifies the array that will store the resulting frequency data.

`Mask: array of LongInt` specifies the array that will store the mask data for the frequency data. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetIData

Calculates I data at the specified carrier frequency from time-domain I and Q data.

Syntax

```
procedure GetIData (Center: Real; Carrier: Real; Offset: Real;
OriginT: Real; DeltaT: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; Data: array of Real);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`Data: array of Real` specifies the array that contains the resulting time-domain I data relative to the carrier frequency.

GetIDataWithMean

Calculates I data and the mean value at the specified carrier frequency, from time-domain I and Q data.

Syntax

```
procedure GetIDataWithMean(Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; var Mean: Real;
Data: array of Real);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`var Mean: Real` returns the mean value of the I data relative to the carrier frequency.

`Data: array of Real` specifies the array that contains the resulting I data relative to the carrier frequency.

GetIQDataBandPass

Passes I and Q data through the band-pass filter. The filter shape factor is obtained by the MakeWeightTable function.

Syntax

```
procedure GetIQDataBandPass(wNum: LongInt; Weight: array of Real;
Center: Real; Filter: Real; Resolution: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real;
ResultI: array of Real; ResultQ: array of Real);
```

Arguments

`wNum: LongInt` specifies the number of data in the array that contains the filter shape factor.

`Weight: array of Real` specifies the array that contains the filter shape factor made by the MakeWeightTable function.

`Center: Real` sets the center frequency.

`Filter: Real` specifies the frequency of the band-pass filter.

`Resolution: Real` specifies the frequency resolution.

`Num: LongInt` specifies the number of I/Q data.

`DataI: array of Real` specifies the array that contains time-domain I data.

`DataQ: array of Real` specifies the array that contains time-domain Q data.

`ResultI: array of Real` specifies the array that contains the resulting time-domain I data after filtering.

`ResultQ: array of Real` specifies the array that contains the resulting time-domain Q data after filtering.

Related Functions

`MakeWeightTable`

GetIQFFT

Multiplies I and Q data by the window function and performs FFT.

Syntax

```
procedure GetIQFFT(Ratio: Real; Window: array of Real;
Num: LongInt; DataI: array of Real; DataQ: array of Real;
ResultI: array of Real; ResultQ: array of Real);
```

Arguments

`Ratio: Real` sets the correction factor for the window function. The value is calculated by the MakeFFTWinodowTable function.

`Window: array of Real` specifies the array that contains the window function. The data is obtained by the MakeFFTWinodowTable function.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`ResultI: array of Real` specifies the array that contains the resulting frequency-domain I data.

`ResultQ: array of Real` specifies the array that contains the resulting frequency-domain Q data.

Related Functions

`MakeFFTWinodowTable`

GetIQFFTOffset

Subtracts offset from I and Q data before multiplying a window function, and then performs FFT.

Syntax

```
procedure GetIQFFTOffset(Ratio: Real; Window: array of Real;
IOffset: Real; QOffset: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; ResultI: array of Real;
ResultQ: array of Real);
```

Arguments

`Ratio: Real` sets the correction factor for the window function. The value is calculated by the MakeFFTWinodowTable function.

`Window: array of Real` specifies the array that contains the window function. The data is obtained by the MakeFFTWinodowTable function.

`IOffset: Real` sets the I offset.

`QOffset: Real` sets the Q offset.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains the time-domain I data.

`DataQ: array of Real` specifies the array that contains the time-domain Q data.

`ResultI: array of Real` specifies the array that contains the resulting frequency-domain I data.

`ResultQ: array of Real` specifies the array that contains the resulting frequency-domain Q data.

Related Functions

`MakeFFTWinodowTable`

GetIQPData

Calculates I, Q, and phase from time-domain I and Q data.

Syntax

```
procedure GetIQPData (Center: Real; Carrier: Real; Offset: Real;
OriginT: Real; DeltaT: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; ResultI: array of Real;
ResultQ: array of Real; ResultP: array of Real;
MaskP: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`ResultI: array of Real` specifies the array that contains the resulting I data relative to the carrier frequency.

`ResultQ: array of Real` specifies the array that contains the resulting Q data relative to the carrier frequency.

`ResultP: array of Real` specifies the array that contains the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for ResultP above. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetIQPDataLevelMask

Calculates I, Q, and phase with the specified carrier from time-domain I and Q data and masks the phase data if the amplitude is less than the specified threshold.

Syntax

```
procedure GetIQPDataLevelMask(Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Threshold: Real;
Num: LongInt; DataI: array of Real; DataQ: array of Real;
ResultI: array of Real; ResultQ: array of Real;
ResultP: array of Real; MaskP: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Threshold: Real` sets the amplitude threshold in dBm for masking the phase data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`ResultI: array of Real` specifies the array that contains the resulting I data relative to the carrier frequency.

`ResultQ: array of Real` specifies the array that contains the resulting Q data relative to the carrier frequency.

`ResultP: array of Real` specifies the array that contains the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for ResultP above. The mask has two values: 0 if the amplitude is greater than the threshold and –1 if less than or equal to the threshold.

GetIQPDataLevelMaskWithMean

Calculates I, Q, phase, and the mean value of I and Q with the specified carrier from time-domain I and Q data and masks the phase data if the amplitude is less than the specified threshold.

Syntax

```
procedure GetIQPDataLevelMaskWithMean(Center: Real;
Carrier: Real; Offset: Real; OriginT: Real; DeltaT: Real;
Threshold: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; var MeanI: Real; var MeanQ: Real;
ResultI: array of Real; ResultQ: array of Real;
ResultP: array of Real; MaskP: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Threshold: Real` sets the amplitude threshold in dBm for masking the phase data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data.

`DataQ: array of Real` specifies the array that contains time-domain Q data.

`var MeanI: Real` returns the mean value of the I data relative to the carrier frequency.

`var MeanQ: Real` returns the mean value of the Q data relative to the carrier frequency.

`ResultI: array of Real` specifies the array that contains the resulting I data relative to the carrier frequency.

`ResultQ: array of Real` specifies the array that contains the resulting Q data relative to the carrier frequency.

`ResultP: array of Real` specifies the array that contains the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for ResultP above. The mask has two values: 0 if the amplitude is greater than the threshold and –1 if less than or equal to the threshold.

GetIQPDataWithMean

Calculates I, Q, phase, and the mean value of I and Q with the specified carrier from time-domain I and Q data.

Syntax

```
procedure GetIQPDataWithMean(Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; var MeanI: Real;
var MeanQ: Real; ResultI: array of Real; ResultQ: array of Real;
ResultP: array of Real; MaskP: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`var MeanI: Real` returns the mean value of the I data relative to the carrier frequency.

`var MeanQ: Real` returns the mean value of the Q data relative to the carrier frequency.

`ResultI: array of Real` specifies the array that contains the resulting I data relative to the carrier frequency.

`ResultQ: array of Real` specifies the array that contains the resulting Q data relative to the carrier frequency.

`ResultP: array of Real` specifies the array that contains the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for ResultP above. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetPhaseData

Calculates phase with the specified carrier from time-domain I and Q data.

Syntax

```
procedure GetPhaseData (Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; Data: array of Real;
Mask: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`Data: array of Real` specifies the array that will store the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for the phase data just above. The mask has two values: 0 for valid data and –1 for invalid data. For example, when the I/Q data is zero and not calculable, the value of –1 returns.

GetPhaseDataLevelMask

Calculates phase with the specified carrier from time-domain I and Q data and masks the phase data if the amplitude is less than the specified threshold.

Syntax

```
procedure GetPhaseDataLevelMask(Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Threshold: Real;
Num: LongInt; DataI: array of Real; DataQ: array of Real;
Data: array of Real; Mask: array of LongInt);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Threshold: Real` sets the amplitude threshold in dBm for masking the phase data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`Data: array of Real` specifies the array that will store the resulting phase data relative to the carrier frequency.

`MaskP: array of longInt` specifies the array that contains the mask data for the phase data just above. The mask has two values: 0 if the amplitude is greater than the threshold and –1 if less than or equal to the threshold.

GetQData

Calculates Q data with the specified carrier from time-domain I and Q data.

Syntax

```
procedure GetQData (Center: Real; Carrier: Real; Offset: Real;
OriginT: Real; DeltaT: Real; Num: LongInt; DataI: array of Real;
DataQ: array of Real; Data: array of Real);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`Data: array of Real` specifies the array that will store the resulting Q data relative to the carrier frequency.

GetQDataWithMean

Calculates Q data and the mean value with the specified carrier from time-domain I and Q data.

Syntax

```
procedure GetQDataWithMean(Center: Real; Carrier: Real;
Offset: Real; OriginT: Real; DeltaT: Real; Num: LongInt;
DataI: array of Real; DataQ: array of Real; var Mean: Real;
Data: array of Real);
```

Arguments

`Center: Real` sets the center frequency.

`Carrier: Real` sets the carrier frequency.

`Offset: Real` sets the phase offset ranging from –180° to 180°.

`OriginT: Real` sets the relative time from the reference frame.

`DeltaT: Real` specifies the time interval of the data.

`Num: LongInt` specifies the number of the I/Q data pairs.

`DataI: array of Real` specifies the array that contains time-domain I data relative to the center frequency.

`DataQ: array of Real` specifies the array that contains time-domain Q data relative to the center frequency.

`var Mean: Real` returns the mean value of the Q data relative to the carrier frequency.

`Data: array of Real` specifies the array that will store the resulting Q data relative to the carrier frequency.

GetWeightTableSize

Returns the array size to store the weighting factor for the filter. The returned value is used to determine if the MakeWeightTable function can be called for the array.

Syntax

```
function GetWeightTableSize(Filter: String; Alpha: Real;
RBW: Real; Resolution: Real): LongInt;
```

Arguments

`Filter: String` selects the filter from `Gaussian`, `Nyquist`, `RootNyquist`, and `Rect`.

`Alpha: Real` sets the roll-off rate when the filter is `Nyquist` or `RootNyquist`. The range is 0 to 1.

`RBW: Real` sets the bandwidth of the filter.

`Resolution: Real` sets the frequency resolution.

Returns

0 if any error occurred. RBW < (5 × Resolution) results in error.

MakeFFTWindowTable

Calculates the correction factor for the periodic FFT window function ("periodic" is defined as W[i] = W[N–i]).

Syntax

```
function MakeFFTWindowTable(Window: String; BHType: String;
Alpha: Real; TableSize: LongInt; WindowTable: array of Real):
Real;
```

Arguments

`Window: String` selects the window function from `Rect`, `Parzen`, `Bartlett`, `Welch`, `SineLobe`, `Hanning`, `SineCubed`, `SineToThe4th`, `Hamming`, `Blackman`, `BlackmanHarris`, `GaussianCurve`, and `KaiserBessel`.

`BHType: String` specifies the number of sanpling points and the window type when the window is `BlackmanHarris`: `3sampleA`, `3sampleB`, `4sampleA`, or `4sampleB`.

`Alpha: Real` sets the factor when the window is `GaussianCurve` or `KaiserBessel`. The value is greater than or equal to zero. Zero means the rectangular window. The value of less than five is realistic.

`WindowTable: array of Real` specifies the array that contains the window function.

Returns

The correction factor. For example, the value is zero for Rect and about 0.5 for Parzen. Divide the FFT-processed data by this correction factor to obtain the frequency domain data.

MakeWeightTable

Makes the weighting factor array and returns the number of valid array elements. As the weighting factor is symmetrical, only the right half is returned with the center of WeightTable[0].

Syntax

```
function MakeWeightTable(Filter: String; Alpha: Real;
RBW: Real; Resolution: Real; TableSize: LongInt;
WeightTable: array of Real): LongInt;
```

Arguments

`Filter: String` selects the filter from `Gaussian`, `Nyquist`, `RootNyquist`, and `Rect`.

`Alpha: Real` sets the roll-off rate when the filter is `Nyquist` or `RootNyquist`. The range is 0 to 1.

`RBW: Real` sets the bandwidth of the filter.

`Resolution: Real` sets the resolution for the frequency data.

`TableSize: LongInt` sets the size of the WeightTable array below.

`WeightTable: array of Real` specifies the array that will store the resulting weighting factor for the filter.

Returns

The number of valid elements of the WeightTable array. If it is greater than TableSize, the error code “0” returns.

Miscellaneous Functions

Ceil

Rounds up a numeric value toward positive infinity.

Syntax `function Ceil(r: Real): Longint;`

Arguments `r: Real` specifies the value.

ChangeFileExt

Changes the file extension.

Syntax

```
function ChangeFileExt(FileName: String; Extension: String):
String;
```

Arguments `FileName: String` specifies the file name.

`Extension: String` specifies the file extension.

CharToInt

Converts the head character of the string to the numeric value from 0 to 255.

Syntax `function CharToInt(s: String): LongInt;`

Arguments `s: String` specifies the string to be converted.

Returns The converted numeric value.

ClearText

Clears output from functions such as writeln().

Syntax `procedure ClearText;`

CompareText

Compares two strings without distinguishing between upper and lower cases.

Syntax

```
function CompareText(s: String; t: String): LongInt;
```

Arguments `s: String` and `t: String` specifies the strings to be compared.

Returns Zero if two strings are identical.

CopyLongIntArray

Copies a long integer-type array.

Syntax

```
procedure CopyLongIntArray(Num: LongInt;
Source: array of longInt; Dest: array of LongInt);
```

Arguments `Num: LongInt` specifies the array size.

`Source: array of longInt` specifies the source array.

`Dest: array of LongInt` specifies the destination array.

CopyRealArray

Copies a real-type array.

Syntax

```
procedure CopyRealArray(Num: LongInt; Source: array of Real;
Dest: array of Real);
```

Arguments `Num: LongInt` specifies the array size.

`Source: array of Real` specifies the source array.

`Dest: array of Real` specifies the destination array.

CutStr

Cuts a string out of the original string between the beginning and the specified delimiter. The delimiter is discarded.

Syntax `function CutStr(var s: String; t: String): String;`

Arguments `var s: String` specifies the original string. The delimiter is not included in the returned string.

`t: String` specifies a delimiter.

Returns The cut out string.

EngToFloat

Converts a string including the SI unit (G, M, k, m, u, or n) to the numeric value.

Syntax `function EngToFloat (s: String): Real;`

Arguments `s: String` specifies the string to be converted.

Returns The converted numeric value. If the conversion is impossible, zero returns.

EngToFloatDef

Converts a string including the SI unit (G, M, k, m, u, or n) into a numeric value.

Syntax `function EngToFloatDef(s: String; def: Real): Real;`

Arguments `s: String` specifies a string to be converted

`def: Real` specifies the numeric value returned if the conversion is impossible.

Returns The converted value.

ExtractFileName

Returns the file name with extension (excluding the path).

Syntax `function ExtractFileName(FileName: String): String;`

Arguments `FileName: String` specifies the file name.

Returns The file name with extension (excluding the path).

ExtractFilePath

Returns the file path.

Syntax `function ExtractFilePath(FileName: String): String;`

Arguments `FileName: String` specifies the file name.

Returns The file path.

FileExists

Checks if a specified file exists.

Syntax `function FileExists(FileName: String): Boolean;`

Arguments `FileName: String` specifies a file path.

Returns `True` indicates the specified file exists.

`False` indicates the specified file does not exist.

FloatToEng

Converts a numeric value to the string including the SI unit (G, M, k, m, u, or n).

Syntax `function FloatToEng(Value: Real): String;`

Arguments `Value: Real` specifies the value to be converted.

Returns The converted string.

FloatToStr

Converts a numeric value to the string.

Syntax `function FloatToStr(Value: Real): String;`

Arguments `Value: Real` specifies the value to be converted.

Returns The converted string.

Floor

Rounds down a numeric value toward negative infinity.

Syntax `function Floor(r: Real): Longint;`

Arguments `r: Real` specifies the value.

FormatEng

Converts a numeric value to the string including the SI unit (G, M, k, m, u, or n).

Syntax

```
function FormatEng(fmt: String; Value: Real): String;
```

Arguments

`fmt: String` specifies the number of digits as '0.###'.
When you do not specify the number of digits, specify space characters.

`Value: Real` specifies the value to be converted.

Returns

The converted string.

FormatFloat

Converts a numeric value to the string.

Syntax

```
procedure FormatFloat(fmt: String; Value: Real): String;
```

Arguments

`fmt: String` specifies the number of digits like '0.###'.
When you do not specify the number of digits, specify space characters.

`Value: Real` specifies the value to be converted.

Returns

The converted string.

GetExePath

Returns a path that includes a script executable file.

Syntax

```
function GetExePath: String;
```

Returns

The file path. For example:

```
C:\PROGRAM FILES\SONY TEKTRONIX\WCA\BIN\
```

GetMinMaxMean

Calculates the maximum, minimum, and mean values of data.

Syntax

```
procedure GetMinMaxMean(Num: Longint; Data: array of Real;
var Min: Real; var Max: Real; var Mean: Real);
```

Arguments

`Num: Longint` specifies the number of data.

`Data: array of Real` specifies the data array.

`var Min: Real` returns the minimum value.

`var Max: Real` returns the maximum value.

`var Mean: Real` returns the mean value.

GetMinMaxMeanRegion

Calculates the maximum, minimum, and mean values of data in the specified range.

Syntax

```
procedure GetMinMaxMeanRegion(DStart: LongInt; DEnd: LongInt;
Num: Longint; Data: array of Real; var Min: Real; var Max: Real;
var Mean: Real);
```

Arguments

`DStart: LongInt` specifies the start position in the calculation range.

`DEnd: LongInt` specifies the end position in the calculation range.

`Num: Longint` specifies the number of data.

`Data: array of Real` specifies the data array.

`var Min: Real` returns the minimum value.

`var Max: Real` returns the maximum value.

`var Mean: Real` returns the mean value.

GetMinMaxMeanRegionWithMask

Calculates the maximum, minimum, and mean value of data not masked in the specified range.

Syntax

```
procedure GetMinMaxMeanRegionWithMask(DStart: LongInt;
DEnd: LongInt; Num: Longint; Data: array of Real;
Mask: array of LongInt; var Min: Real; var Max: Real;
var Mean: Real);
```

Arguments

`DStart: LongInt` specifies the start position in the calculation range.

`DEnd: LongInt` specifies the end position in the calculation range.

`Num: Longint` specifies the number of data.

`Data: array of Real` specifies the data array.

`Mask: array of LongInt` specifies the mask array for the data array just above. The mask value is 0 for valid data and –1 for invalid data. Set the value for each data.

`var Min: Real` returns the minimum value.

`var Max: Real` returns the maximum value.

`var Mean: Real` returns the mean value.

GetMinMaxMeanWithMask

Calculates the maximum, minimum, and mean values of data not masked.

Syntax

```
procedure GetMinMaxMeanWithMask(Num: Longint;
Data: array of Real; Mask: array of LongInt; var Min: Real;
var Max: Real; var Mean: Real);
```

Arguments

`Num: Longint` specifies the number of data.

`Data: array of Real` specifies the data array.

`Mask: array of LongInt` specifies the mask array for the data array just above. The mask value is 0 for valid data and –1 for invalid data. Set the value for each data.

`var Min: Real` returns the minimum value.

`var Max: Real` returns the maximum value.

`var Mean: Real` returns the mean value.

GetRunState

Returns the run status immediately before calling the main procedure or a procedure preceded with `Sleep:`. Although the run status is restored immediately after the procedure is completed, the SetRunState procedure allows you to change the status.

Syntax

```
function GetRunState: Longint;
```

Returns

The run status code. The meanings are also shown below:

RUN_IDLE	= 0;	The data acquisition stopped.
RUN_ROLL	= 1;	The analyzer operated in the Roll mode.
RUN_BLOCK	= 2	The analyzer operated in the Block mode.

Related Functions

`SetRunState`

GetSmooth

Smooths data not masked.

Syntax

```
procedure GetSmooth(Level: LongInt; Num: Longint;
Data: array of Real; Mask: array of LongInt;
ResultS: array of Real; MaskS: array of LongInt);
```

Arguments

`Level: LongInt` sets the smoothing level:

Level=0 outputs the raw data without smoothing.

Level ≥ 1 averages [1 + 2^(Level–1)] data.

`Num: Longint` specifies the number of data.

`Data: array of Real` specifies the array that contains the data.

`Mask: array of LongInt` specifies the mask array for the data array just above. The mask value is 0 for valid data and –1 for invalid data. Set the value for each data.

`ResultS: array of Real` returns the smoothed data.

`MaskS: array of LongInt` specifies the array that returns the mask data for the smoothed data.

GetViewName

Returns the name of the View that started a script.

Syntax

```
function GetViewName: String;
```

IntToHex

Converts a value to the corresponding hexadecimal representation.

Syntax

```
function IntToHex(Value: Longint; Digits: Longint): String;
```

Arguments

`Value: Longint` specifies a value to be converted.

`Digits: Longint` indicates the minimum number of digits of the hexadecimal representation.

MakePointsArray

Makes an integer array with an equal interval between two adjacent values. For example, the integer array is used to index another array data. See Figure 2–2 as an example.

Syntax

```
procedure MakePointsArray(DStart: LongInt; DNum: LongInt;
Num: LongInt; Points: array of LongInt);
```

Arguments

`DStart: LongInt` specifies the first integer value.

`DNum: LongInt` = (The last integer value) – (DStart) +1.

`Num: LongInt` is the array size.

`Points: array of LongInt` specifies the integer array.

The interval between two adjacent values is taken as equal as possible.

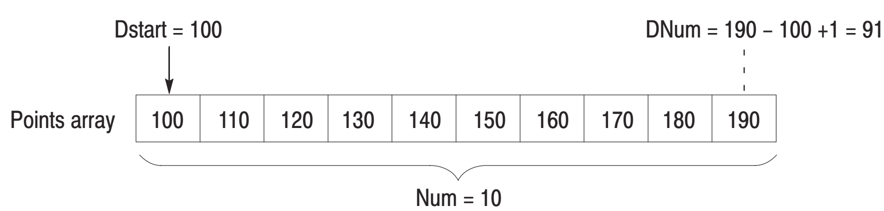


Figure 2–2: Example of using MakePointsArray

ScriptMessageBox

Displays a message box.

Syntax

```
function ScriptMessageBox(Text: String; Caption: String;
Flags: LongInt): LongInt;
```

Arguments

`Text: String` specifies the string to be displayed in the message box.

`Caption: String` specifies the title of the message box.

`Flags: LongInt` specifies the buttons on the message box as follows:

Flags	Buttons added
MB_OK	[OK] button
MB_OKCANCEL	{OK} and [Cancel] buttons
MB_RETRYCANCEL	[Retry] and [Cancel] buttons
MB_YESNO	[Yes] and [No] buttons
MB_YESNOCANCEL	[Yes], [No], and [Cancel] buttons

Returns

The event code as follows:

Event code	Meaning
IDOK = 1	[OK] has been selected.
IDCANCEL = 2	[Cancel] has been selected.
IDABORT = 3	[Stop] has been selected.
IDRETRY = 4	[Retry] has been selected.
IDIGNORE = 5	[Ignore] has been selected.
IDYES = 6	[Yes] has been selected.
IDNO = 7	[No] has been selected.

SetRunState

Sets the run state of the main procedure or a procedure preceded with `Sleep:` immediately after it was invoked.

Syntax `procedure SetRunState(state: Longint);`

Arguments `state: Longint` as follows:

RUN_IDLE = 0;
RUN_ROLL = 1;
RUN_BLOCK = 2

SetStatusMessage

Sets the string shown in the status display area of the script view.

Syntax `procedure SetStatusMessage(s: String);`

Arguments `s: String` specifies the string.

SetTitle

Sets a string that appears at the top of the script menu.

Syntax `procedure SetTitle(s: String);`

SetViewTitle

Sets the string shown at the top of the script view.

Syntax `procedure SetViewTitle(s: String);`

Arguments `s: String` specifies the string.

StrToIntDef

Converts a string to the numeric value.

Syntax `function StrToIntDef(s: String; def: LongInt): LongInt;`

Arguments `s: String` specifies the string to be converted.

`def: LongInt` defines the returned value when the string can not be converted.

Returns The converted value.

Terminated

Determines whether a processing is forcefully terminated with the **Break** side key. Check the returned value of this function to exit a long running loop.

Syntax `function Terminated: Boolean;`

Returns `True` when a processing is forcefully terminated using the Break side key.

Trim

Deletes spaces at the head and end of the string.

Syntax `function Trim(s: String): String;`

Arguments `s: String` specifies the original string.

Returns The string that has no space at the head and end.

WaitMilliSecond

Waits in millisecond.

Syntax

```
procedure WaitMilliSecond(wait: Longint);
```

Arguments

`wait: Longint` sets the waiting time in ms.

Part II Using Command

Getting Started on Command

Getting Started on Command

You can write computer programs that remotely change the analyzer front panel controls, take and store measurements for further analysis, or read those measurements, using the GPIB or TCP/IP Ethernet (optional) interface.

To help you get started with programming the analyzer, this section includes the following sections:

- *Overview of the Manual* – summarizes the type of programming information contained in each major section of this manual.
- *Setting Up Remote Communications* – describes how to physically connect the analyzer to a controller and set the appropriate front panel controls.

Overview

The information contained in each major section is described below.

Syntax and Commands

The *Syntax and Commands* chapter describes the structure and content of the messages your program sends to the analyzer, Figure 3–1 shows command parts as described in the *Command Syntax* subsection.

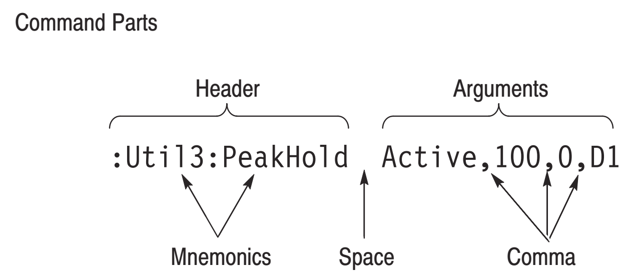


Figure 3–1: Common message elements

Chapter 2 also describes the effect of each command and provides examples of how you might use it. The *Command Groups* section provides a list by functional area. The command description sections starting from the *Configuration Commands* on page 4–53 arrange commands alphabetically for each command group (see Figure 3–2).

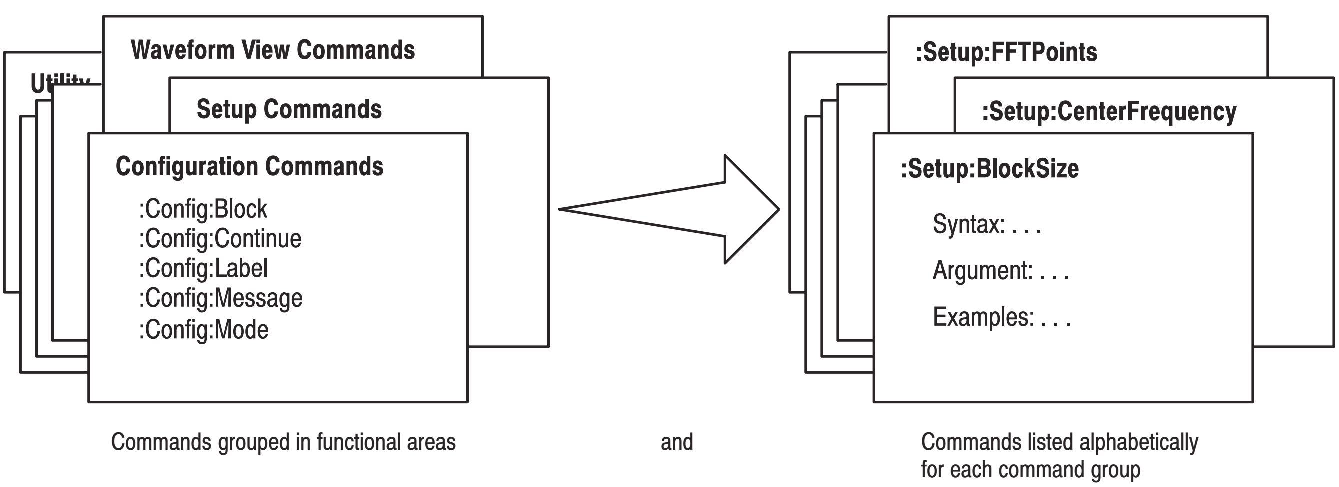


Figure 3–2: Functional groupings and an alphabetical list of commands

Status and Events

The program may request information from the analyzer. The analyzer provides information in the form of status and error messages. Figure 3–3 illustrates the basic operation of this system.

The *Status and Events* chapter starting on page 5–1 describes how to obtain status and event messages in your programs.

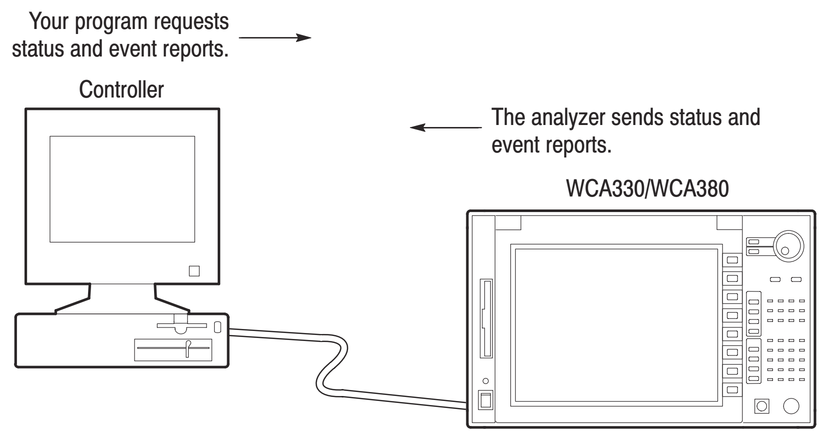


Figure 3–3: Event (interrupt) driven programs

Programming Examples

The *Programming Examples* chapter starting on page 6–1 describes some example programs of the PERL source codes (PERL is a freeware).

```
#
# Main
#
tcp_client(S, '', '3000');
tcp_client(SS, '', '3001');
print S "NewLine LF\n";
print S "Config:Pause On\n";
print S "Config:View1 Waveform\n";
print S "View1:Trace2:Source D1\n";
```

Figure 3–4: Example programs of the PERL source codes

Connecting the Interface

The analyzer has IBM PC based expansion slots on its rear panel, as shown in Figure 3–5. The analyzer is shipped standard with a GPIB port. In addition, the analyzer can be configured with optional networking cards.

- **GPIB interface**
  This connector has a D-type shell and conforms to IEEE Std 488.1–1987 (The cable is available from Tektronix as part number 012–0991–xx).
- **Ethernet interface**
  You can connect the analyzer directly to an Ethernet network using a twisted pair (10BASE-T/100BASE-TX) cable.

***NOTE**. Only qualified service personnel can install the network card. Contact Tektronix for details.*

You can also connect the following peripheral devices to the analyzer. Refer to the *User Manual* for information on using these interfaces.

- Mouse and keyboard ports
- Parallel port for connecting a printer
- VGA output port for connecting a monitor

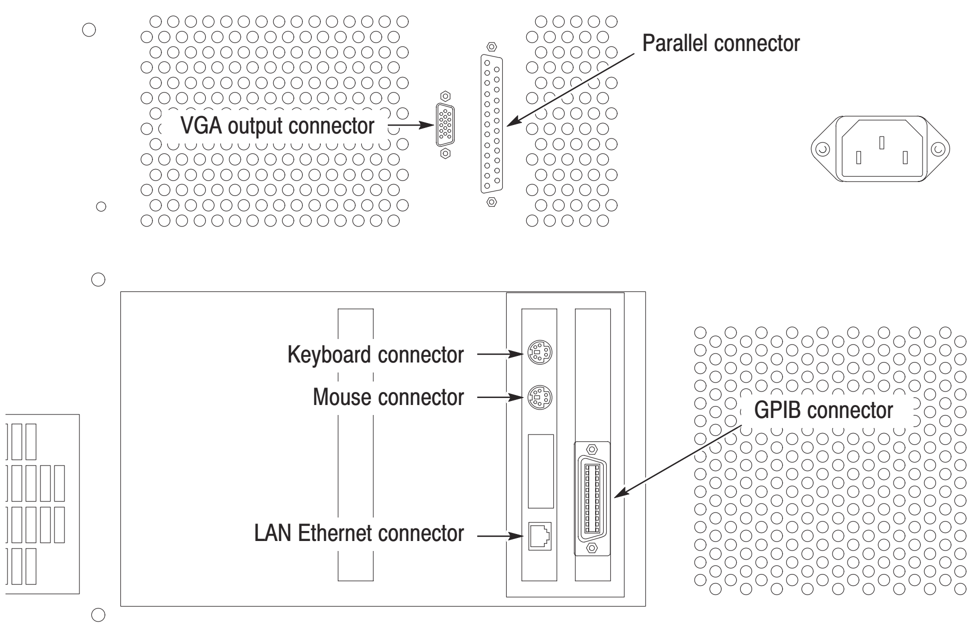


Figure 3–5: Interface connector location on the rear panel

Using Communication Ports

The analyzer can communicate with the external controller or other devices on the GPIB and Ethernet network.

Using the GPIB

The analyzer has the Talker/Listener and Controller functions. Refer to the next section for details about setting the parameters for the port.

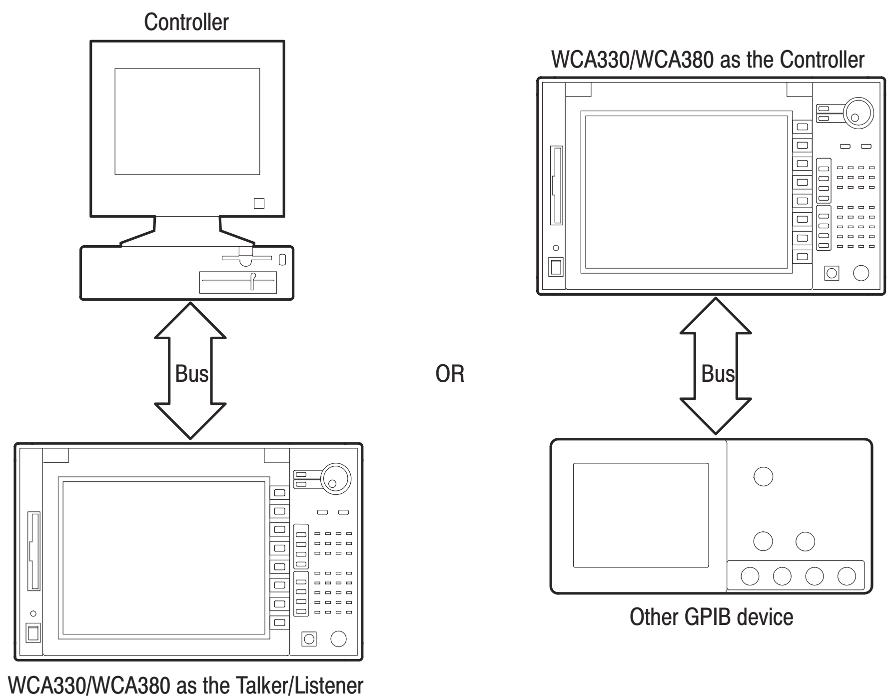


Figure 3–6: GPIB connection

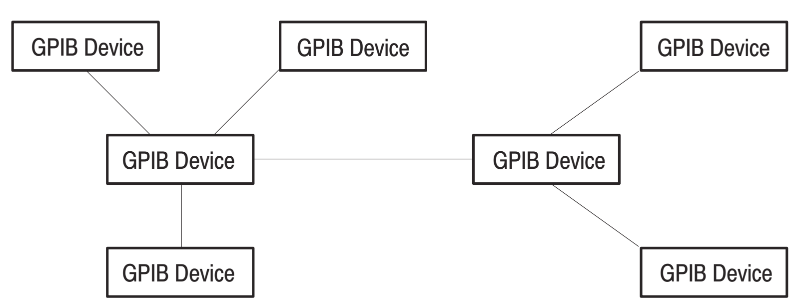


Figure 3–7: Typical GPIB network configurations

GPIB Requirements. Observe these rules when you use your analyzer with a GPIB network:

- Assign a unique device address to each device on the bus. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices to a bus.
- Turn on at least two-thirds of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration. Do not use loop or parallel configurations (see Figure 3–7).

Using TCP/IP

The analyzer communicates with the external devices through the following two ports on the TCP/IP Ethernet network:

- **Command port** receives commands from, or returns responses to the external controller.
- **Event port** sends events to the external controller.

Refer to the next section for details about setting the parameters for these ports.

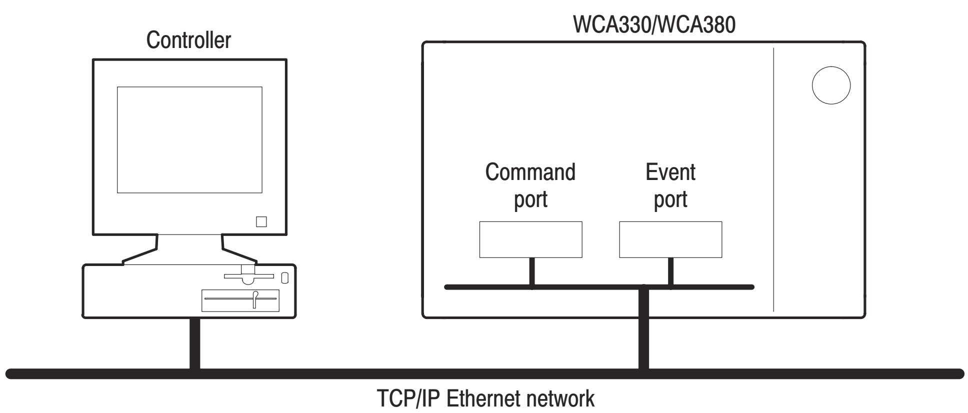


Figure 3–8: Ethernet connection

Setting Communication Parameters from the Front Panel

You can set the communication parameters from the front panel menu or with the Remote commands. This section describes the front panel operation procedures. For the remote control commands, refer to *Remote Commands* on page 4–473.

1. Press the CONFIG:**UTILITY** key to display the Utility menu.
2. Press the **More...** side key.
3. Press the **Util H** side key to display the Remote menu. See Figure 3–9.

- If you use the GPIB interface, go to the next section *Setting the GPIB Parameters* on page 3–8.
- If you use the TCP/IP interface, go to the section *Setting the TCP/IP Parameters* on page 3–9.

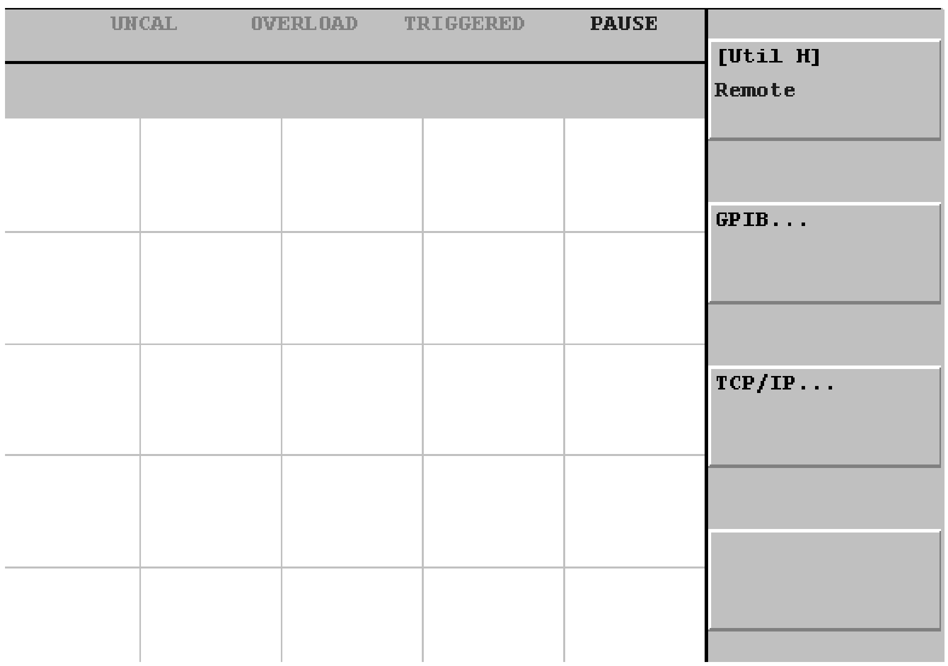


Figure 3–9: Setting the communication parameters

Setting the GPIB Parameters

You need to set the GPIB parameters of the analyzer to match the configuration of the bus. Once you have set these parameters, you can control the analyzer through the GPIB interface.

1. Press the **GPIB** side key to display the GPIB configuration menu. See Figure 3–10.

2. Press the **Interface** side key, and select **Talker/Listener** using either the general purpose knob or the keypad.

3. Press the **Primary Address** side key, and set GPIB address using either the general purpose knob or the keypad.

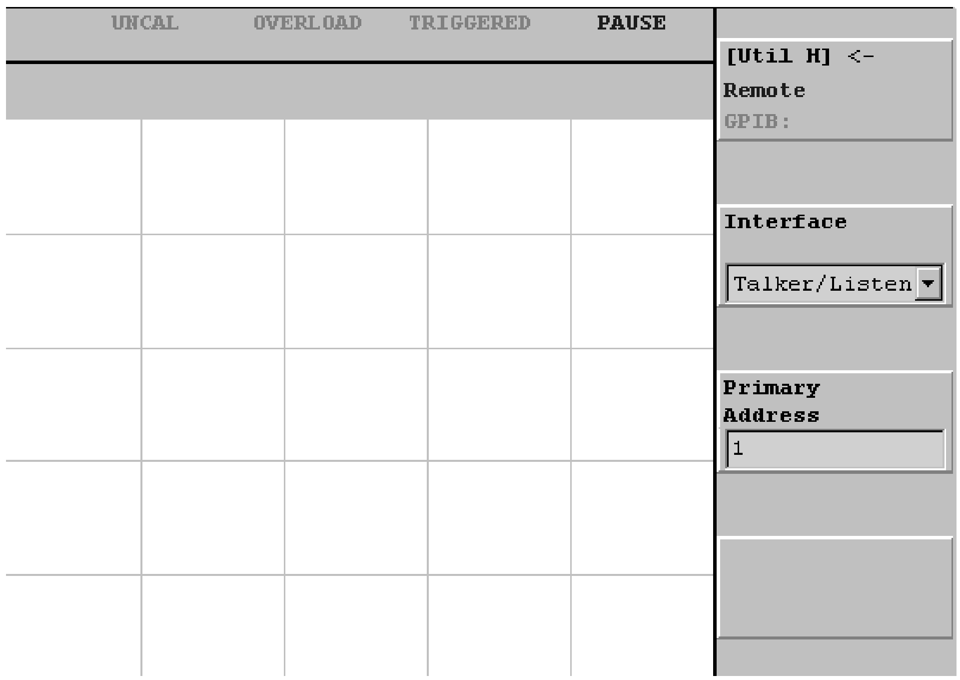


Figure 3–10: Setting the GPIB parameters

The analyzer is set up for bidirectional communication with your controller. If you wish to isolate the analyzer from the bus:

- Press the **Interface** side key, and select **Off** using either the general purpose knob or the keypad.

Setting the TCP/IP Parameters

You need to set the TCP/IP parameters of the analyzer to match the network configuration. Once you have set these parameters, you can control the analyzer on the Ethernet network.

1. Press the **TCP/IP** side key to display the TCP/IP configuration menu. See Figure 3–11.
2. Press the **Command Port** side key, and set the command port number using either the general purpose knob or the keypad. The number ranges 1024 to 32767.
3. Press the **Event Port** side key, and set the event port number using either the general purpose knob or the keypad. The number ranges 1024 to 32767. It must be different from the command port number.
4. Press the **New Line** side key, and select the new-line character of returned value using either the general purpose knob or the keypad.

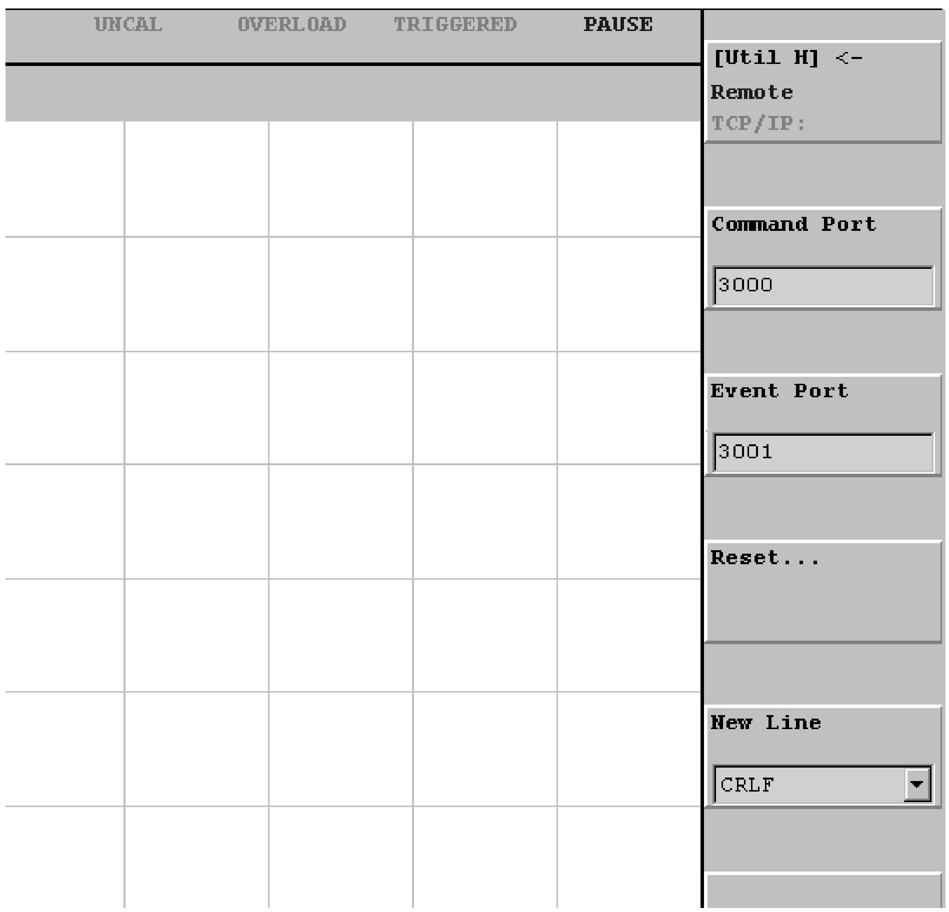


Figure 3–11: Setting the TCP/IP parameters

Syntax and Commands

Command Syntax

This section contains general information on command structure and syntax usage. You should familiarize yourself with this material before using the analyzer command descriptions.

This manual describes commands and queries using the Backus-Naur Form (BNF) notation. Table 4–1 defines the standard BNF symbols.

Table 4–1: BNF symbols and meanings

Symbol	Meaning
< >	Defined element
::=	Is defined as
|	Exclusive OR
{ }	Group; one element is required
[ ]	Optional; can be omitted
. . .	Previous element(s) may be repeated
( )	Comment

Commands and Queries

The analyzer commands are based on a hierarchical or tree structure (see Figure 4–1) that represents a subsystem. The top level of the tree is the root node; it is followed by one or more lower-level nodes.

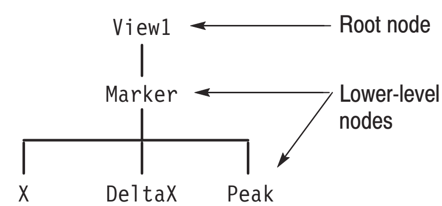


Figure 4–1: Example of subsystem hierarchy tree

You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

Creating Commands

The analyzer commands are created by stringing together the nodes of a subsystem hierarchy and separating each node by a colon (:).

In Figure 4–1, View1 is the root node and Marker, X, DeltaX, and Peak are lower-level nodes. To create a command, start with the root node View1 and move down the tree structure adding nodes until you reach the end of a branch. Most commands and some queries have parameters; you must include a value for these parameters. The command descriptions, which start on page 4–7, list the valid values for all parameters.

For example, `:View1:Marker:X 1.5` is a valid command created from the hierarchy tree in Figure 4–1.

***NOTE**. If you specify a parameter value that is out of range, the parameter will be set to a default value.*

Creating Queries

To create a query, start at the root node of a tree structure, move down to the end of a branch, and add a question mark. `:View1:Marker:X?` is an example of a valid query using the hierarchy tree in Figure 4–1.

Headers in Query Responses

You can control whether the analyzer returns headers as part of the query response. Use the `[:Util8]:Header` command (page 4–484) to control this feature. If header is on, the query response returns command headers and formats itself as a valid set command. When header is off, the response includes only the values. This may make it easier to parse and extract the information from the response. Table 4–2 shows the difference in responses.

Table 4–2: Comparison of header off and on responses

Query	Header off response	Header on response
`:View1:Source?`	`"FILE1.AP"`	`:View1:Source "File1.AP"`
`:Setup:Span?`	`5M`	`:Setup:Span 5M`

Parameter Types

Parameters are indicated by angle brackets, such as <file_name>. There are several different types of parameters, as listed in Table 4–3. The parameter type is listed after the parameter. Some parameter types are defined specifically for the analyzer command set and some are defined by ANSI/IEEE 488.2-1987.

Table 4–3: Parameter types used in syntax descriptions

Parameter Type	Description	Example
boolean	Boolean numbers or values	On, Off
discrete	A list of specific values	MIN, MAX
NR1 [1] numeric	Integers	0, 1, 15, –1
NR2 [1] numeric	Decimal numbers	1.2, 3.141516, –6.5
NR3 [1] numeric	Floating point numbers	3.1415E–9, –16.1E5
NRf [1] numeric	Flexible decimal number that may be type NR1, NR2, or NR3	See NR1, NR2, NR3 examples
string [2]	Alphanumeric characters (must be within quotation marks)	"Testing 1, 2, 3"

1 **An ANSI/IEEE 488.2-1992-defined parameter type.**

2 **Defined in ANSI/IEEE 488.2 as "String Response Data" and "String Program Data."**

Special Characters

The Line Feed (LF) character (ASCII 10) and all characters in the range of ASCII 127-255 are defined as special characters. Using these characters in any command yields unpredictable results.

Abbreviating Commands, Queries, and Parameters

You must not abbreviate any commands, queries, or parameters. For example, shortening the command `:Setup:CenterFrequency` to `:Setup:CenterFreq` causes an error.

Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, then add a semicolon (;), and finally add more commands or queries and semicolons until you are done. If the command following a semicolon is a root node, precede it with a colon (:). Figure 4–2 illustrates a chained message consisting of several commands and queries. The chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

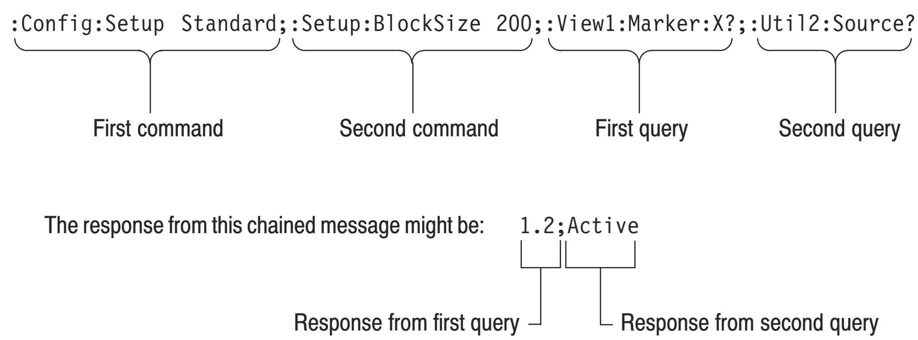


Figure 4–2: Example of chaining commands and queries

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes. In Figure 4–3, the second command has the same root node (Marker) as the first command, so these nodes can be omitted.

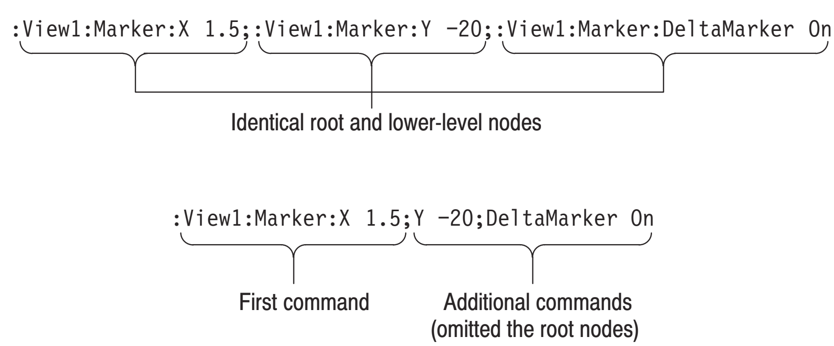


Figure 4–3: Example of omitting root and lower-level nodes in a chained message

Unit and SI Prefix

If a decimal numeric argument refers to amplitude, frequency, or time, you can express it using SI units instead of the scaled explicit point input value format <NR3>. (SI units conform to the Systeme International d'Unites standard.) For example, you can use the input format 200 m or 1.0 M instead of 200.0E–3 or 1.0E+6, respectively, to specify the argument.

***NOTE**. You must omit the unit, such as Hz and dB. For example, 5 MHz of frequency is represented by "5M" in a command.*

The SI prefixes, which must be included, are shown below. Note that either lower or upper case prefixes can be used.

SI prefix	p/P	n/N	u/U	m/M	k/K	m/M	g/G
Corresponding power	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{3}	10^{6}	10^{9}

*** Note that the prefix m/M indicates 10^{-3} when the decimal numeric argument denotes amplitude or time, but 10^{6} when it denotes frequency.**

General Rules

Here are three general rules for using the analyzer commands, queries, and parameters:

- You must use double (" ") quotation marks for quoted strings.

  correct: "This string uses quotation marks correctly."

  incorrect: 'This string also uses quotation marks correctly.'

  incorrect: "This string does not use quotation marks correctly.'

- You can use upper case, lower case, or a mixture of both cases for all commands, queries, and parameters.

  ```
  :VIEW1:MASK:RBW1M:FREQUENCY 25M
  ```

  is the same as

  ```
  :view1:mask:rbw1m:frequency 25m
  ```

  and

  ```
  :view1:mask:rbw1m:FREQUENCY 25M
  ```

- No embedded spaces are allowed between or within nodes.

  correct: `:VIEW1:MASK:RBW1M:FREQUENCY 25M`

  incorrect: `:VIEW1: MASK: RBW1M: FREQ UENCY 25M`

Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a view mnemonic can be either `View1`, `View2`, ... or `View8`. You use these mnemonics in the command just as you do any other mnemonic. For example, there is a `View1:Format` command, and there is also a `View2:Format` command. In the command descriptions, this list of choices is abbreviated as `View<x>`.

Table 4–4: Constructed mnemonics

Symbol	Meaning
View<x>	A view specifier where <x> = 1 to 8, corresponding to A – H
Utility<x>	A utility specifier where <x> = 1 to 8, corresponding to A – H
Dev<x>	A GPIB device where <x> is the GPIB address
Result<x>	A result specifier where <x> depends on commands, starting with 1.
<address>	GPIB address, ranging 1 to 30
<command>	IEEE 488.2 command e.g. *IDN
<message>	Specify the message for the command
<item>	Specify the item for the command

Command Groups

The following sections list the WCA330 and WCA380 spectrum analyzer commands in two ways. They first present them by functional groups, then list them alphabetically. The functional group lists start below. The alphabetical lists provide more detail on each command and start on page 4–53.

Items followed by question marks are queries; items without question marks are commands. Some items in this section have a question mark in parentheses (?) in the command header section; this indicates that the item can be both a command and a query.

Functional Groups

There are four main groups:

- **Configuration** command group configures the analyzer for the specific measurement. Also, it starts or stops data acquisition.
- **Setup** command group sets up the analyzer hardware, such as frequency, span, trigger, and memory.
- **View** command group displays waveform and performs measurement.
- **Utility** command group controls the utilities, such as self calibration, file management, averaging, and remote operation.

Table 4–5 lists the functional groups and subgroups of the commands. The analyzer has its system program modules corresponding to each subgroup.

Table 4–5: Functional groups in the command set

Group	Subgroup	Description
Configuration	–	Configure the analyzer and start/stop acquisition
Setup	Standard	Set up the analyzer for the standard measurement
	CDMA	Set up the analyzer for the CDMA measurement
View	Waveform	Control the Waveform view
	Analog	Control the Analog view
	FSK	Control the FSK view
	Spectrogram	Control the Spectrogram view
	Waterfall	Control the Waterfall view
	Polar	Control the Polar view
	Eye diagram	Control the Eye diagram view
	Symbol table	Control the Symbol table view
	EVM	Control the EVM view
	CDMA Waveform	Control the CDMA Waveform view
	CDMA Polar	Control the CDMA Polar view
	CDMA Time	Control the CDMA Time view

Table 4–5: Functional groups in the command set (Cont.)

Group	Subgroup	Description
View (Cont.)	CodeSpectrogram	Control the code-domain spectrogram view (cdmaOne standard)
	CodePolar	Control the polar view (cdmaOne standard)
	CodePower	Control the code-domain power view (cdmaOne standard)
	CodeWSpectrogram	Control the code-domain spectrogram view (W-CDMA standard)
	CodeWPolar	Control the polar view (W-CDMA standard)
	CodeWPower	Control the code-domain power view (W-CDMA standard)
	3gppACPView	Control the ACP measurement according to the 3GPP standard.
	3gppSpectrogram	Control the code-domain spectrogram view (3GPP standard)
	3gppPolar	Control the polar view (3GPP standard)
	3gppPower	Control the code-domain power view (3GPP standard)
	GSM	Control the GSM measurement
	GSMPolar	Control the polar view (GSM standard)
	GSMMask	Control the Analog view with the GSM mask (GSM standard)
	CCDF	Control the CCDF measurement
	CCDFView	Control the CCDF view
Utility	Self gain-calibration	Control the self gain-calibration
	Save/Load	Control to save and load data to/from files
	Average	Control averaging
	Remote	Control remote operation

General Programming Sequence

The programs for controlling the analyzer have the following steps in general:

1. *Configuration:* Use the Configuration command group to set the basic operation mode and assign the Views.

   Also, use the Remote command subgroup (Util8) to set the communication parameters with other device.

2. *Setup:* Use the Setup command group to set the hardware, such as frequency, span, trigger, and memory.

3. *Acquisition:* Use the Configuration command group to start or stop the waveform acquisition.

   You can load previously acquired data with the Save/Load command subgroup (Util2).

4. *Measurement and Display:* Use the View command group to measure and display the result. For example, if you have selected the Spectrogram View in the configuration, use the Spectrogram View command subgroup.

   You can use the Average command subgroup (Util3) to average waveforms. Also, you can save the data to a file with the Save/Load command subgroup (Util2).

Section 4, *Programming Examples* shows some example programs.

Powering on the analyzer initializes its settings to the factory defaults. For the details about the factory defaults, refer to *Appendix B: Factory Initialization Settings*.

Configuration Commands

These commands configure the system. These commands are equivalent to the **CONFIG** and **START/STOP** menu on the front panel.

Table 4-6: Configuration commands

Header	Description
:Config:Block	Start or stop data acquisition in the Block mode
:Config:Continue	Continue the data acquisition
:Config:FileUtil:CopyFile	Copy a file
:Config:FileUtil:CreateDir	Create a directory
:Config:FileUtil:DeleteDir	Delete a directory
:Config:FileUtil:DeleteFile	Delete a file
:Config:FlushPipeline	Clear the pipeline of the FFT processor
:Config:Label	Display a label on the screen
:Config:Message	Display a message in red on the screen
:Config:Mode	Load settings from the basic configuration file
:Config:Mode:DeleteInitFile	Delete the power-on setting file
:Config:Mode:FactoryReset	Restore the factory default settings
:Config:Mode:Load	Load settings from a file saved before
:Config:Mode:LoadFromInitFile	Load settings from the power-on setting file
:Config:Mode:Save	Save settings to a file
:Config:Mode:SaveToInitFile	Save settings to the power-on setting file
:Config:Next	Stop and restart a block data acquisition
:Config:Pause (?)	Pause the data acquisition on each block
:Config:PrintScreen	Print a screen image to a printer
:Config:PrintScreen:Printer (?)	Specify the printer to print screen images
:Config:PrintScreen:Items?	Return the list of available printers
:Config:PrintScreen:Save	Save a screen image to a file
:Config:Remote (?)	Inhibit front panel key operations
:Config:Roll	Start or stop the data acquisition in the Roll mode
:Config:Setup (?)	Load the setup program for the SETUP menu
:Config:Setup:Activate	Show the Setup menu at the menu level displayed last time
:Config:Setup:Items?	Return the list of available programs for the setup
:Config:Setup:Show	Display the Setup menu at the top menu level

Table 4-6: Configuration commands (Cont.)

Header	Description
:Config:Start	Start a data acquisition in the Block mode
:Config:StartAgain	Stop and restart the acquisition in the Block mode
:Config:Status:Overload?	Query the overload status of the A/D converter
:Config:Status:Pause?	Query if the data acquisition pauses
:Config:Status:Triggered?	Query if the analyzer is triggered
:Config:Status:Uncal?	Query if the analyzer is uncalibrated
:Config:Stop	Stop the data acquisition
:Config:System?	Query the instrument name
:Config:Util:Items?	Return the list of available programs for the utility
:Config:Util<x> (?)	Assign the Utility program to a Utility side key
:Config:Util<x>:Activate	Show the Utility menu at the menu level displayed last time
:Config:Util<x>:Show	Display the Utility menu at the top menu level
:Config:Version?	Query the version of the Configuration program
:Config:View:BackgroundColor (?)	Select the background color for waveform view area
:Config:View:Items?	Return the list of available programs for the view
:Config:View:MarkerLink (?)	Determine whether the markers in different views move in unison or separately
:Config:View:Style (?)	Select the view layout
:Config:View<x> (?)	Assign the View program to a View side key
:Config:View<x>:Activate	Show the View menu at the menu level displayed last time
:Config:View<x>:BringToFront	Put the waveform display area of the view in front on screen
:Config:View<x>:Show	Display the View menu at the top menu level

Setup Commands

Standard Setup Commands

When you select `Standard` in the `Config:Setup` command, use the following commands to control parameters for the standard measurement.
These commands are equivalent to the **SETUP** menu on the front panel.

Table 4–7: Standard Setup commands

Header	Description
:Setup:Band (?)	Select the input frequency band
:Setup:BlockSize (?)	Set the number of frames per block
:Setup:CenterFrequency (?)	Set the center frequency
:Setup:FFTPoints (?)	Set the number of FFT sampling points per frame
:Setup:FFTType (?)	Select the FFT type, either hardware or software
:Setup:FFTWindow (?)	Select the FFT window
:Setup:FramePeriod (?)	Set the frame period in the Block mode
:Setup:FrequencyOffset (?)	Set the frequency display offset
:Setup:IFMode (?)	Select the intermediate frequency (IF) mode
:Setup:InputCoupling (?)	Select the input coupling to the RF input
:Setup:LevelOffset (?)	Set the level display offset
:Setup:Load	Load a trigger mask from a file
:Setup:Manual (?)	Prioritize RF attenuator or mixer level setting
:Setup:MarkerToFreq	Set the center frequency to the value at the marker
:Setup:MaxSpan	Set the span to the maximum
:Setup:MemoryMode (?)	Select the memory mode
:Setup:MixerLevel (?)	Set a mixer level
:Setup:ReferenceLevel (?)	Set the reference level
:Setup:ReferenceOsc (?)	Select the reference oscillator
:Setup:RFAtt (?)	Set an RF attenuator
:Setup:Save	Save a trigger mask to a file
:Setup:Span (?)	Select the span
:Setup:Trigger (?)	Select the trigger mode
:Setup:TriggerCount (?)	Turn the trigger counter on or off
:Setup:TriggerDelayed (?)	Set the delay time in the Delayed trigger mode
:Setup:TriggerDomain (?)	Select the trigger domain
:Setup:TriggerInterval (?)	Set the time interval in the Interval trigger mode

Table 4–7: Standard Setup commands (Cont.)

Header	Description
:Setup:TriggerPosition (?)	Set the trigger position
:Setup:TriggerSlope (?)	Select the trigger slope
:Setup:TriggerSource (?)	Select the trigger source
:Setup:TriggerTimeout (?)	Set the time-out in the Timeout trigger mode
:Setup:TriggerTimes (?)	Set the trigger count in the Count trigger mode
:Setup:Version?	Query the version of the Setup program
:Setup:Zoom:Execute	Expand the waveform
:Setup:Zoom:FFTType (?)	Select the FFT type in the Zoom mode
:Setup:Zoom:FFTWindow (?)	Select the FFT window in the Zoom mode
:Setup:Zoom:Frequency (?)	Set the center frequency for zooming
:Setup:Zoom:Mag (?)	Select the magnification factor for zooming

CDMA Setup Commands

When you select `CDMA` in the `Config:Setup` command, use the following commands to control parameters for the CDMA measurement.
These commands are equivalent to the **SETUP** menu on the front panel.

Table 4-8: CDMA Setup commands

Header	Description
:Setup:BlockSize (?)	Set the number of frames per block
:Setup:CDMA:Channel (?)	Select the channel
:Setup:CDMA:Span30M	Set the span to 30 MHz
:Setup:CDMA:Span50M	Set the span to 50 MHz
:Setup:CDMA:Span5MAuto	Set the span to 5 MHz and the trigger mode to Auto
:Setup:CDMA:Span5MNormal	Set the span to 5 MHz and the trigger mode to Normal
:Setup:CDMA:Standard (?)	Select the CDMA standard
:Setup:CDMA:TriggerLevel (?)	Set the trigger mask level in the time domain
:Setup:MarkerToFreq	Set the center frequency to the value at the marker
:Setup:MaxSpan	Set the span to the maximum
:Setup:ReferenceLevel (?)	Set the reference level
:Setup:ReferenceOsc (?)	Select the reference oscillator
:Setup:Span (?)	Select the span
:Setup:Trigger (?)	Select the trigger mode
:Setup:TriggerCount (?)	Turn the trigger counter on or off
:Setup:TriggerDelayed (?)	Set the delay time in the Delayed trigger mode
:Setup:TriggerDomain (?)	Select the trigger domain
:Setup:TriggerInterval (?)	Set the time interval in the Interval trigger mode
:Setup:TriggerPosition (?)	Set the trigger position in a block
:Setup:TriggerSlope (?)	Select the trigger slope
:Setup:TriggerSource (?)	Select the trigger source
:Setup:TriggerTimeout (?)	Set the time-out in the Timeout trigger mode
:Setup:TriggerTimes (?)	Set the trigger count in the Count trigger mode
:Setup:Version?	Query the version of the CDMA Setup program

3gppACP Setup Commands

When you select `3gppACP` in the `Config:Setup` command, use the following commands to control parameters for the standard measurement.
These commands are equivalent to the **SETUP** menu on the front panel.

Table 4-9: 3gppACP Setup commands

Header	Description
:Setup:ACP:BlockSize (?)	Set the number of frames per block
:Setup:ACP:CarrierWidth (?)	Set the frequency bandwidth of the carrier
:Setup:ACP:CenterFrequency (?)	Set the center frequency
:Setup:ACP:Span (?)	Set the span
:Setup:Band (?)	Set the input frequency band
:Setup:FFTPoints (?)	Set the number of FFT sampling points per frame
:Setup:FFTType (?)	Select the FFT type, hardware or software
:Setup:FFTWindow (?)	Select the FFT window
:Setup:FrequencyOffset (?)	Set the frequency display offset
:Setup:IFMode (?)	Select the Intermediate Frequency (IF) mode
:Setup:InputCoupling (?)	Select the input coupling to the RF input
:Setup:LevelOffset (?)	Set the level display offset
:Setup:Manual (?)	Select the way to set RF attenuator and mixer level
:Setup:MarkerToFreq	Set the center frequency to the value at the marker
:Setup:MemoryMode (?)	Select the memory mode
:Setup:MixerLevel (?)	Select the mixer level
:Setup:ReferenceLevel (?)	Set the reference level
:Setup:ReferenceOsc (?)	Select the reference oscillator
:Setup:RFAtt (?)	Select the RF attenuator
:Setup:Version?	Query the version of the Setup program

View Commands

Waveform View Commands

These commands control the Waveform view.

Table 4–10: Waveform View commands

Header	Description
:View<x>:Average (?)	Determine whether to display the result of averaging
:View<x>:Average:AllFrames	Make all frames averaged
:View<x>:Average:BeginZ (?)	Specify the first frame for averaging
:View<x>:Average:EndZ (?)	Specify the last frame for averaging
:View<x>:Average:Execute	Perform averaging
:View<x>:Average:MarkerToFrame	Specify the averaging range with the delta marker
:View<x>:Average:Reset	Restart averaging
:View<x>:Average:Times (?)	Set the number of frames for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:Compression (?)	Select the way to compress data for displaying
:View<x>:CopyFrom	Copy display data from a text file to the data register
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Edit (?)	Determine whether or not to display the trigger mask
:View<x>:Edit:DrawHorizontal	Draw the entire trigger mask line horizontally through the marker
:View<x>:Edit:DrawLine	Draw the trigger mask line between the marker and the delta marker
:View<x>:Edit:DrawMax	Draw the entire trigger mask line horizontally at the maximum level
:View<x>:Edit:DrawMin	Draw the entire trigger mask line horizontally at the minimum level
:View<x>:Edit:Y (?)	Set the vertical position of the marker in editing the trigger mask
:View<x>:Format (?)	Select the waveform display format
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position

Table 4–10: Waveform View commands (Cont.)

Header	Description
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:Band (?)	Determine whether to display the band marker
:View<x>:Marker:Band:Center (?)	Set the center frequency of the band marker
:View<x>:Marker:Band:Left (?)	Set the left side frequency of the band marker
:View<x>:Marker:Band:Right (?)	Set the right side frequency of the band marker
:View<x>:Marker:Band:Width (?)	Set the bandwidth of the band marker
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:Trace (?)	Select the trace on which the markers are placed
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:MaskVisible (?)	Specify whether or not to display the trigger mask
:View<x>:Measurement (?)	Select the measurement item
:View<x>:Measurement:ACP:BW (?)	Set the bandwidth for the ACP measurement
:View<x>:Measurement:ACP:Lock (?)	Determine whether to lock the band power marker
:View<x>:Measurement:ACP:Marker (?)	Select the marker position for the ACP measurement
:View<x>:Measurement:ACP:SP (?)	Set the frequency interval between adjacent channels for the ACP measurement
:View<x>:Measurement:OBW (?)	Set the occupied bandwidth
:View<x>:Position (?)	Set the display position in a block
:View<x>:RBW (?)	Set the resolution bandwidth (RBW)
:View<x>:RBW:Alpha (?)	Set the shape factor of the RBW filter
:View<x>:RBW:Calculation (?)	Select the RBW filter type
:View<x>:Result<y>?	Query the measurement results
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically

Table 4–10: Waveform View commands (Cont.)

Header	Description
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:XStartZero (?)	Determine whether to set the time origin to zero for each frame
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Trace2:Compression (?)	Select the compression method to display the trace 2
:View<x>:Trace2:Format (?)	Select the display format of the trace 2
:View<x>:Trace2:Source (?)	Select the input data source of the trace 2
:View<x>:Trace2:Z (?)	Set the frame number of the trace 2
:View<x>:Version?	Query the version of the Waveform View program
:View<x>:Z (?)	Set the frame number

Analog View Commands

These commands control the Analog view to analyze an analog modulating signal. This view is capable of demodulating and displaying PM (Phase Modulation), AM (Amplitude Modulation), or FM (Frequency Modulation) signals.

Table 4–11: Analog View commands

Header	Description
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Format (?)	Select the waveform display format
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker

Table 4–11: Analog View commands (Cont.)

Header	Description
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:DeltaX (?)	Query the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:XStartZero (?)	Determine whether to set the time origin to zero for each frame
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the Analog View program
:View<x>:Z (?)	Set the frame number

FSK View Commands

These commands control the FSK view. In this view, the FSK (Frequency Shift Keying) signal is demodulated and displayed.

Table 4–12: FSK View commands

Header	Description
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers

Table 4–12: FSK View commands (Cont.)

Header	Description
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turns on or off the delta marker
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:XStartZero (?)	Determine whether to set the time origin to zero for each frame
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the FSK View program
:View<x>:Z (?)	Set the frame number

Spectrogram View Commands

These commands control the Spectrogram view.

Table 4–13: Spectrogram View commands

Header	Description
:View<x>:Compression (?)	Select the way to compress data for displaying
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:DeltaT?	Query the delta marker position on the time axis
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:DeltaZ (?)	Set the delta marker position on the Z axis
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:T?	Query the marker position on the time axis
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Marker:Z (?)	Set the marker position on the Z axis
:View<x>:Monochrome (?)	Set the display to monochrome
:View<x>:NumberColors (?)	Select the number of display colors
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:TimeScale (?)	Determine whether to display the time scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Scale:ZScale (?)	Scale the Z axis
:View<x>:Scale:ZStart (?)	Set the start point of the Z axis on the screen

Table 4–13: Spectrogram View commands (Cont.)

Header	Description
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the Spectrogram View program
:View<x>:ZGap (?)	Set the interval between two frames on the Z axis

Waterfall View Commands

These commands control the Waterfall view.

Table 4–14: Waterfall View commands

Header	Description
:View<x>:Compression (?)	Select the way to compress data for displaying
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:DeltaT?	Query the delta marker on the time axis
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:DeltaZ (?)	Set the delta marker position on the Z axis
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:T?	Query the marker position on the time axis
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Marker:Z (?)	Set the marker position on the Z axis
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis

Table 4-14: Waterfall View commands (Cont.)

Header	Description
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Scale:ZScale (?)	Scale the Z axis
:View<x>:Scale:ZStart (?)	Set the start point of the Z axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the Waterfall View program
:View<x>:YHeight (?)	Specify the number of pixels of vertical full-scale
:View<x>:ZGap (?)	Set interval between adjacent Z axes on the screen

Polar View Commands

These commands control the Polar view.

Table 4-15: Polar View commands

Header	Description
:View<x>:AlphaBT (?)	Set α/BT
:View<x>:AutoCarrier (?)	Turn on or off the carrier search function
:View<x>:Burst:BlockSize (?)	Set the range for searching a burst
:View<x>:Burst:NumberFrames (?)	Set the range for acquiring burst data
:View<x>:Burst:Offset (?)	Set the start point for acquiring burst data
:View<x>:Burst:PeakThreshold (?)	Set the threshold to determine whether the input signal is a burst
:View<x>:Burst:Search (?)	Determine whether to search a burst
:View<x>:Burst:Threshold (?)	Set the threshold to detect the rising edge of a burst
:View<x>:Carrier (?)	Set the carrier frequency
:View<x>:Display (?)	Select the display data source
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:A?	Query the amplitude at the marker position
:View<x>:Marker:DeltaT (?)	Set the delta marker position on the time axis
:View<x>:Marker:P?	Query the phase at the marker position
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:Marker:X?	Query the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Mask:Frequency (?)	Determine whether the frequency-domain mask is enabled
:View<x>:Mask:Frequency:Center (?)	Set the center frequency of the mask

Table 4–15: Polar View commands (Cont.)

Header	Description
:View<x>:Mask:Frequency:Left (?)	Set the left edge frequency of the mask
:View<x>:Mask:Frequency:Right (?)	Set the right edge frequency of the mask
:View<x>:Mask:Frequency:Width (?)	Set the frequency range of the mask
:View<x>:Mask:MarkerLink (?)	Determine whether the mask frequency links the marker frequency in the other view
:View<x>:MeasDestination (?)	Select the data register to which the measurement data is output
:View<x>:MeasFilter (?)	Select the filter for the measurement data
:View<x>:Modulation (?)	Select the modulation type
:View<x>:Position (?)	Set display position in a block
:View<x>:RefDestination (?)	Select the data register to which the reference data is output
:View<x>:RefFilter (?)	Select the filter for the reference data
:View<x>:Result<y>?	Query the measurement results
:View<x>:Source (?)	Select the input data source
:View<x>:Standard:CDPD	Set parameters according to the CDPD standard
:View<x>:Standard:GSM	Set parameters according to the GSM standard
:View<x>:Standard:NADC	Set parameters according to the NADC standard
:View<x>:Standard:PDC	Set parameters according to the PDC standard
:View<x>:Standard:PHS	Set parameters according to the PHS standard
:View<x>:Standard:TETRA	Set parameters according to the TETRA standard
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:Version?	Query the version of the Polar View program
:View<x>:Z (?)	Set the frame number

Eye Diagram View Commands

These commands control the Eye Diagram view.

Table 4–16: Eye Diagram View commands

Header	Description
:View<x>:EyeLength (?)	Set the number of symbols on the horizontal axis
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:Marker:Y?	Query the vertical position of the marker

Table 4–16: Eye Diagram View commands (Cont.)

Header	Description
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the Eye Diagram View program

Symbol Table View Commands

These commands control the Symbol Table view.

Table 4–17: Symbol Table View commands

Header	Description
:View<x>:CopyTo	Store display data to a file or data register
:View<x>:Marker:Data?	Query the value at the marker position
:View<x>:Marker:Symbol?	Query the symbol location where the marker is placed
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:Radix (?)	Select the radix to display the Symbol Table
:View<x>:Rotate (?)	Set the reference phase
:View<x>:Source (?)	Select the input data source
:View<x>:Symbol (?)	Set the symbol location where the marker is placed
:View<x>:Version?	Query the version of the Symbol Table View program

EVM View Commands

These commands control the EVM (Error Vector Magnitude) view.

Table 4–18: EVM View commands

Header	Description
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Format (?)	Select the waveform display format
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off

Table 4–18: EVM View commands (Cont.)

Header	Description
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Mask:Left (?)	Specify the left edge of EVM calculation range
:View<x>:Mask:Percent (?)	Specify the area to be excluded from the calculation
:View<x>:Mask:Right (?)	Specify the right edge of EVM calculation range
:View<x>:Mask:Type (?)	Select the way to specify the EVM calculation range
:View<x>:Result<y>?	Query the measurement results
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Symbol (?)	Set the marker position with the symbol number
:View<x>:Version?	Query the version of the EVM View program

CDMA Waveform View Commands

These commands control the CDMA Waveform view.

Table 4–19: CDMA Waveform View commands

Header	Description
:View<x>:Average:Times (?)	Set the number of frames for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:CDMA:Channel (?)	Select the channel
:View<x>:CDMA:Standard (?)	Select the CDMA standard
:View<x>:Compression (?)	Select the way to compress data for displaying

Table 4-19: CDMA Waveform View commands (Cont.)

Header	Description
:View<x>:CopyFrom	Copy display data from a text file to data register
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Format (?)	Select the waveform display format
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Mask:RBW1M:Frequency (?)	Set frequency of the reference line at 1 MHz RBW
:View<x>:Mask:RBW1M:Level (?)	Set level of the reference line at 1 MHz RBW
:View<x>:Mask:RBW30k:Frequency1 (?)	Set the inner frequency of the reference line at 30 kHz RBW
:View<x>:Mask:RBW30k:Frequency2 (?)	Set the outer frequency of the reference line at 30 kHz RBW
:View<x>:Mask:RBW30k:Level1 (?)	Set the inner level of the reference line at 30 kHz RBW
:View<x>:Mask:RBW30k:Level2 (?)	Set the outer level of the reference line at 30 kHz RBW
:View<x>:Measurement (?)	Select the measurement function
:View<x>:Measurement:OBW (?)	Set the occupied bandwidth

Table 4-19: CDMA Waveform View commands (Cont.)

Header	Description
:View<x>:Measurement:Separation (?)	Set the spurious resolution
:View<x>:Measurement:SortedBy (?)	Select the sort key of spurious
:View<x>:Measurement:Spurious-Search	Turn the spurious search function on or off
:View<x>:Measurement:Threshold (?)	Set the threshold for the spurious search
:View<x>:Position (?)	Set the display position in a block
:View<x>:RBW (?)	Set the resolution bandwidth
:View<x>:Result<y>?	Query the measurement results
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to hold or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:XStartZero (?)	Determine whether to set the time origin to zero for each frame
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the CDMA Waveform View program
:View<x>:Z (?)	Set the frame number

CDMA Polar View Commands

These commands control the CDMA Polar view.

Table 4-20: CDMA Polar View commands

Header	Description
:View<x>:AlphaBT (?)	Set α/BT
:View<x>:AutoCarrier (?)	Turn the carrier search function on or off
:View<x>:Burst:BlockSize (?)	Set the range for searching a burst
:View<x>:Burst:NumberFrames (?)	Set the range for acquiring burst data
:View<x>:Burst:Offset (?)	Set the start point for acquiring burst data
:View<x>:Burst:PeakThreshold (?)	Set the threshold to determine whether the input signal is a burst

Table 4–20: CDMA Polar View commands (Cont.)

Header	Description
:View<x>:Burst:Search (?)	Determine whether to search a burst
:View<x>:Burst:Threshold (?)	Set the threshold to detect the rising edge of a burst
:View<x>:Carrier (?)	Set the carrier frequency
:View<x>:Display (?)	Select the display data source
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:A?	Query the amplitude at the marker position
:View<x>:Marker:DeltaT (?)	Set the delta marker position on the time axis
:View<x>:Marker:P?	Query the phase at the marker position
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:Marker:X?	Query the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Mask:Frequency (?)	Determine whether the frequency-domain mask is enabled
:View<x>:Mask:Frequency:Center (?)	Set the center frequency of the mask
:View<x>:Mask:Frequency:Left (?)	Set the left edge frequency of the mask
:View<x>:Mask:Frequency:Right (?)	Set the right edge frequency of the mask
:View<x>:Mask:Frequency:Width (?)	Set the frequency range of the mask
:View<x>:Mask:MarkerLink (?)	Determine whether the mask frequency links the marker frequency in the other view
:View<x>:MeasDestination (?)	Select the data register to which the measurement data is output
:View<x>:MeasFilter (?)	Select the filter for the measurement data
:View<x>:Modulation (?)	Select the modulation type
:View<x>:Position (?)	Set the display position in a block
:View<x>:RefDestination (?)	Select the data register to which the reference data is output
:View<x>:RefFilter (?)	Select the filter for the reference data
:View<x>:Result<y>?	Query the measurement results
:View<x>:Source (?)	Select the input data source
:View<x>:Standard:CDPD	Set parameters according to the CDPD standard
:View<x>:Standard:GSM	Set parameters according to the GSM standard
:View<x>:Standard:IS95	Set parameters according to the IS-95 standard
:View<x>:Standard:NADC	Set parameters according to the NADC standard
:View<x>:Standard:PDC	Set parameters according to the PDC standard

Table 4-20: CDMA Polar View commands (Cont.)

Header	Description
:View<x>:Standard:PHS	Set parameters according to the PHS standard
:View<x>:Standard:TETRA	Set parameters according to the TETRA standard
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:Version?	Query the version of the CDMA Polar View program
:View<x>:Z (?)	Set the frame number

CDMA Time View Commands

These commands control the CDMA Time view.

Table 4-21: CDMA Time View commands

Header	Description
:View<x>:Average:Times (?)	Set the number of frames for averaging
:View<x>:Block (?)	Set the block number
:View<x>:BreakMeasure	Stop executing the Measure function
:View<x>:BreakMeasureData	Stop executing the Measure Data function
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker

Table 4–21: CDMA Time View commands (Cont.)

Header	Description
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Mask:OffLeft (?)	Set the reference time just before the rising edge
:View<x>:Mask:OffLevel (?)	Set the reference off-level
:View<x>:Mask:OffRight (?)	Set the reference time just after the falling edge
:View<x>:Mask:OnLeft (?)	Set the reference time after rising edge
:View<x>:Mask:OnLevel (?)	Set the reference on-level
:View<x>:Mask:OnRight (?)	Set the reference time just before the falling edge
:View<x>:Measure	Average newly acquired data and perform PASS/FAIL test
:View<x>:MeasureData	Average data on the memory and perform PASS/FAIL test
:View<x>:Position (?)	Set the display position in a block
:View<x>:Result<y>?	Query measurement results
:View<x>:Scale:FallingEdge	Expand the falling edge of waveform on the screen
:View<x>:Scale:FullScale	Display the whole waveform
:View<x>:Scale:HoldYScale (?)	Determine whether to hold or reset the vertical scale when you change the input source
:View<x>:Scale:RisingEdge	Expand the rising edge of waveform on the screen
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Trace2:TraceVisible (?)	Determine whether or not to display Trace 2
:View<x>:TraceVisible (?)	Determine whether or not to display Trace 1
:View<x>:Version?	Query the version of the CDMA Time View program

CodeSpectrogram View Commands

These commands control the code-domain spectrogram view according to the cdmaOne standard.

Table 4–22: CodeSpectrogram View commands

Header	Description
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:TotalPower?	Query the total power of the time slot at the marker
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Z (?)	Set the marker position on the Z axis
:View<x>:Monochrome (?)	Set the display to monochrome
:View<x>:NumberColors (?)	Select the number of display colors
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Scale:ZScale (?)	Scale the Z axis
:View<x>:Scale:ZStart (?)	Set the start point of the Z axis on the screen
:View<x>:Version?	Query the version of the CodeSpectrogram View program
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the Y (color) axis
:View<x>:ZGap (?)	Set the symbol display interval on the Z axis

CodePolar View Commands

These commands control the vector diagram view according to the cdmaOne standard.

Table 4–23: CodePolar View commands

Header	Description
:View<x>:AlphaBT (?)	Set α/BT
:View<x>:Analysis:Symbol (?)	Select the symbol to display the constellation
:View<x>:Analyze	Perform analysis on the background for all symbols
:View<x>:AutoCarrier (?)	Turn on or off the carrier search function
:View<x>:BreakAnalyze	Stop analysis
:View<x>:Carrier (?)	Set the carrier frequency
:View<x>:ChipRate (?)	Set the chip rate
:View<x>:Display (?)	Select the display data source
:View<x>:Format (?)	Select the waveform display format
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:MeasFilter (?)	Select the filter to create measurement data
:View<x>:Modulation (?)	Select the modulation type
:View<x>:RefFilter (?)	Select the filter to create reference data
:View<x>:Source (?)	Select the input data source
:View<x>:Standard:IS95	Configure the modulating system according to the IS-95 standard without the equalizer
:View<x>:Standard:IS95EQ	Configure the modulating system according to the IS-95 standard with the equalizer
:View<x>:Version?	Query the version of the CodePolar View program

CodePower View Commands

These commands control the code-domain power view according to the cdmaOne standard.

Table 4–24: CodePower View commands

Header	Description
:View<x>:Average (?)	Determine whether to display average results
:View<x>:Average:AllFrames	Specify the average range to all frames
:View<x>:Average:BeginZ (?)	Set the uppermost frame in the average range
:View<x>:Average:EndZ (?)	Set the lowermost frame in the average range
:View<x>:Average:Execute	Execute averaging

Table 4–24: CodePower View commands (Cont.)

Header	Description
:View<x>:Average:MarkerToFrame	Specify the average range with the delta marker
:View<x>:Average:Times (?)	Set the number of acquisitions for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:TotalPower?	Query the total power of the time slot
:View<x>:Version?	Query the version of the CodePower View program
:View<x>:XAxis (?)	Select the variable for the horizontal axis
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the vertical axis
:View<x>:Z (?)	Set the symbol number

CodeWSpectrogram View Commands

These commands control the code-domain spectrogram view according to the W-CDMA standard.

Table 4–25: CodeWSpectrogram View commands

Header	Description
:View<x>:Marker:Channel?	Query the channel number at the marker position
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:SymbolRate?	Query the symbol rate at the marker position
:View<x>:Marker:TimeSlot?	Query the time slot number at the marker
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:TotalPower?	Query the total power of the time slot at the marker
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Z (?)	Set the marker position on the Z axis
:View<x>:Monochrome (?)	Set the display to monochrome
:View<x>:NumberColors (?)	Select the number of display colors
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Scale:ZScale (?)	Scale the Z axis
:View<x>:Scale:ZStart (?)	Set the start point of the Z axis on the screen
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:Version?	Query the version of the CodeWSpectrogram View program
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the Y (color) axis
:View<x>:ZGap (?)	Set the symbol display interval on the Z axis

CodeWPolar View Commands

These commands control the vector diagram view according to the W-CDMA standard.

Table 4–26: CodeWPolar View commands

Header	Description
:View<x>:AlphaBT (?)	Set α/BT
:View<x>:Analysis:TimeSlot (?)	Specify the symbol to display the constellation
:View<x>:Analyze	Perform analysis on the background for all symbols
:View<x>:AutoCarrier (?)	Turn on or off the carrier search function
:View<x>:BreakAnalyze	Stop analysis
:View<x>:Carrier (?)	Set the carrier frequency
:View<x>:ChipRate (?)	Set the chip rate
:View<x>:Constellation:SymbolRate (?)	Set the symbol rate to display symbol constellation
:View<x>:Display (?)	Select the display data source
:View<x>:Format (?)	Select the waveform display format
:View<x>:HideLMSPart (?)	Determine whether to display the LMS
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:MeasFilter (?)	Select the filter to create measurement data
:View<x>:Modulation (?)	Select the modulation type
:View<x>:RefFilter (?)	Select the filter to create reference data
:View<x>:ShortCode (?)	Specify the short code
:View<x>:Source (?)	Select the input data source
:View<x>:Standard:WCDMA16M	Configure the modulating system according to the W-CDMA standard with the chip rate of 16 M/s
:View<x>:Standard:WCDMA4M	Configure the modulating system according to the W-CDMA standard with the chip rate of 4 M/s
:View<x>:Standard:WCDMA8M	Configure the modulating system according to the W-CDMA standard with the chip rate of 8 M/s
:View<x>:SymbolConstellation (?)	Determine whether to display symbol constellation
:View<x>:TimeSlot (?)	Specify the time slot
:View<x>:Version?	Query the version of the CodeWPolar View program

CodeWPower View Commands

These commands control the code-domain power view according to the W-CDMA standard.

Table 4–27: CodeWPower View commands

Header	Description
:View<x>:Average (?)	Determine whether to display average results
:View<x>:Average:AllFrames	Specify the average range of all frames
:View<x>:Average:BeginZ (?)	Set the uppermost frame in the average range
:View<x>:Average:EndZ (?)	Set the lowermost frame in the average range
:View<x>:Average:Execute	Execute averaging
:View<x>:Average:MarkerToFrame	Specify the averaging range with the delta marker
:View<x>:Average:Times (?)	Set the number of acquisitions for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:Channel?	Query the channel number at the marker position
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:SymbolRate?	Query the symbol rate at the marker position
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis

Table 4–27: CodeWPower View commands (Cont.)

Header	Description
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:ShortCode (?)	Specify the short code
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:TimeSlot?	Query the time slot number
:View<x>:TotalPower?	Query the total power of the time slot
:View<x>:Version?	Query the version of the CodeWPower View program
:View<x>:XAxis (?)	Select the variable for the horizontal axis
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the vertical axis
:View<x>:Z (?)	Set the time slot number

3gppACPView Commands

These commands control the 3gppACPView according to the 3GPP standard.

Table 4–28: 3gppACPView commands

Header	Description
:View<x>:Average (?)	Determine whether to display the result of averaging
:View<x>:Average:AllFrames	Specify the averaging range of all frames
:View<x>:Average:BeginZ (?)	Set the uppermost frame for the averaging range
:View<x>:Average:EndZ (?)	Set the lowermost frame for the averaging range
:View<x>:Average:Execute	Start averaging
:View<x>:Average:MarkerToFrame	Specify the averaging range between two markers
:View<x>:Average:Reset	Restart averaging
:View<x>:Average:Times (?)	Set the number of frames for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:CopyFrom	Copy display data from a text file to the data register
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position

Table 4–28: 3gppACPView commands (Cont.)

Header	Description
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:Band (?)	Determine whether to display the band marker
:View<x>:Marker:Band:Center (?)	Set the center frequency of the band marker
:View<x>:Marker:Band:Left (?)	Set the left side frequency of the band marker
:View<x>:Marker:Band:Right (?)	Set the right side frequency of the band marker
:View<x>:Marker:Band:Width (?)	Set the bandwidth of the band marker
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:DeltaX (?)	Set the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Measurement (?)	Select the measurement item
:View<x>:Measurement:ACP:BW (?)	Set the bandwidth for the ACP measurement
:View<x>:Measurement:ACP:SP (?)	Set the frequency interval between adjacent channels for the ACP measurement
:View<x>:Measurement:Filter (?)	Turn the receive filter on or off
:View<x>:Measurement:Filter:Alpha (?)	Set the shape factor of the receive filter
:View<x>:Measurement:Filter:BW (?)	Set the frequency bandwidth of the receive filter
:View<x>:Measurement:Filter:SP (?)	Set the interval between adjacent receive filters
:View<x>:Measurement:OBW (?)	Set the occupied bandwidth
:View<x>:RBW (?)	Set the resolution bandwidth (RBW)
:View<x>:RBW:Alpha (?)	Set the shape factor of the RBW filter
:View<x>:RBW:Calculation (?)	Select the type of the RBW filter
:View<x>:Result<y>?	Query the measurement results
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale

Table 4–28: 3gppACPView commands (Cont.)

Header	Description
:View<x>:Scale:HoldYScale (?)	Determine whether to hold or reset the vertical scale when you change the input source
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the Waveform View program

3gppSpectrogram View Commands

These commands control the code-domain spectrogram view according to the 3GPP standard.

Table 4–29: 3gppSpectrogram View commands

Header	Description
:View<x>:Marker:Channel?	Query the channel number at the marker position
:View<x>:Marker:DeltaMarker (?)	Turn the delta marker on or off
:View<x>:Marker:PSCHPower?	Query the absolute power of PSCH (Primary Synchronization Channel) at the marker position
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SCG?	Query SCG (Scrambling Code Group) at the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:SSCH?	Query SSCH (Secondary Synchronization Channel) at the marker position
:View<x>:Marker:SSCHPower?	Query the absolute power of PSCH (Secondary Synchronization Channel) at the marker position
:View<x>:Marker:SymbolRate?	Query the symbol rate at the marker position
:View<x>:Marker:TimeSlot?	Query the time slot number at the marker
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:TotalPower?	Query the total power of the time slot at the marker
:View<x>:Marker:X (?)	Set the horizontal position of the marker

Table 4–29: 3gppSpectrogram View commands (Cont.)

Header	Description
:View<x>:Marker:Z (?)	Set the marker position on the Z axis
:View<x>:Monochrome (?)	Set the display to monochrome
:View<x>:NumberColors (?)	Select the number of display colors
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Scale:ZScale (?)	Scale the Z axis
:View<x>:Scale:ZStart (?)	Set the start point of the Z axis on the screen
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:Version?	Query the version of the CodeWSpectrogram View program
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the Y (color) axis
:View<x>:ZGap (?)	Set the symbol display interval on the Z axis

3gppPolar View Commands

These commands control the vector diagram view according to the 3GPP standard.

Table 4–30: 3gppPolar View commands

Header	Description
:View<x>:AlphaBT (?)	Set α/BT
:View<x>:Analysis:TimeSlot (?)	Specify the symbol to display the constellation
:View<x>:Analyze	Perform analysis on the background for all symbols
:View<x>:AutoCarrier (?)	Turn on or off the carrier search function
:View<x>:BreakAnalyze	Stop analysis
:View<x>:Carrier (?)	Set the carrier frequency
:View<x>:ChipRate (?)	Set the chip rate
:View<x>:Constellation:SymbolRate (?)	Set the symbol rate to display symbol constellation
:View<x>:Display (?)	Select the display data source
:View<x>:Downlink:ScramblingCode (?)	Specify the scrambling code

Table 4–30: 3gppPolar View commands (Cont.)

Header	Description
:View<x>:Downlink:ScramblingCode-Search (?)	Determine whether or not to search for the scrambling code to analyze the down-link signal
:View<x>:Format (?)	Select the waveform display format
:View<x>:HideSCHPart (?)	Determine whether or not to display SCH
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:MeasFilter (?)	Select the filter to create measurement data
:View<x>:Modulation (?)	Select the modulation type
:View<x>:RefFilter (?)	Select the filter to create reference data
:View<x>:ShortCode (?)	Specify the short code
:View<x>:Source (?)	Select the input data source
:View<x>:Standard:WCDMA	Configure the modulating system according to the W-CDMA standard
:View<x>:SymbolConstellation (?)	Determine whether to display symbol constellation
:View<x>:TimeSlot (?)	Specify the time slot
:View<x>:Version?	Query the version of the CodeWPolar View program

3gppPower View Commands

These commands control the code-domain power view according to the 3GPP standard.

Table 4–31: 3gppPower View commands

Header	Description
:View<x>:Average (?)	Determine whether to display average results
:View<x>:Average:AllFrames	Specify the average range of all frames
:View<x>:Average:BeginZ (?)	Set the uppermost frame in the average range
:View<x>:Average:EndZ (?)	Set the lowermost frame in the average range
:View<x>:Average:Execute	Execute averaging
:View<x>:Average:MarkerToFrame	Specify the averaging range with the delta marker
:View<x>:Average:Times (?)	Set the number of acquisitions for averaging
:View<x>:Average:Type (?)	Select the average type
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position

Table 4–31: 3gppPower View commands (Cont.)

Header	Description
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:Channel?	Query the channel number at the marker position
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:SymbolRate?	Query the symbol rate at the marker position
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:PSCHPower?	Query the absolute power of PSCH (Primary Synchronization Channel)
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:SCG?	Query SCG (Scrambling Code Group)
:View<x>:ShortCode (?)	Specify the short code
:View<x>:SSCH?	Query SSCH (Secondary Synchronization Channel)
:View<x>:SSCHPower?	Query the absolute power of PSCH (Secondary Synchronization Channel)
:View<x>:SymbolRate (?)	Set the symbol rate
:View<x>:TimeSlot?	Query the time slot number
:View<x>:TotalPower?	Query the total power of the time slot
:View<x>:Version?	Query the version of the CodeWPower View program
:View<x>:XAxis (?)	Select the variable for the horizontal axis
:View<x>:YAxis (?)	Determine whether to represent relative or absolute power along the vertical axis
:View<x>:Z (?)	Set the time slot number

GSM View Commands

These commands control the GSM (Global System for Mobile Communication) analysis.

Table 4–32: GSM View commands

Header	Description
:View<x>:Analyze	Measure data acquired on the memory or loaded from a file
:View<x>:Average (?)	Set the number of bursts to acquire
:View<x>:Burst (?)	Set the number of the burst to analyze
:View<x>:Measure	Start to acquire and process data
:View<x>:Script (?)	Select the GSM measurement item

GSMPolar View Commands

These commands control the polar view according to the GSM standard.

Table 4–33: GSMPolar View commands

Header	Description
:View<x>:Burst:AnalysisLength (?)	Set the process range in bits for GSM measurement
:View<x>:Burst:SyncWordLength (?)	Set the number of bits of the sync word
:View<x>:Burst:SyncWordPattern (?)	Select the sync word for GSM measurement
:View<x>:Burst:SyncWordPosition (?)	Set the sync word position in waveform display for GSM measurement
:View<x>:Marker:P?	Query the phase at the marker position
:View<x>:Marker:T (?)	Set the marker position on the time axis
:View<x>:Marker:X?	Query the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Result<y>?	Query the measurement results
:View<x>:Version?	Query the version of the GSMPolar View program

GSMMask View Commands

These commands control the power vs. time Pass/Fail test according to the GSM standard.

Table 4–34: GSMMask View commands

Header	Description
:View<x>:CopyTo	Copy display data to a file or data register
:View<x>:Line:DeltaX<n>?	Query the difference between two vertical line markers
:View<x>:Line:DeltaY<n>?	Query the difference between two horizontal line markers
:View<x>:Line:X<n> (?)	Set the vertical line marker position
:View<x>:Line:X<n>:Visible (?)	Specify whether to display the vertical line marker
:View<x>:Line:Y<n> (?)	Set the horizontal line marker position
:View<x>:Line:Y<n>:Visible (?)	Specify whether to display the horizontal line marker
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:DeltaX?	Query the horizontal position of the delta marker
:View<x>:Marker:DeltaY?	Query the vertical position of the delta marker
:View<x>:Marker:Peak	Move the marker to the adjacent peak
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:ToggleDelta	Change the main marker and delta marker positions each other
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Marker:Y?	Query the vertical position of the marker
:View<x>:Result1?	Query the power vs. time Pass/Fail test result
:View<x>:Scale:AutoYScale	Scale the vertical axis automatically
:View<x>:Scale:FallingEdge	Expand the falling edge horizontally on screen
:View<x>:Scale:FullYScale	Set the vertical scale to the default full-scale
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:RisingEdge	Expand the rising edge horizontally on screen
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen

Table 4–34: GSMMask View commands (Cont.)

Header	Description
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Version?	Query the version of the GSMMask View program

CCDF Commands

These commands control the CCDF (Cumulative Complementary Distribution Function) analysis.

Table 4–35: CCDF commands

Header	Description
:View<x>:AllFrames	Specify the CCDF calculation range of all frames
:View<x>:Average:Reset	Restart the CCDF measurement
:View<x>:BeginZ (?)	Set the uppermost frame in the calculation range
:View<x>:Destination	Select the destination to output the calculation results
:View<x>:EndZ (?)	Set the lowermost frame in the calculation range
:View<x>:Execute	Execute CCDF calculation
:View<x>:Marker:DeltaMarker (?)	Turn on or off the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:MarkerToFrame	Specify the calculation range between two markers
:View<x>:OutputFormat (?)	Select the format for displaying calculation results
:View<x>:Position (?)	Set the display position in a block
:View<x>:Resolution (?)	Set the resolution of the histogram
:View<x>:Result<y>?	Query the measurement results
:View<x>:Scale:AutoScale	Scale the vertical axis automatically
:View<x>:Scale:HoldYScale (?)	Determine whether to retain or reset the vertical scale when you change the input source
:View<x>:Scale:Origin	Reset the vertical and horizontal axis to the default
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen

Table 4–35: CCDF commands (Cont.)

Header	Description
:View<x>:Scale:XStartZero (?)	Determine whether to set the time origin to zero for each frame
:View<x>:Scale:YScale (?)	Scale the vertical axis
:View<x>:Scale:YStart (?)	Set the start point of the vertical axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the CCDF program
:View<x>:Z (?)	Set the frame number

CCDFView Commands

These commands control the view of CCDF (Cumulative Complementary Distribution Function) measurement results.

Table 4–36: CCDFView commands

Header	Description
:View<x>:CopyFrom	Copy display data from a data register or text file
:View<x>:CopyTo	Copy display data to a data register or text file
:View<x>:Marker:DeltaMarker (?)	Turns on or off the delta marker
:View<x>:Marker:ResetDelta	Move the delta marker to the marker position
:View<x>:Marker:SearchMax	Search the maximum peak and place the marker on it
:View<x>:Marker:SearchMin	Search the minimum peak and place the marker on it
:View<x>:Marker:SearchSeparation (?)	Set the resolution to separate two peaks
:View<x>:Marker:X (?)	Set the horizontal position of the marker
:View<x>:Scale:AutoScale	Scale the vertical axis automatically
:View<x>:Scale:LYStart (?)	Set the bottom edge of the vertical axis on the screen
:View<x>:Scale:LYStop (?)	Set the top edge of the vertical axis on the screen
:View<x>:Scale:Origin (?)	Reset the vertical and horizontal axis scale to the default
:View<x>:Scale:XScale (?)	Scale the horizontal axis
:View<x>:Scale:XStart (?)	Set the start point of the horizontal axis on the screen
:View<x>:Source (?)	Select the input data source
:View<x>:Version?	Query the version of the CCDFView program

Utility Commands

These commands are equivalent to the **UTILITY** menu on the front panel.

Self Gain-Calibration Commands

These commands control the self gain-calibration.

Table 4–37: Self Gain-Calibration commands

Header	Description
:Util1:Gain:Auto (?)	Determine whether to perform the self gain-calibration automatically
:Util1:Gain:Execute	Perform the self gain-calibration
:Util1:IQOffset:Execute	Compensate the offset of the I and Q input signals
:Util1:Result1?	Query the calibration results
:Util1:Version?	Query the version of the Self Gain–Calibration program
:Util1:WideIQBalance:Execute	Balance DC components of I and Q signals in the Wide IF mode
:Util1:WideOffset:Request	Compensate the offset in the Wide IF mode at the next data acquisition

Save/Load Commands

These commands control to save or load data.

Table 4–38: Save/Load commands

Header	Description
:Util2:AllFrames	Specify that the all frames are saved
:Util2:BeginZ (?)	Select the first frame to be saved
:Util2:Buffer:<header> (?)	Set the file header in the buffer
:Util2:Buffer:CopyHeader (?)	Store file header to the buffer
:Util2:Buffer:SaveHeader (?)	Save the file header in the buffer to the file
:Util2:Data:Load	Load data from a file in the IQ format
:Util2:Data:Save	Save data to a file in the IQ or AP format
:Util2:Data:SaveAP	Save data to a file in the AP format
:Util2:Data:Saved	Inform other system programs of the name of the updated file
:Util2:Data:SaveDateTime	Add date and time for the specified frame to the file
:Util2:Data:SaveFlatness	Add calibration data to the file

Table 4–38: Save/Load commands (Cont.)

Header	Description
:Util2:Data:SaveFrame	Add frame data to the file
:Util2:Data:SaveHeader	Save file header for the specified frames to the file
:Util2:Data:SaveIQ	Save data to a file in the IQ format
:Util2:EndZ (?)	Specify the last frame to be saved
:Util2:MarkerToFrame	Specify that the frames between the main marker and the delta marker are saved
:Util2:Source (?)	Select the data source to be saved
:Util2:Version?	Query the version of the Save/Load program

Average Commands

These commands control averaging.

Table 4–39: Average commands

Header	Description
:Util3:AllFrames	Specify that all frames are processed
:Util3:BeginZ (?)	Specify the first frame to be processed
:Util3:Destination (?)	Specify the destination to which the result of the process is output
:Util3:EndZ (?)	Specify the last frame to be processed
:Util3:MarkerToFrame	Specify that the frames between the main marker and the delta marker are processed
:Util3:PeakHold	Perform average in the peak hold mode
:Util3:RMS	Perform average in the RMS mode
:Util3:Source (?)	Select the input data source
:Util3:Version?	Query the version of the Average program

Remote Commands

These commands control remote operation.

IEEE 488.2 common commands. The Remote commands include several IEEE 488.2 common commands. The syntax for an IEEE 488.2 common command is an asterisk (*) followed by a command and, optionally, a space and parameter value. The syntax for an IEEE 488.2 common query is an asterisk (*) followed by a query and a question mark.

Table 4-40: Remote commands

Header	Description
*CLS	Clear status
*ESE (?)	Standard Event Status Enable command
*ESR?	Standard Event Status Register query
*IDN?	Identification query
*LRN?	Learn device setting
*OPC (?)	Operation complete command
*RST	Reset the analyzer to the factory default
*SRE (?)	Service Request Enable command
*STB?	Read status byte
[:Util8]:Clipboard?	Query the contents of the clipboard
[:Util8]:Data:<type>?	Get data of the specified type from a frame
[:Util8]:Dev<x>:<command>	Send the command to the GPIB device
[:Util8]:Error?	Query the error code and message
[:Util8]:Event:<message>	Generate an event
[:Util8]:FactoryReset	Restore the factory default settings
[:Util8]:Format (?)	Select the waveform display format
[:Util8]:GPIB:Interface (?)	Select the GPIB configuration
[:Util8]:GPIB:PrimaryAddress (?)	Set the GPIB primary address
[:Util8]:GPIB:RWTimeout (?)	Set the time-out for transferring data via GPIB
[:Util8]:Header (?)	Determine whether to include headers on responses
[:Util8]:Id?	Query identifying information about the analyzer
[:Util8]:Key	Equivalent to pressing the button on the front panel
[:Util8]:NumericOutput (?)	Select the numerical format of the returned value
[:Util8]:ReadFile (?)	Transfer a file
[:Util8]:Register:Data	Write a value to the data register
[:Util8]:Register:Data:<register> (?)	Write values to the specified data register

Table 4–40: Remote commands (Cont.)

Header	Description
[:Util8]:Register:Header	Specify axes scaling and labeling for the data register
[:Util8]:RemoteCommand:<command>	Send the command to another analyzer via TCP/IP
[:Util8]:Set?	Query the instrument settings
[:Util8]:Source (?)	Select the input data source
[:Util8]:Source:<item>?	Query the setting for the item
[:Util8]:Status?	Query the status of the analyzer
[:Util8]:TCPIP:NewLine (?)	Select the new-line character of query responses
[:Util8]:TCPIP:Port:Command (?)	Set the TCP/IP command port number
[:Util8]:TCPIP:Port:Command:Reset	Close the TCP/IP command port
[:Util8]:TCPIP:Port:Event (?)	Set the TCP/IP event port number
[:Util8]:TCPIP:Port:Event:Reset	Close the TCP/IP event port
[:Util8]:TCPIP:Port:RemoteCommand	Connect a slave analyzer via TCP/IP
[:Util8]:TCPIP:Port:RemoteCommand :Reset	Disconnect the slave analyzer on the TCP/IP network
[:Util8]:TCPIP:RwTimeout (?)	Set a time-out for transferring data via TCP/IP
[:Util8]:Version?	Query the version of the Remote program
[:Util8]:ViewName?	Query the name of the Remote program
[:Util8]:Z (?)	Set the frame number
[:Util8]:Z:<item>?	Query information about the frame of input data

Configuration Commands

Use the Configuration commands to set the analyzer with the basic configuration pattern before acquiring, displaying, and measuring spectra.

:Config:Block (No Query Form)

Starts data acquisition in the Block mode. During data acquisition, this command stops it.

Syntax `:Config:Block`

Arguments None

Examples `:Config:Block`
starts data acquisition in the Block mode. During data acquisition, this command stops it.

Related Commands `:Config:Start`, `:Config:Stop`

:Config:Continue (No Query Form)

Breaks the pause and starts data acquisition again.

Syntax `:Config:Continue`

Arguments None

Examples `:Config:Continue`
breaks the pause and starts data acquisition again.

Related Commands `:Config:Pause`

:Config:FileUtil:CopyFile (No Query Form)

Copies a source file to a destination file.

Syntax

`:Config:FileUtil:CopyFile <source>,<destination>`

Arguments

`<source>::=<string>` specifies a source file.

`<destination>::=<string>` specifies a destination file.

If the source file is *.IQ and has a *.IQT file, the *.IQT file is also automatically copied.

If the source file is *.AP and has a *.APT file, the *.APT file is also automatically copied.

If the source file is *.CFG and has a *.TRG file, the *.TRG file is also automatically copied.

Examples

`:Config:FileUtil:CopyFile "C:\Temp\TTT.IQ","C:\Temp\TTT2.IQ"`
copies the file C:\Temp\TTT.IQ to C:\Temp\TTT2.IQ.

Related Commands

`:Config:FileUtil:DeleteFile`

:Config:FileUtil:CreateDir (No Query Form)

Makes a new directory.

Syntax

`:Config:FileUtil:CreateDir <string>`

Arguments

`<string>` specifies a directory name.

Examples

`:Config:FileUtil:CreateDir "C:\Temp\TTT"`
makes the new directory C:\Temp\TTT.

Related Commands

`:Config:FileUtil:DeleteDir`

:Config:FileUtil:DeleteDir (No Query Form)

Deletes a directory.

Syntax

`:Config:FileUtil:DeleteDir <string>`

Arguments

`<string>` specifies a directory name.

Examples

`:Config:FileUtil:DeleteDir "C:\Temp\TTT"`
deletes the directory C:\Temp\TTT.

Related Commands

`:Config:FileUtil:CreateDir`

:Config:FileUtil:DeleteFile (No Query Form)

Deletes a file.

Syntax

`:Config:FileUtil:DeleteFile <string>`

Arguments

`<string>` specifies a file name.

If the file is *.IQ and has a *.IQT file, the *.IQT file is also automatically deleted.

If the file is *.AP and has a *.APT file, the *.APT file is also automatically deleted.

If the file is *.CFG and has a *.TRG file, the *.TRG file is also automatically deleted.

Examples

`:Config:FileUtil:DeleteFile "C:\Temp\TTT.IQ"`
deletes the file C:\Temp\TTT.IQ.

Related Commands

`:Config:FileUtil:CreateDir`

:Config:FlushPipeline (No Query Form)

Clears the FFT processor pipeline. Execute this command before each block data acquisition in the Block mode, except the first block data acquisition before that the analyzer always clears the pipeline.

Syntax `:Config:FlushPipeline`

Arguments None

Examples `:Config:FlushPipeline`
clears the FFT processor pipeline.

:Config:Label (No Query Form)

Displays label in black on the upper left of screen.

Syntax `:Config:Label <string>`

Arguments `<string>`
Labels must be alphanumeric characters within 32 characters.

Examples `:Config:Label "Sample Waveform 1"`
displays “Sample Waveform 1” on the upper left of screen.

Related Commands `:Config:Message`

:Config:Message (No Query Form)

Displays message in red on the upper left of screen.

Syntax `:Config:Message <string>`

Arguments `<string>`
Messages must be alphanumeric characters within 32 characters.

Examples `:Config:Message "Measure the spectrum"`
displays “Measure the spectrum” in red on the upper left of screen.

Related Commands `:Config:Label`

:Config:Mode (No Query Form)

Loads the configuration parameters from the analyzer standard configuration file. For details on the parameter settings, refer to the factory initialization settings of the :Config:Mode command on page B–22.

Syntax

```
:Config:Mode { Dual | Freq1 | Freq2 | Zoom | opCDMA1 |
opCDMA2 | opCDMA3 | opCDMA4 | opCDMA5 | opCDMA6 | opDemod1 |
opCode1 | opCodeW1 | opCCDF | op3gpp1 | op3gpp2 | opGSM1 }
```

Arguments

`Dual` loads the configuration parameters for the Dual mode.

`Freq1` loads the configuration parameters to observe spectrum.

`Freq2` loads the configuration parameters to observe spectrum and spectrogram.

`Zoom` loads the configuration parameters for the Zoom mode.

`opCDMA1` loads the configuration parameters to measure, analyze, and display the EVM (Error Vector Magnitude), ϱ (Rho meter), frequency error, and origin offset error for the IS-95 standard.

`opCDMA2` loads the configuration parameters to measure, analyze, and display the power, occupied bandwidth (OBW), and spurious for the IS-95 standard.

`opCDMA3` loads the configuration parameters to measure, analyze, and display the time characteristic for the burst signal for the IS-95 standard.

`opCDMA4` loads the configuration parameters to measure, analyze, and display the EVM (Error Vector Magnitude), ϱ (Rho meter), frequency error, and origin offset error for the T-53 standard.

`opCDMA5` loads the configuration parameters to measure, analyze, and display the power, occupied bandwidth (OBW), and spurious for the T-53 standard.

`opCDMA6` loads the configuration parameters to measure, analyze, and display the time characteristic for the burst signal for the T-53 standard.

`opDemod1` loads the configuration parameters to observe digital modulating signals.

`opCode1` loads the configuration parameters for the cdmaOne forward-link signal analysis.

`opCodeW1` loads the configuration parameters for the W-CDMA down-link signal analysis.

`opCCDF` loads the configuration parameters for the CCDF analysis.

`op3gpp1` loads the configuration parameters for the 3GPP ACP analysis.

`op3gpp2` loads the configuration parameters for the 3GPP down-link signal analysis.

`opGSM1` loads the configuration parameters for the GSM analysis.

Examples
`:Config:Mode Dual`
loads the configuration file for the Dual mode.

:Config:Mode:DeleteInitFile (No Query Form)

Deletes the power-on setting file.

Syntax `:Config:Mode:DeleteInitFile`

Arguments None

Examples
`:Config:Mode:DeleteInitFile`
deletes the power-on setting file.

Related Commands `:Config:Mode:LoadFromInitFile`, `:Config:Mode:SaveToInitFile`

:Config:Mode:FactoryReset (No Query Form)

Restores the factory default settings.

***NOTE**. This command does not affect the Remote-related settings. (For the Remote commands, refer to page 4–473.)*

Syntax `:Config:Mode:FactoryReset`

Arguments None

Examples
`:Config:Mode:FactoryReset`
restores the factory default settings except the Remote-related settings.

:Config:Mode:Load (No Query Form)

Loads the configuration parameters from the file to which you have stored them previously.

Syntax `:Config:Mode:Load <file_name>`

Arguments `<file_name>::=<string>` specifies the configuration file ("*.CFG").

Examples `:Config:Mode:Load "SAMPLE1.CFG"`
loads the configuration parameters from the file SAMPLE1.CFG.

Related Commands `:Config:Mode:Save`

:Config:Mode:LoadFromInitFile (No Query Form)

Loads the configuration parameters from the power-on setting file.

Syntax `:Config:Mode:LoadFromInitFile`

Arguments None

Examples `:Config:Mode:LoadFromInitFile`
loads the configuration parameters from the power-on file.

Related Commands `:Config:Mode:SaveToInitFile`

:Config:Mode:Save (No Query Form)

Saves the configuration parameters to a file.

Syntax `:Config:Mode:Save <file_name>`

Arguments `<file_name>::=<string>` specifies the configuration file ("*.CFG").

Examples `:Config:Mode:Save "SAMPLE1.CFG"`
saves the parameters to the configuration file SAMPLE1.CFG.

Related Commands `:Config:Mode:Load`

:Config:Mode:SaveToInitFile (No Query Form)

Saves the configuration parameters to the power-on setting file.

Syntax `:Config:Mode:SaveToInitFile`

Arguments None

Examples `:Config:Mode:SaveToInitFile`
saves the configuration parameters to the power-on setting file.

Related Commands `:Config:Mode:LoadFromInitFile`

:Config:Next (No Query Form)

Stops to acquire the current block and starts to acquire the next block.

Syntax `:Config:Next`

Arguments None

Examples `:Config:Next`
stops to acquire the current block and starts to acquire the next block.

:Config:Pause (?)

Turns the Pause mode on or off. In the Pause mode, the data acquisition pauses every block to allow you to process the data. Use the `:Config:Continue` command to resume.

Syntax

```
:Config:Pause { On | Off }
```

```
:Config:Pause?
```

Arguments

`On` turns the Pause mode on.

`Off` turns the Pause mode off.

Examples

`:Config:Pause On`
turns the Pause mode on.

Related Commands

`:Config:Continue`

:Config:PrintScreen (No Query Form)

Prints the screen image to the printer specified with the `:Config:PrintScreen:Printer` command.

Syntax

```
:Config:PrintScreen
```

Arguments

None

Examples

`:Config:PrintScreen`
prints the screen image to the printer.

Related Commands

`:Config:PrintScreen:Printer`

:Config:PrintScreen:Printer (?)

Specifies or queries the printer to print the screen image.

Syntax

```
:Config:PrintScreen:Printer <printer_name>

:Config:PrintScreen:Printer?
```

Arguments

<printer_name>::=<string> specifies the printer.

Examples

```
:Config:PrintScreen:Printer "PRINTER1"
```
specifies that the screen image is output to the printer named PRINTER1.

Related Commands

```
:Config:PrintScreen
```

:Config:PrintScreen:Printer:Items? (Query Only)

Returns the list of available printers.

Syntax

```
:Config:PrintScreen:Printer:Items?
```

Returns

```
<printer_name>[,<printer_name>...]
```
where <printer_name>::=<string>.

Examples

```
:Config:PrintScreen:Printer:Items?
```
might return "PRINTER1","PRINTER2","PRINTER3".

Related Commands

```
:Config:PrintScreen:Printer
```

:Config:PrintScreen:Save (No Query)

Save the screen image to a file.

Syntax `:Config:PrintScreen:Save <file_name>`

Arguments `<file_name>::=<string>` must be “*.BMP” (bitmap file).

Examples `:Config:PrintScreen:Save "C:\SAMPLE1.BMP"`
save the screen image to the file C:\SAMPLE1.BMP.

Related Commands `:Config:PrintScreen`

:Config:Remote (?)

Inhibits the front panel key operation to operate the analyzer remotely.

Syntax `:Config:Remote { On | Off }`

Arguments `On` disables the front panel keys except the LOCAL key for remote operation. Pressing the LOCAL key enables the front panel keys.

`Off` enables the front panel key operation to operate the analyzer locally.

Examples `:Config:Remote On`
disables the front panel keys for remote operation.

:Config:Roll (No Query Form)

Starts data acquisition in the Roll mode. During data acquisition, this command stops it.

Syntax `:Config:Roll`

Arguments None

Examples `:Config:Roll`
starts the data acquisition in the Roll mode. During data acquisition, this command stops it.

Related Commands `:Config:Block`

:Config:Setup (?)

Loads the Setup program for the SETUP menu.

Syntax `:Config:Setup { None | Standard | CDMA }`

`:Config:Setup?`

Arguments `None` disables the SETUP menu.

`Standard` loads the Standard Setup program.

`CDMA` loads the CDMA Setup program.

Examples `:Config:Setup Standard`
loads the Standard Setup program for the SETUP menu.

:Config:Setup:Activate (No Query Form)

Displays the Setup menu at the menu level used last time.

Syntax `:Config:Setup:Activate`

Arguments None

Examples `:Config:Setup:Activate`
displays the Setup menu at the menu level used last time.

Related Commands `:Config:Setup:Show`

:Config:Setup:Items? (Query Only)

Returns the list of available programs for the Setup. Select the program with the `:Config:Setup` command.

Syntax `:Config:Setup:Items?`

Returns <program_name>[,<program_name>...]
where <program_name>::=<string>.

Examples `:Config:Setup:Items?`
might return `None,Standard,CDMA`.

Related Commands `:Config:Setup`

:Config:Setup:Show (No Query Form)

Displays the Setup menu at the top menu level.

Syntax `:Config:Setup:Show`

Arguments None

Examples `:Config:Setup:Show`
displays the Setup menu at the top menu level.

Related Commands `:Config:Setup:Activate`

:Config:Start (No Query Form)

Starts data acquisition in the Block mode. During data acquisition, the analyzer ignores this command.

Syntax
`:Config:Start`

Arguments
None

Examples
`:Config:Start`
starts data acquisition in the Block mode.

Related Commands
`:Config:Stop`

:Config:StartAgain (No Query Form)

Stops the data acquisition and restarts data acquisition in the Block mode.

Syntax
`:Config:StartAgain`

Arguments
None

Examples
`:Config:StartAgain`
stops the data acquisition and restarts data acquisition in the Block mode.

Related Commands
`:Config:Start`, `:Config:Stop`

:Config:Status:Overload? (Query Only)

Queries whether the analyzer's A/D converter overloads.

Syntax `:Config:Status:Overload?`

Returns `On` indicates the analyzer's A/D converter overloads.

`Off` indicates the analyzer's A/D converter does not overload.

Examples `:Config:Status:Overload?`
might return `On` that indicates the analyzer's A/D converter overloads.

:Config:Status:Pause? (Query Only)

Queries whether data acquisition pauses.

Syntax `:Config:Status:Pause?`

Returns `On` indicates data acquisition stops.

`Off` indicates the analyzer is acquiring data.

Examples `:Config:Status:Pause?`
might return `On` that indicates data acquisition stops.

:Config:Status:Triggered? (Query Only)

Queries whether the analyzer is triggered.

Syntax `:Config:Status:Triggered?`

Returns `On` indicates the analyzer is triggered.

`Off` indicates the analyzer is not triggered.

Examples `:Config:Status:Triggered?`
might return `On` that indicates the analyzer is triggered.

:Config:Status:Uncal? (Query Only)

Queries whether the analyzer is in an uncalibrated state due to the increased temperature.

***NOTE**. If the instrument is in the uncal state, perform the self gain-calibration to ensure the normal operation.*

Syntax
`:Config:Status:Uncal?`

Returns
`On` indicates the analyzer is not calibrated.

`Off` indicates the analyzer is calibrated.

Examples
`:Config:Status:Uncal?`
might return `On` which indicates the analyzer is not calibrated.

Related Commands
`:Util1:Execute`

:Config:Stop (No Query Form)

Stops data acquisition. When data acquisition stops, the analyzer ignores this command.

Syntax
`:Config:Stop`

Arguments
None

Examples
`:Config:Stop`
stops the data acquisition.

Related Commands
`:Config:Start`, `:Config:StartAgain`

:Config:System? (Query Only)

Queries the instrument name.

Syntax
:Config:System?

Returns
<NR1>

Examples
:Config:System?
might return WCA380.

:Config:Util:Items? (Query Only)

Returns the list of available programs for the Utility 1 to 8 (A to H). Select the program with the :Config:Util<x> command.

Syntax
:Config:Util:Items?

Returns
<program_name>[,<program_name>...]
where <program_name>::=<string>.

Examples
:Config:Util:Items?
might return None,SelfGainCal,SaveLoad,Average.

Related Commands
:Config:Util<x>

:Config:Util<x> (?)

Assigns the utility program to the **UTILITY** side key.

***NOTE**. UTILITY 1, 2, 3, and 8 are already used; Self Gain-Calibration, Save/Load, Average, and Remote, respectively. You can assign UTILITY 4 to 7 when new utility programs are released.*

Syntax

`:Config:Util<x> { <name> | None }`

`:Config:Util<x>?`

Arguments

`<name>::=<string>` is the utility program name.

`None` disables the **UTILITY** side key.

Examples

`:Config:Util5 SAMPLE`
assigns the utility program SAMPLE to the **UTILITY 5** (UTILITY E) key.

`:Config:Util1 None`
disables the **UTILITY 1** (UTILITY A) key.

:Config:Util<x>:Activate (No Query Form)

Displays the Utility menu at the menu level used last time.

Syntax

`:Config:Util<x>:Activate`

Arguments

None

Examples

`:Config:Util1:Activate`
displays the Utility 1 (self gain-calibration) menu at the menu level used last time.

Related Commands

`:Config:Util<x>:Show`

:Config:Util<x>:Show (No Query Form)

Displays the Utility menu at the top menu level.

Syntax

`:Config:Util<x>:Show`

Arguments

None

Examples

`:Config:Util1:Show`
displays the Utility 1 (self gain-calibration) menu at the top menu level.

Related Commands

`:Config:Util<x>:Activate`

:Config:Version? (Query Only)

Queries the version of the Configuration program.

Syntax

`:Config:Version?`

Returns

`<NR2>`

Examples

`:Config:Version?`
might return `1.1`.

:Config:View:BackgroundColor (?)

Selects or queries the background color of the waveform display area.

Syntax

`:Config:View:BackgroundColor { Black | White }`

`:Config:View:BackgroundColor?`

Arguments

`Black` selects black for the background color.

`White` selects white for the background color.

Examples

`:Config:View:BackgroundColor Black`
selects black for the background color.

:Config:View:Items? (Query Only)

Returns the list of available programs for the View 1 to 8 (A to H). Select the program with the `:Config:View<x>` command.

Syntax `:Config:View:Items?`

Returns `<program_name>[,<program_name>...]`
where `<program_name>::=<string>`.

Examples `:Config:View:Items?`
might return `None,Waveform,Spectrogram,Waterfall`.

Related Commands `:Config:View<x>`, `:Config:View<x>:Show`

:Config:View:MarkerLink (?)

Determines whether the markers in different views move together in unison or separately.

Syntax `:Config:View:MarkerLink { On | Off }`

`:Config:View:MarkerLink?`

Arguments `On` ties the markers in different views together.

`Off` frees the markers in different views to move separately.

Examples `:Config:View:MarkerLink On`
specifies that the markers in different views move in unison.

:Config:View:Style (?)

Specifies or queries the layout of the views on screen.

Syntax

```
:Config:View:Style { OneByOne | OneByTwo | OneByFour | TwoByTwo }
```

```
:Config:View:Style?
```

Arguments

`OneByOne` specifies that one view displays on the screen.

`OneByTwo` selects 1 × 2 view display.

`OneByFour` selects 1 × 4 view display.

`TwoByTwo` selects 2 × 2 view display.

Examples

`:Config:View:Style TwoByTwo`
selects 2 × 2 view display.

:Config:View<x> (?)

Assigns the View program to the **VIEW** key.

Syntax

```
:Config:View<x> { None | Waveform | Analog | FSK | Spectrogram |
Waterfall | Polar | EyeDiagram | SymbolTable | EVM |
CDMAWaveform | CDMAPolar | CDMATime |
CodeSpectrogram | CodePolar | CodePower |
CodeWSpectrogram | CodeWPolar | CodeWPower |
3gppACPView | 3gppSpectrogram | 3gppPolar | 3gppPower |
GSM | CCDF | CCDFView }
```

```
:Config:View<x>?
```

Arguments

Each argument represents the system program for the specific measurement. For detail settings, refer to the section starting on the page listed below.

Table 4–41: :Config:View<x> command arguments

Argument	System program	Display form	Detail settings
None	None	None	–
Waveform	Waveform View	Spectrum	p. 4–119
Analog	Analog View	Time domain for AM/FM/PM	p. 4–157
FSK	FSK View	Time domain for FSK	p. 4–169
Spectrogram	Spectrogram View	Spectrogram	p. 4–179
Waterfall	Waterfall View	Waterfall	p. 4–195
Polar	Polar View	Polar	p. 4–207
EyeDiagram	Eye Diagram View	Eye Diagram	p. 4–227
SymbolTable	Symbol Table View	Symbol Table	p. 4–231
EVM	EVM View	Error vector magnitude	p. 4–237
CDMAWaveform	CDMA Waveform View	Spectrum for IS-95/T-53	p. 4–249
CDMAPolar	CDMA Polar View	Polar for IS-95/T-53	p. 4–267
CDMATime	CDMA TIme View	Time domain for IS-95/T-53	p. 4–277
CodeSpectrogram	CodeSpectrogram View	Spectrogram (cdmaOne)	p. 4–295
CodePolar	CodePolar View	Polar (cdmaOne)	p. 4–305
CodePower	CodePower View	Spectrum (cdmaOne)	p. 4–313
CodeWSpectrogram	CodeWSpectrogram View	Spectrogram (cdmaOne)	p. 4–325
CodeWPolar	CodeWPolar View	Polar (cdmaOne)	p. 4–339
CodeWPower	CodeWPower View	Spectrum (cdmaOne)	p. 4–351
3gppACPView	Waveform View	ACP measurement (3GPP)	p. 4–371
3gppSpectrogram	CodeWSpectrogram View	Spectrogram (3GPP)	p. 4–385
3gppPolar	CodeWPolar View	Polar (3GPP)	p. 4–393
3gppPower	CodeWPower View	Spectrum (3GPP)	p. 4–401
GSM	GSM View	Script (GSM)	p. 4–411
CCDF	CCDF	Time domain for CCDF	p. 4–429
CCDFView	CCDFView	CCDF	p. 4–441

***NOTE**. The GSM view is designed to work only in View 4 (View D) for the GSM analysis. When you perform GSM measurement, select* opGSM1 *with the* `:Config:Mode` *command or define View 4 as* GSM.

Examples `:Config:View1 Waveform`
assigns the Waveform View program to the **VIEW 1** (VIEW A) key.

Related Commands `:Config:View<x>:Items`

:Config:View<x>:Activate (No Query Form)

Displays the View menu at the menu level used last time.

Syntax `:Config:View<x>:Activate`

Arguments None

Examples `:Config:View1:Activate`
displays the View A menu at the menu level used last time.

Related Commands `:Config:View<x>:Show`

:Config:View<x>:BringToFront (No Query Form)

Puts the waveform display area of the specified View in front on screen.

Syntax `:Config:View<x>:BringToFront`

Arguments None

Examples `:Config:View1:BringToFront`
puts the waveform display area of View A in front on screen.

:Config:View<x>:Show (No Query Form)

Displays the View menu at the top menu level.

Syntax

```
:Config:View<x>:Show
```

Arguments

None

Examples

```
:Config:View1:Show
```

displays the View A menu at the top menu level.

Related Commands

```
:Config:View<x>:Activate
```

Standard Setup Commands

When you select `Standard` in the `Config:Setup` command, use the commands in this section to set the details for the standard setup.

:Setup:Band (?)

Selects the input frequency band.

Syntax

```
:Setup:Band { Baseband | RF | RF1 | RF2 | RF3 | RF4 | IQ }
```

```
:Setup:Band?
```

Arguments Each argument represents the following band.

Argument	Input frequency range
Baseband	0 to 10 MHz
RF (WCA330) / RF1 (WCA380)	10 MHz to 3 GHz (IF mode: Normal, HiRes) 50 MHz to 3 GHz (IF mode: Wide)
RF2 (WCA380)	2.5 GHz to 3.5 GHz
RF3 (WCA380)	3.5 GHz to 6.5 GHz
RF4 (WCA380)	5.0 GHz to 8.0 GHz
IQ	Use the IQ signals input from the rear connectors.

Examples

```
:Setup:Band Baseband
```

selects Baseband.

:Setup:BlockSize (?)

Sets or queries the number of frames per block.

Syntax

```
:Setup:BlockSize <value>
```

```
:Setup:BlockSize?
```

Arguments `<value>::=<NR1>` depends on the FFT points and the memory mode:

FFT points	Memory mode	<value>
256	–	1 to 16000
1024	Frequency	1 to 4000
	Other than Frequency	1 to 2000

Examples

`:Setup:BlockSize 200`
sets the number of frames per block to 200.

Related Commands

`:Setup:FFTPoints`, `:Setup:MemoryMode`

:Setup:CenterFrequency (?)

Sets or queries center frequency. Setting center frequency reflects the current span setting.

Syntax

`:Setup:CenterFrequency <value>`

`:Setup:CenterFrequency?`

Arguments

`<value>::=<NR3>` depends on the input frequency band (:Setup:Band).

Argument	Center frequency range
Baseband	Span/2 Hz to 10 MHz – Span/2 Hz
RF (WCA330) / RF1 (WCA380)	Span/2 Hz to 3 GHz – Span/2 Hz
RF2 (WCA380)	2.5 GHz + Span/2 to 3.5 GHz – Span/2 Hz
RF3 (WCA380)	3.5 GHz + Span/2 to 6.5 GHz – Span/2 Hz
RF4 (WCA380)	5.0 GHz + Span/2 to 8.0 GHz – Span/2 Hz

Examples

`:Setup:CenterFrequency 1.5G`
sets the center frequency to 1.5 GHz.

Related Commands

`:Setup:Band`

:Setup:FFTPoints (?)

Sets or queries the number of FFT sampling points per frame.

Syntax

```
:Setup:FFTPoints <value>
```

```
:Setup:FFTPoints?
```

Arguments

`<value>::=<NR1>` depends on the memory mode:

Memory mode	<value>
Frequency	{ 1024 | 256 }
Other than Frequency	1024

Examples

`:Setup:FFTPoints 1024`
sets the number of FFT points to 1024.

Related Commands

`:Setup:MemoryMode`

:Setup:FFTType (?)

Selects or queries the FFT type: either hardware or software processing. For details on the FFT type, refer to the user manual.

Syntax

```
:Setup:FFTType { HW | SW }
```

```
:Setup:FFTType?
```

Arguments

`HW` selects the hardware FFT processing. FFT is usually performed by hardware. It is faster than the software processing.

`SW` selects the FFT software processing. It uses floating decimal numbers to perform FFT, resulting in better accuracy but a lower speed.

Examples

`:Setup:FFTType HW`
selects the hardware FFT processing.

:Setup:FFTWindow (?)

Selects or queries the FFT window.

Syntax

```
:Setup:FFTWindow { Blackman | Hamming | Rect }
```

```
:Setup:FFTWindow?
```

Arguments

`Blackman` selects the Blackman-Harris window.

`Hamming` selects the Hamming window.

`Rect` selects the Rectangular window.

Examples

`:Setup:FFTWindow Blackman`
selects the Blackman-Harris window.

:Setup:FramePeriod (?)

Sets or queries the frame period in the Block mode. The setting is effective in the Frequency and Dual memory mode. In the Zoom mode, this setting is ignored.

Syntax

```
:Setup:FramePeriod <value>
```

```
:Setup:FramePeriod?
```

Arguments

`<value>::=<NR3>` ranges 20 μs to 60 s.

Examples

`:Setup:FramePeriod 30m`
sets the frame period to 30 ms.

Related Commands

`:Setup:MemoryMode`

:Setup:FrequencyOffset (?)

Sets or queries the frequency display offset. This setting is useful, for example, when a down converter connects externally. It has no effect on the analyzer hardware settings.

Syntax

`FrequencyOffset <value>`

`FrequencyOffset?`

Arguments

`<value>::=<NR3>` ranges –100 GHz to 100 GHz.

Examples

`:Setup:FrequencyOffset 100G`
sets the frequency display offset to 100 GHz.

Related Commands

`:Setup:CenterFrequency`

:Setup:IFMode (?)

Selects the intermediate frequency (IF) mode. For details on the IF mode, refer to the user manual.

Syntax

`:Setup:IFMode { Normal | HiRes | Wide }`

`:Setup:IFMode?`

Arguments

`Normal` has an IF bandwidth of 10 MHz and features a high degree of phase flatness. It is suitable for digital modulation analysis with a span below 6 MHz, or general measurement that does not require a wide dynamic range.

`HiRes` has a relatively narrow IF bandwidth of 6 MHz, but has the widest dynamic range. It is suitable for ACP and spurious measurement. FFT is slow because it is always performed by software.

`Wide` has an IF bandwidth of 32 MHz, which is the widest of all the three modes. It is suitable for modulation analysis of wide bandwidth signals or code-domain analysis of W-CDMA. A span can be expanded to maximum 30 MHz per frame.

Examples

`:Setup:IFMode Normal`
selects the Normal IF mode.

:Setup:InputCoupling (?)

Selects or queries the input coupling to the RF INPUT connector on the front panel. In the IQ mode, this command is ignored.

Syntax

`:Setup:InputCoupling <value>`

`:Setup:InputCoupling?`

Arguments

`<value>` depends on the input frequency band (:Setup:Band):

Input mode	Input coupling
Baseband	{ AC | GND | DC }
RF (WCA330) / RF 1 to 4 (WCA380)	AC
IQ	{ AC | DC }

Examples

`:Setup:InputCoupling AC`
sets the RF input coupling to AC.

Related Commands

`:Setup:Band`

:Setup:LevelOffset (?)

Sets or queries the level display offset. This setting is useful, for example, when an attenuator connects externally. It has no effect on the analyzer hardware settings.

Syntax

`:Setup:LevelOffset <value>`

`:Setup:LevelOffset?`

Arguments

`<value>::=<NR3>` ranges –100 dB to 100 dB.

Examples

`:Setup:LevelOffset 100`
sets the level display offset to 100 dB.

Related Commands

`:Setup:ReferenceLevel`

:Setup:Load (No Query Form)

Loads the trigger mask from the specified file.

Syntax

```
:Setup:Load <file_name>
```

Arguments

<file_name>::=<string> specifies the file that contains the trigger mask. The file name must be “*.TRG”.

Examples

`:Setup:Load "SAMPLE1.TRG"`
loads the trigger mask from the file SAMPLE1.TRG.

Related Commands

`:Setup:Save`

:Setup:Manual (?)

Determines which setting has priority, a mixer level or an RF attenuator.

Syntax

```
:Setup:Manual { Mixer | RFAtt }
```

```
:Setup:Manual?
```

Arguments

`Mixer` prioritizes the mixer level setting. The RF attenuator is set automatically.

`RFAtt` prioritizes the RF attenuator setting. The mixer level is set automatically.

Examples

`:Setup:Manual Mixer`
prioritizes the mixer level setting.

Related Commands

`:Setup:MixerLevel`, `:Setup:RFAtt`

:Setup:MarkerToFreq (No Query Form)

Sets the center frequency to the value at the marker.

Syntax `:Setup:MarkerToFreq`

Arguments None

Examples `:Setup:MarkerToFreq`
sets the center frequency to the value at the marker.

Related Commands `:Setup:CenterFrequency`

:Setup:MaxSpan (No Query Form)

Sets the span to the maximum.

Syntax `:Setup:MaxSpan`

Arguments None

Examples `:Setup:MaxSpan`
sets the span to the maximum.

Related Commands `:Setup:Span`

:Setup:MemoryMode (?)

Selects or queries the memory mode.

Syntax

`:Setup:MemoryMode { Frequency | Dual | Zoom }`

`:Setup:MemoryMode?`

Arguments

`Frequency` selects the Frequency mode. Only frequency domain data is written into the memory.

`Dual` selects the Dual mode. The data for both frequency and time domain are written into the memory simultaneously.

`Zoom` expands a specific part of the spectrum.

In the IQ mode, the memory mode is fixed to the Zoom.

Examples

`:Setup:MemoryMode Frequency`
sets the memory mode to Frequency.

Related Commands

`:Setup:BlockSize`, `:Setup:FFTPoints`, `:Setup:FramePeriod`

:Setup:MixerLevel (?)

Selects or queries the mixer level.

Syntax

`:Setup:MixerLevel <value>`

`:Setup:MixerLevel?`

Arguments

`<value>::=<NR1>` depends on the input frequency band (:Setup:Band).

Argument	Mixer level (dBm)
RF (WCA330) / RF1, RF2 (WCA380)	–5, –10, –15, –20, –25
RF3, RF4 (WCA380)	–5, –15, –25

Examples

`:Setup:MixerLevel -5`
sets the mixer level to –5 dBm.

Related Commands :Setup:Band, :Setup:Manual

:Setup:ReferenceLevel (?)

Sets or queries the reference level.

Syntax

```
:Setup:ReferenceLevel <value>
```

```
:Setup:ReferenceLevel?
```

Arguments `<value>::=<NR1>` sets the reference level in 1 dB step. The value depends on the input frequency band (:Setup:Band).

Argument	Reference level (dBm)
RF (WCA330) / RF 1 to 4 (WCA380)	–50 to +30 [1]
Baseband	–30 to +30

[1] **The maximum and the minimum depend on the MixerLevel and RFAtt settings.**

Examples `:Setup:ReferenceLevel 30`
sets the reference level to 30 dBm.

Related Commands :Setup:Band, :Setup:LevelOffset, :Setup:MixerLevel, :Setup:RFAtt

:Setup:ReferenceOsc (?)

Selects or queries the reference oscillator.

Syntax

```
:Setup:ReferenceOsc { Internal | External }
```

```
:Setup:ReferenceOsc?
```

Arguments `Internal` selects the internal reference oscillator. It generates 10 MHz sine wave.

`External` selects the external reference oscillator. It is connected to the 10 MHz REF INPUT connector on the rear panel.

Examples `:Setup:ReferenceOsc Internal`
selects the internal reference oscillator.

:Setup:RFAtt (?)

Selects or queries the RF attenuator setting.

Syntax

`:Setup:RFAtt <value>`

`:Setup:MixerLevel?`

Arguments

`<value>::=<NR1>` depends on the input frequency band (:Setup:Band).

Argument	RF attenuator (dB)
RF (WCA330 RF1, RF2 (WCA380)	{ 50 | 47 | 45 | 42 | 40 | 37 | 35 | 32 | 20 | 27 | 25 | 22 | 20 | 17 | 15 | 12 | 10 | 7 | 5 | 2 | 0 }
RF3, RF4 (WCA380)	{ 50 | 40 | 30 | 20 | 10 | 0 }

Examples

`:Setup:RFAtt 0`
sets the RF attenuator to 0 dB.

Related Commands

`:Setup:Band`, `:Setup:Manual`

:Setup:Save (No Query Form)

Saves the trigger mask to the specified file.

Syntax

`:Setup:Save <file_name>`

Arguments

`<file_name>::=<string>` specifies the file to save the trigger mask.
The file name must be "*.TRG".

Examples

`:Setup:Save "SAMPLE1.TRG"`
saves the trigger mask to the file SAMPLE1.TRG.

Related Commands

`:Setup:Load`

:Setup:Span (?)

Sets or queries the span.

Syntax

```
:Setup:Span <value>
```

```
:Setup:Span?
```

Arguments

`<value>` depends on the input mode and the memory mode:

Input mode	Memory mode	<value>
Baseband	Other than Zoom	{ 10M | 6M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
	Zoom	{ 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
RF	Other than Zoom	{ 3G | 2G | 1G | 500M | 200M | 100M | 50M | 30M | 20M | 10M | 6M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
	Zoom	{ 3G | 2G | 1G | 500M | 200M | 100M | 50M | 30M | 20M | 10M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
Wide, IQ	–	{ 30M | 20M | 10M }

Examples

```
:Setup:Span 1M
```
sets the span to 1 MHz.

Related Commands

`:Setup:Band`, `:Setup:MemoryMode`

:Setup:Trigger (?)

Selects or queries the trigger mode. For details on trigger, refer to the user manual.

Syntax

```
:Setup:Trigger { Auto | Delayed | Interval | Never | Normal |
Quick | QuickInterval | Timeout }
```

```
:Setup:Trigger?
```

Arguments

`Auto` acquires data regardless of trigger generation as follows:
If you have turned off the trigger count, the data acquisition will repeat until you stop it.
If you have turned on the trigger count, the data will be acquired the number of times specified with the `:Setup:TriggerTimes` command after you start the acquisition.

`Delayed` causes the data acquisition to stop for the time after the trigger specified with the `:Setup:TriggerDelayed` command.

`Interval` specifies that a block data is acquired and displayed at the time interval set with the `:Setup:TriggerInterval` command.

`Never` ignores any trigger settings. The data acquisition repeats until you stop it.

`Normal` waits for a valid trigger event.

`Quick` is the same as the `Normal` except that data is displayed after all blocks are acquired. It shortens the time interval between two block data acquisitions. In the Normal mode, the interval is several decade milli-seconds. In the Quick mode, it is several hundred micro-seconds.

`QuickInterval` is the same as the `Interval`, except that data is displayed after all blocks are acquired. You can capture phenomena which are missed during data display in the `Interval` mode.

`Timeout` stops the data acquisition if the trigger event does not occur within the time specified with the `:Setup:TriggerTimeout` command. This argument is effective only when the trigger source is set to Internal.

***NOTE**. In the IQ and Wide modes, you can not select* `Quick` *and* `QuickInterval`.

Examples

`:Setup:Trigger Auto`
selects the Auto trigger mode.

Related Commands

`:Config:Block`, `:Config:Start`, `:Config:Stop`, `:Setup:TriggerCount`, `:Setup:TriggerDelayed`, `:Setup:TriggerDomain`, `:Setup:TriggerInterval`, `:Setup:TriggerPosition`, `:Setup:TriggerSlope`, `:Setup:TriggerSource`, `:Setup:TriggerTimeout`, `:Setup:TriggerTimes`

:Setup:TriggerCount (?)

Turns the trigger counter on or off.

Syntax

`:Setup:TriggerCount { On | Off }`

`:Setup:TriggerCount?`

Arguments

`On` turns the trigger counter on.

`Off` turns the trigger counter off.

Examples

`:Setup:TriggerCount On`
turns the trigger counter on.

Related Commands

`:Setup:Trigger`, `:Setup:TriggerTimes`

:Setup:TriggerDelayed (?)

Sets or queries the delay time for the Delayed Trigger mode.

Syntax

`:Setup:TriggerDelayed <value>`

`:Setup:TriggerDelayed?`

Arguments

`<value>::=<NR3>` specifies the delay time. The range is 0 to 60 s.

Examples

`:Setup:TriggerDelayed 100m`
sets the trigger delay time to 100 ms.

Related Commands

`:Setup:Trigger`

:Setup:TriggerDomain (?)

Sets or queries the domain in which the analyzer is triggered.

Syntax

`:Setup:TriggerDomain { Frequency | Time }`

`:Setup:TriggerDomain?`

Arguments

`Frequency` selects the frequency domain for trigger.

`Time` selects the time domain for trigger.

Examples

`:Setup:TriggerDomain Frequency`
selects the frequency domain for trigger.

Related Commands

`:Setup:Trigger`

:Setup:TriggerInterval (?)

Sets or queries the time interval for the Interval and Quick Interval trigger modes.

Syntax

```
:Setup:TriggerInterval <value>
:Setup:TriggerInterval?
```

Arguments

`<value>::=<NR3>` ranges 1 s to 3600 s.

Examples

`:Setup:TriggerInterval 3000`
sets the time interval for the Interval and Quick Interval trigger modes to 3000 s.

Related Commands

`:Setup:Trigger`

:Setup:TriggerPosition (?)

Sets or queries the trigger position.

Syntax

```
:Setup:TriggerPosition <value>
:Setup:TriggerPosition?
```

Arguments

`<value>::=<NR2>` ranges 0 to 100 % in 1 % step.
The trigger position represents the ratio of the number of frames preceding the trigger generation to that of one block.

Examples

`:Setup:TriggerPosition 10`
For example, if the block size is 1,000 frames, 100 frames will be acquired before the trigger generation, and 900 frames after it.

Related Commands

`:Setup:Trigger`

:Setup:TriggerSlope (?)

Selects or queries a rising or falling edge for the trigger.

Syntax

```
:Setup:TriggerSlope { Rise | Fall }
```

```
:Setup:TriggerSlope?
```

Arguments

`Rise` specifies to trigger on the rising or positive edge of a signal.

`Fall` specifies to trigger on the falling or negative edge of a signal.

Examples

`:Setup:TriggerSlope Rise`
specifies to trigger on the rising edge of a signal.

Related Commands

`:Setup:Trigger`, `:Setup:TriggerSource`

:Setup:TriggerSource (?)

Selects or queries the trigger source.

Syntax

```
:Setup:TriggerSource { Internal | External }
```

```
:Setup:TriggerSource?
```

Arguments

`Internal` uses the trigger mask pattern stored in the internal register for trigger generation.

`External` uses the signal from the EXT TRIG connector on the front panel as a trigger input.

In the IQ mode, the source is fixed to External.

Examples

`:Setup:TriggerSource Internal`
uses the trigger mask pattern stored in the internal register as the trigger source.

Related Commands

`:Setup:Trigger`

:Setup:TriggerTimeout (?)

Sets or queries the time-out value in the Timeout trigger mode.

Syntax

```
:Setup:TriggerTimeout <value>
```

```
:Setup:TriggerTimeout?
```

Arguments

`<value>::=<NR3>` specifies the time-out value. The range is 0 to 60 s.

Examples

`:Setup:TriggerTimeout 30`
sets the time-out to 30 s.

Related Commands

`:Setup:Trigger`

:Setup:TriggerTimes (?)

Sets or queries the trigger count for the Count trigger mode.

Syntax

```
:Setup:TriggerTimes <value>
```

```
:Setup:TriggerTimes?
```

Arguments

`<value>::=<NR1>` depends on the FFT points and the memory mode:

FFT points	Memory mode	<value>
256	–	1 to 16000/(Block size)
1024	Frequency	1 to 4000/(Block size)
	Other than Frequency	1 to 2000/(Block size)

Examples

`:Setup:TriggerTimes 10`
sets the trigger count to 10.

Related Commands

`:Setup:Trigger`

:Setup:Version? (Query Only)

Queries the version of the Setup program.

Syntax `:Setup:Version?`

Returns <NR2> indicates the version number.

Examples `:Setup:Version?`
might return `1.1`.

:Setup:Zoom:Execute (No Query Form)

Zooms waveform with the expansion factor specified by the `:Setup:Zoom:Mag` command.

Syntax `:Setup:Zoom:Execute`

Arguments None

Examples `:Setup:Zoom:Execute`
zooms waveform.

Related Commands `:Setup:MemoryMode`, `:Setup:Zoom:Frequency`, `:Setup:Zoom:Mag`

:Setup:Zoom:FFTType (?)

Selects the FFT type in the Zoom mode: either hardware or software processing. For details on the FFT type, refer to the *User Manual*.

Syntax

`:Setup:Zoom:FFTType { HW | SW }`

`:Setup:Zoom:FFTType?`

Arguments

`HW` selects the hardware FFT processing. FFT is usually performed by hardware. It is faster than the software processing.

`SW` selects the FFT software processing. It uses floating decimal numbers to perform FFT, resulting in better accuracy but a lower speed.

Examples

`:Setup:Zoom:FFTType HW`
selects the hardware FFT processing in the Zoom mode.

:Setup:Zoom:FFTWindow (?)

Selects or queries the FFT window in the Zoom mode.
For details on the FFT window, refer to the *User Manual*.

Syntax

`:Setup:Zoom:FFTWindow { Blackman | Hamming | Rect }`

`:Setup:Zoom:FFTWindow?`

Arguments

`Blackman` selects the Blackman-Harris window.

`Hamming` selects the Hamming window.

`Rect` selects the Rectangular window.

Examples

`:Setup:Zoom:FFTWindow Blackman`
selects the Blackman-Harris window in the Zoom mode.

:Setup:Zoom:Frequency (?)

Sets or queries the center frequency for the zoom.

Syntax

```
:Setup:Zoom:Frequency <value>
```

```
:Setup:Zoom:Frequency?
```

Arguments

`<value>::=<NR3>` specifies the center frequency for the zoom.

Examples

`:Setup:Zoom:Frequency 1.5G`
sets the center frequency for the zoom to 1.5 GHz.

Related Commands

`:Setup:Zoom:Execute`, `:Setup:Zoom:Mag`

:Setup:Zoom:Mag (?)

Sets or queries the expansion factor in the Zoom mode.

Syntax

```
:Setup:Zoom:Mag <value>
```

```
:Setup:Zoom:Mag?
```

Arguments

`<value>::=<NR1>` is the expansion factor. It depends on the span.

Span before zoom	Expansion factor
5 MHz	2 | 5 | 10 | 20 | 50 | 100 | 200 | 500 | 1000
Other than 5 MHz	2 | 4 | 10 | 20 | 40 | 100 | 200 | 400 | 1000

In IQ and Wide modes, the expansion factor is the same as in 5 MHz span.

Examples

`:Setup:Zoom:Mag 10`
magnifies the display 10 times.

Related Commands

`:Setup:Zoom:Execute`, `:Setup:Zoom:Frequency`

CDMA Setup Commands

When you select `CDMA` in the `Config:Setup` command, use the commands in this section to set the details for the CDMA setup.

:Setup:BlockSize (?)

Sets or queries the number of frames per block.

Syntax

```
:Setup:BlockSize <value>
```

```
:Setup:BlockSize?
```

Arguments

`<value>::=<NR1>` depends on the FFT points and the memory mode:

FFT points	Memory mode	<value>
256	–	1 to 16000
1024	Frequency	1 to 4000
	Other than Frequency	1 to 2000

Examples

`:Setup:BlockSize 200`
sets the number of frames per block to 200.

Related Commands

`:Setup:FFTPoints`, `:Setup:MemoryMode`

:Setup:CDMA:Channel (?)

Selects or queries the channel.

Syntax

```
:Setup:CDMA:Channel <value>
```

```
:Setup:CDMA:Channel?
```

Arguments

`<value>::=<NR1>` specifies the channel number.

For the IS-95 standard, the range is 1 to 777. Channel 1 and 777 correspond to 825.03 and 848.31 MHz, respectively. The frequency difference between two adjacent channels is 0.03 MHz.

For the T-53 standard, the range is 1 to 1199. Channel 1 and 1199 correspond to 915.0125 and 888.9875 MHz, respectively. The frequency difference between two adjacent channels is 0.0125 MHz.

Examples

`:Setup:CDMA:Channel 777`
sets the channel number to 777.

Related Commands

`:Setup:CDMA:Standard`

:Setup:CDMA:Span30M (No Query Form)

Sets the span to 30 MHz. This command is used for the IS-95 standard.

Syntax

```
:Setup:CDMA:Span30M
```

Arguments

None

Examples

`:Setup:CDMA:Span30M`
sets the span to 30 MHz.

Related Commands

`:Setup:Span`

:Setup:CDMA:Span50M (No Query Form)

Sets the span to 50 MHz. This command is used for the T-53 standard.

Syntax
`:Setup:CDMA:Span50M`

Arguments
None

Examples
`:Setup:CDMA:Span50M`
sets the span to 50 MHz.

Related Commands
`:Setup:Span`

:Setup:CDMA:Span5MAuto (No Query Form)

Sets the span to 5 MHz and the trigger mode to Auto.

Syntax
`:Setup:CDMA:Span5MAuto`

Arguments
None

Examples
`:Setup:CDMA:Span5MAuto`
sets the span to 5 MHz and the trigger mode to Auto.

Related Commands
`:Setup:Span`

:Setup:CDMA:Span5MNormal (No Query Form)

Sets the span to 5 MHz and the trigger mode to Normal.

Syntax
`:Setup:CDMA:Span5MNormal`

Arguments
None

Examples
`:Setup:CDMA:Span5MNormal`
sets the span to 5 MHz and the trigger mode to Normal.

Related Commands
`:Setup:Span`

:Setup:CDMA:Standard (?)

Selects or queries the CDMA standard.

Syntax
`:Setup:CDMA:Standard { IS95 | T53 }`

`:Setup:CDMA:Standard?`

Arguments
`IS95` selects the IS-95 standard.

`T53` selects the T-53 standard.

Examples
`:Setup:CDMA:Standard IS95`
selects the IS-95 standard.

:Setup:CDMA:TriggerLevel (?)

Sets or queries the trigger level in the time domain.

Syntax

`:Setup:CDMA:TriggerLevel <value>`

`:Setup:CDMA:TriggerLevel?`

Arguments

`<value>::=<NR2>` specifies the trigger level. The range is –40 dB to 0 dB.

Examples

`:Setup:CDMA:TriggerLevel 0`
sets the trigger level to 0 dB.

:Setup:MarkerToFreq (No Query Form)

Sets the center frequency to the value at the marker.

Syntax

`:Setup:MarkerToFreq`

Arguments

None

Examples

`:Setup:MarkerToFreq`
sets the center frequency to the value at the marker.

:Setup:MaxSpan (No Query Form)

Sets the span to the maximum.

Syntax

`:Setup:MaxSpan`

Arguments

None

Examples

`:Setup:MaxSpan`
sets the span to the maximum.

Related Commands

`:Setup:Span`

:Setup:ReferenceLevel (?)

Sets or queries the reference level.

Syntax

```
:Setup:ReferenceLevel <value>
:Setup:ReferenceLevel?
```

Arguments

`<value>::=<NR1>` specifies the reference level.
The range is –50 dBm to 30 dBm.

Examples

`:Setup:ReferenceLevel 30`
sets the reference level to 30 dBm.

:Setup:ReferenceOsc (?)

Selects or queries the reference oscillator.

Syntax

```
:Setup:ReferenceOsc { Internal | External }
:Setup:ReferenceOsc?
```

Arguments

`Internal` selects the internal reference oscillator. It generates 10 MHz sine wave.

`External` selects the external reference oscillator. It is connected to the 10 MHz REF INPUT connector on the rear panel.

Examples

`:Setup:ReferenceOsc Internal`
selects the internal reference oscillator.

:Setup:Span (?)

Sets or queries the span.

Syntax

```
:Setup:Span <value>
```

```
:Setup:Span?
```

Arguments

`<value>` specifies the span. It depends on the input mode and the memory mode:

Input mode	Memory mode	Span
RF (WCA330) RF1 to RF4 (WCA380)	Other than Zoom	{ 3G | 2G | 1G | 500M | 200M | 100M | 50M | 30M | 20M | 10M | 6M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
	Zoom	{ 3G | 2G | 1G | 500M | 200M | 100M | 50M | 30M | 20M | 10M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
Baseband	Other than Zoom	{ 10M | 6M | 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }
	Zoom	{ 5M | 2M | 1M | 500k | 200k | 100k | 50k | 20k | 10k | 5k | 2k | 1k | 500 | 200 | 100 }

Examples

`:Setup:Span 5M`
sets the span to 5 MHz.

Related Commands

`:Setup:CDMA:Span30M`, `:Setup:CDMA:Span5MAuto`, `:Setup:CDMA:Span5MNormal`, `:Setup:MaxSpan`

:Setup:Trigger (?)

Selects or queries the trigger mode.

The command usage is the same as that of the :Setup:Trigger command in the Standard Setup menu. Refer to page 4–92.

:Setup:TriggerCount (?)

Turns the trigger counter on or off.

The command usage is the same as that of the :Setup:TriggerCount command in the Standard Setup menu. Refer to page 4–93.

:Setup:TriggerDelayed (?)

Sets or queries the delay time for the Delayed trigger mode.

The command usage is the same as that of the :Setup:TriggerDelayed command in the Standard Setup menu. Refer to page 4–94.

:Setup:TriggerDomain (?)

Sets or queries the domain in which the analyzer is triggered.

The command usage is the same as that of the :Setup:TriggerDomain command in the Standard Setup menu. Refer to page 4–94.

:Setup:TriggerInterval (?)

Sets or queries the time interval for the Interval and Quick Interval trigger modes.

The command usage is the same as that of the :Setup:TriggerInterval command in the Standard Setup menu. Refer to page 4–95.

:Setup:TriggerPosition (?)

Sets or query the trigger position.

The command usage is the same as that of the :Setup:TriggerPosition command in the Standard Setup menu. Refer to page 4–95.

:Setup:TriggerSlope (?)

Selects either a rising or falling edge for the trigger.

The command usage is the same as that of the :Setup:TriggerSlope command in the Standard Setup menu. Refer to page 4–96.

:Setup:TriggerSource (?)

Selects or queries the trigger source.

The command usage is the same as that of the :Setup:TriggerSource command in the Standard Setup menu. Refer to page 4–96.

:Setup:TriggerTimeout (?)

Sets or queries the time-out value in the Timeout trigger mode.

The command usage is the same as that of the :Setup:TriggerTimeout command in the Standard Setup menu. Refer to page 4–97.

:Setup:TriggerTimes (?)

Sets or queries the trigger count in the Count trigger mode.

The command usage is the same as that of the :Setup:TriggerTimes command in the Standard Setup menu. Refer to page 4–97.

:Setup:Version? (Query Only)

Queries the version of the CDMA Setup program.

Syntax

`:Setup:Version?`

Returns

`<NR2>` indicates the version number.

Examples

`:Setup:Version?`
might return `1.1`.

3gppACP Setup Commands

When you select `3gppACP` in the `Config:Setup` command, use the commands in this section to set the details for the ACP measurement according to the 3GPP standard.

:Setup:ACP:BlockSize (?)

Sets or queries the number of sweeps when pressing the **BLOCK** key.

Syntax

```
:Setup:ACP:BlockSize <value>
:Setup:ACP:BlockSize?
```

Arguments `<value>::=<NR1>` ranges 1 to 1,000,000.

Examples `:Setup:ACP:BlockSize 1000000`
sets the number of sweeps to 1000000 in the Block mode.

:Setup:ACP:CarrierWidth (?)

Sets or queries the carrier bandwidth. The IF filter is not used for the specified range.

Syntax

```
:Setup:ACP:CarrierWidth <value>
:Setup:ACP:CarrierWidth?
```

Arguments `<Value>::=<NR3>` ranges 1 MHz to 10 MHz.

Examples `:Setup:ACP:CarrierWidth 10M`
sets the carrier bandwidth to 10 MHz.

:Setup:ACP:CenterFrequency (?)

Sets or queries the center frequency.

Syntax

```
ACP:CenterFrequency <value>
ACP:CenterFrequency?
```

Arguments <value>::=<NR3> depends on the input frequency band (:Setup:Band).

Table 4–42: Center frequency setting range

Input frequency band	Center frequency setting range
RF (WCA330) / RF1 (WCA380)	Span/2 Hz to 3 GHz – (Span/2) Hz
RF2 (WCA380)	2.5 GHz + (Span/2) Hz to 3.5 GHz – (Span/2) Hz
RF3 (WCA380)	3.5 GHz + (Span/2) Hz to 6.5 GHz – (Span/2) Hz
RF4 (WCA380)	5 GHz + (Span/2) Hz to 8 GHz – (Span/2) Hz

Examples `:Setup:ACP:CenterFrequency 1.5G`
sets the center frequency to 1.5 GHz.

Related Commands `:Setup:Band`

:Setup:ACP:Span (?)

Selects or queries the span.

Syntax

```
:Setup:ACP:Span { 30M | 15M }
:Setup:ACP:Span?
```

Arguments Select 30 MHz or 15 MHz for the span.

Examples `:Setup:ACP:Span 30M`
sets the span to 30 MHz.

:Setup:Band (?)

Selects or queries the input frequency band.

Syntax

```
:Setup:Band { RF | RF1 | RF2 | RF3 | RF4 }
:Setup:Band?
```

Arguments Each argument represents the input frequency band as follows:

Table 4–43: Input frequency band

Argument	Input frequency band
RF (WCA330) / RF1 (WCA380)	10 MHz to 3 GHz
RF2 (WCA380)	2.5 GHz to 3.5 GHz
RF3 (WCA380)	3.5 GHz to 6.5 GHz
RF4 (WCA380)	5 GHz to 8 GHz

Examples `:Setup:Band RF1`
selects the RF1 band.

:Setup:FFTPoints (?)

Sets or queries the number of FFT sampling points per frame.

The command usage is the same as that of the :Setup:FFTPoints command in the Standard Setup. Refer to page 4–82.

:Setup:FFTType (?)

Selects or queries the FFT type: either hardware or software processing.

The command usage is the same as that of the :Setup:FFTType command in the Standard Setup. Refer to page 4–82.

:Setup:FFTWindow (?)

Selects or queries the FFT window.

The command usage is the same as that of the :Setup:FFTWindow command in the Standard Setup. Refer to page 4–83.

:Setup:FrequencyOffset (?)

Sets or queries the frequency display offset.

The command usage is the same as that of the :Setup:FrequencyOffset command in the Standard Setup. Refer to page 4–84.

:Setup:IFMode (?)

Selects the intermediate frequency (IF) mode. For details on the IF mode, refer to the user manual.

Syntax

```
:Setup:IFMode { Normal | HiRes }
```

```
:Setup:IFMode?
```

Arguments

`Normal` has an IF bandwidth of 10 MHz and features a high degree of phase flatness. It is suitable for digital modulation analysis with a span below 6 MHz, or general measurement that does not require a wide dynamic range.

`HiRes` has a relatively narrow IF bandwidth of 6 MHz, but has the widest dynamic range. It is suitable for ACP and spurious measurement. FFT is slow because it is always performed by software.

Examples

`:Setup:IFMode Normal`
sets the IFmode to Normal.

:Setup:InputCoupling (?)

Selects or queries the input coupling to the RF INPUT connector on the front panel.

The command usage is the same as that of the :Setup:InputCoupling command in the Standard Setup. Refer to page 4–85.

:Setup:LevelOffset (?)

Sets or queries the level display offset.

The command usage is the same as that of the :Setup:LevelOffset command in the Standard Setup. Refer to page 4–85.

:Setup:Manual (?)

Determines which setting has priority, a mixer level or an RF attenuator.

The command usage is the same as that of the :Setup:Manual command in the Standard Setup. Refer to page 4–86.

:Setup:MarkerToFreq (No Query Form)

Sets the center frequency to the value at the marker.

The command usage is the same as that of the :Setup:MarkerToFreq command in the Standard Setup. Refer to page 4–87.

:Setup:MemoryMode (?)

Selects the memory mode.

The command usage is the same as that of the :Setup:MemoryMode command in the Standard Setup. Refer to page 4–88.

:Setup:MixerLevel (?)

Selects the mixer level.

The command usage is the same as that of the :Setup:MixerLevel command in the Standard Setup. Refer to page 4–88.

:Setup:ReferenceLevel (?)

Sets the reference level.

The command usage is the same as that of the :Setup:ReferenceLevel command in the Standard Setup. Refer to page 4–89.

:Setup:ReferenceOsc (?)

Selects the reference oscillator.

The command usage is the same as that of the :Setup:ReferenceOsc command in the Standard Setup. Refer to page 4–89.

:Setup:RFAtt (?)

Selects the value of the RF attenuator.

The command usage is the same as that of the :Setup:RFAtt command in the Standard Setup. Refer to page 4–90.

:Setup:Version? (Query Only)

Queries the version of the 3gppACP Setup program.

Syntax `:Setup:Version?`

Returns <NR2> indicates the version number.

Examples `:Setup:Version?`
might return `1.1`.

Waveform View Commands

When you select `Waveform` in the `Config:View<x>` command, use the commands in this section to set the details for the Waveform view.

:View<x>:Average (?)

Determines whether or not to display the result of averaging.

Syntax

```
:View<x>:Average { off | On }
```

```
:View<x>:Average?
```

Arguments

`On` displays the results of averaging.

`Off` does not display the result of averaging.

Examples

`:View1:Average On`
displays the results of averaging.

:View<x>:Average:AllFrames (No Query Form)

Specifies that averaging is performed for data in all frames acquired.

Syntax

```
:View<x>:Average:AllFrames
```

Arguments

None

Examples

`:View1:Average:AllFrames`
specifies that averaging is performed for data in all frames.

Related Commands

`:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`,
`:View<x>:Average:MarkerToFrame`

:View<x>:Average:BeginZ (?)

Specifies or queries the uppermost frame in the averaging range.

Syntax

```
:View<x>:Average:BeginZ <value>
```

```
:View<x>:Average:BeginZ?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Average:BeginZ 199`
sets the uppermost frame number to 199.

Related Commands

`:View<x>:Average:EndZ`

:View<x>:Average:EndZ (?)

Specifies or queries the lowermost frame in the averaging range.

Syntax

```
:View<x>:Average:EndZ <value>
```

```
:View<x>:Average:EndZ?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Average:EndZ 100`
sets the lowermost frame number to 100.

Related Commands

`:View<x>:Average:BeginZ`

:View<x>:Average:Execute (No Query Form)

Starts averaging.

Syntax

`:View<x>:Average:Execute`

Arguments

None

Examples

`:View1:Execute`
starts averaging.

Related Commands

`:View<x>:Average:Reset`

:View<x>:Average:MarkerToFrame (No Query Form)

Specifies that the frames between the main marker and the delta marker are averaged.

Syntax

`:View<x>:Average:MarkerToFrame`

Arguments

None

Examples

`:View1:Average:MarkerToFrame`
specifies that the frames between the main marker and the delta marker are averaged.

Related Commands

`:View<x>:Average:AllFrames`, `:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:Reset (No Query Form)

Restarts averaging.

Syntax

`:View<x>:Average:Reset`

Arguments

None

Examples

`:View1:Average:Reset`
restarts averaging.

Related Commands

`:View<x>:Average:Execute`

:View<x>:Average:Times (?)

Sets or queries the number of frames that make up an averaged waveform.

Syntax

`:View<x>:Average:Times <value>`

`:View<x>:Average:Times?`

Arguments

`<value>::=<NR1>` is the number of frames for averaging, from 1 to 1000.

Examples

`:View1:Average:Times 1000`
specifies that an averaged waveform will show the result of combining 1000 frames.

Related Commands

`:View<x>:Average:Type`

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode. For more information about the averaging, refer to the user manual.

Syntax

`:View<x>:Average:Type { RMSExpo | RMS | PeakHold }`

`:View<x>:Average:Type?`

Arguments

`RMSExpo` averages with the RMS (root mean squared) exponential. This mode weights older acquisition data so that they have a progressively smaller effect on the average.

`RMS` averages with the RMS (root mean squared).

`PeakHold` holds the peak value for each data point.

Examples

`:View1:Average:Type RMSExpo`
averages waveform with the exponential RMS.

Related Commands

`:View<x>:Average:Time`

:View<x>:Compression (?)

Selects or queries the display data compression method. It specifies how to take or discard each acquired data point for a corresponding pixel on the screen because the number of horizontal pixels is usually less than that of data points.

Syntax

`:View<x>:Compression { Sample | MinMax | Max | Min }`

`:View<x>:Compression?`

Arguments

`Sample` takes the acquired data points at regular intervals to obtain a waveform display.

`MinMax` takes the minimum and the maximum data points for a corresponding pixel. The minimum and the maximum data points are displayed with a vertical bar on the screen.

`Max` takes the maximum data point for a corresponding pixel.

`Min` takes the minimum data point for a corresponding pixel.

Examples `:View1:Compression Sample`
takes the acquired data points at regular intervals to obtain a waveform display.

:View<x>:CopyFrom (No Query Form)

Loads the display data from the text file.

Syntax `:View<x>:CopyFrom <file_name>`

Arguments `<file_name>::=<string>` is “*.TXT” (ASCII text file).
The file is the one to which the display data has been stored with the `:View<x>:CopyTo` command.

Examples `:View1:CopyFrom "SAMPLE1.TXT"`
loads the display data from the file SAMPLE1.TXT.

Related Commands `:View<x>:CopyTo`

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
D8 | <file_name> }
```

Arguments `Clipboard` copies the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D8` are the data register 1 to 8, respectively.

`<file_name>::=<string>` is “*.TXT” (ASCII text file). The extension is set to “.TXT” automatically.

Examples `:View1:CopyTo Clipboard`
stores the display data to the clipboard.

Related Commands `:View<x>:CopyFrom`

:View<x>:Edit (?)

Determines whether to display the trigger mask on the analyzer screen. Displaying the mask enables you to edit it.

Syntax

```
:View<x>:Edit { On | Off }
```

```
:View<x>:Edit?
```

Arguments

`On` displays the trigger mask and enables you to edit it.

`Off` does not display the trigger mask and disables you to edit it.

Examples

`:View1:Edit On`
displays the trigger mask and enables you to edit it.

:View<x>:Edit:DrawHorizontal (No Query Form)

Fills the trigger mask area below the horizontal line on which the marker exists.

Syntax

```
:View<x>:Edit:DrawHorizontal
```

Arguments

None

Examples

`:View1:Edit:DrawHorizontal`
fills the trigger mask area below the horizontal line on which the marker exists.

Related Commands

`:View<x>:Edit:Y`

:View<x>:Edit:DrawLine (No Query Form)

Fills the trigger mask area below the line connecting the main marker and the delta marker.

Syntax

`:View<x>:Edit:DrawLine`

Arguments

None

Examples

`:View1:Edit:DrawLine`
fills the trigger mask area below the line connecting the main marker and the delta marker.

Related Commands

`:View<x>:Edit:Y`, `:View<x>:Marker:ResetDelta`

:View<x>:Edit:DrawMax (No Query Form)

Fills the trigger mask area below the maximum line i.e. the level 40 dB higher than the reference level.

Syntax

`:View<x>:Edit:DrawMax`

Arguments

None

Examples

`:View1:Edit:DrawMax`
fills the trigger mask area below the maximum line.

Related Commands

`:View<x>:Edit:DrawMin`

:View<x>:Edit:DrawMin (No Query Form)

Fills the trigger mask area below the minimum line i.e. the level 70 dB lower than the reference level.

Syntax `:View<x>:Edit:DrawMin`

Arguments None

Examples `:View1:Edit:DrawMin`
fills the trigger mask area below the minimum line.

Related Commands `:View<x>:Edit:DrawMax`

:View<x>:Edit:Y (?)

Sets or queries the marker vertical position when editing the trigger mask.

Syntax `:View<x>:Edit:Y <value>`

`:View<x>:Edit:Y?`

Arguments `<value>::=<NR3>` ranges from the minimum (low) edge to the maximum (high) edge of the the vertical axis.

Examples `:View1:Edit:Y -50`
places the marker on –50 of the vertical position.

Related Commands `:View<x>:Edit:DrawHorizontal`, `:View<x>:Edit:DrawLine`

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { FreqAmpl | FreqPhase | FreqI | FreqQ | TimeAmpl
| TimePhase | TimeI | TimeQ }
```

```
:View<x>:Format?
```

Arguments

Defines the parameters associated with the horizontal and vertical axes as follows:

Argument	Horizontal axis	Vertical axis
FreqAmpl	Frequency (span)	Amplitude
FreqPhase	Frequency (span)	Phase
FreqI	Frequency (span)	I (In–Phase)
FreqQ	Frequency (span)	Q (Quadrature-Phase)
TimeAmpl	Time	Amplitude
TimePhase	Time	Phase
TimeI	Time	I (In–Phase)
TimeQ	Time	Q (Quadrature-phase)

Examples

```
:View1:Format FreqAmpl
```
shows the waveform with frequency along the horizontal axis and amplitude along the vertical axis.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

Syntax `:View<x>:Line:DeltaX<n>` where n = 1 to 8

Returns `<NR3>`

Examples `:View1:Line:DeltaX2?`
might return `10M`, indicating that the differnce between the vertical line marker 2 and the vertical line marker 1 is 10 MHz when the horizontal axis represents frequency.

Related Commands `:View<x>:Line:DeltaY?`, `:View<x>:Line:X<n>`

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

Syntax `:View<x>:Line:DeltaY<n>` where n = 1 to 8

Returns `<NR3>`

Examples `:View1:Line:DeltaY2?`
might return 10, indicating that the differnce between the horizontal line marker 2 and the horizontal line marker 1 is 10 dB when the vertical axis represents amplitude.

Related Commands `:View<x>:Line:DeltaX?`, `:View<x>:Line:Y<n>`

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

Syntax

`:View<x>:Line:X<n> <value>` where n = 1 to 8

`:View<x>:Line:X<n>?`

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the horizontal axis.

Examples

`:View1:Line:X1 1G`
positions the vertical line marker 1 at 1 GHz.

Related Commands

`:View<x>:Line:DeltaX?`, `:View<x>:Line:X<n>:Visible`

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

Syntax

`:View<x>:Line:X<n>:Visible { Off | On }` where n = 1 to 8

`:View<x>:Line:X<n>:Visible?`

Arguments

`Off` does not display the vertical line marker n.

`On` displays the vertical line marker n.

Examples

`:View1:Line:X1:Visible On`
displays the vertical line marker 1.

Related Commands

`:View<x>:Line:X<n>`

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

Syntax

`:View<x>:Line:Y<n> <value>` where n = 1 to 8

`:View<x>:Line:Y<n>?`

Arguments

`<value>::=<NR3>` ranges from the minimum (bottom) edge to the maximum (top) edge of the vertical axis.

Examples

`:View1:Line:Y1 -10`
positions the horizontal line marker 1 at –10 dBm.

Related Commands

`:View<x>:Line:DeltaY?`, `:View<x>:Line:Y<n>:Visible`

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

Syntax

`:View<x>:Line:Y<n>:Visible { Off | On }` where n = 1 to 8

`:View<x>:Line:Y<n>:Visible?`

Arguments

`Off` does not display the horizontal line marker n.

`On` displays the horizontal line marker n.

Examples

`:View1:Line:X1:Visible On`
displays the horizontal line marker 1.

Related Commands

`:View<x>:Line:Y<n>`

:View<x>:Marker:Band (?)

Determines whether the band marker (the two vertical bar cursors) is displayed.

Syntax

`:View<x>:Marker:Band { Off | On }`

`:View<x>:Marker:Band?`

Arguments

`Off` does not display the band power marker.

`On` displays the band power marker.

Examples

`:View1:Marker:Band On`
displays the band marker.

Related Commands

`:View<x>:Marker:Band:Center`, `:View<x>:Marker:Band:Left`, `:View<x>:Marker:Band:Right`, `:View<x>:Marker:Band:Width`

:View<x>:Marker:Band:Center (?)

Sets or queries the center frequency of the band marker (the two vertical bar cursors).

Syntax

`:View<x>:Marker:Band:Center <value>`

`:View<x>:Marker:Band:Center?`

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:Band:Center 1G`
sets the center frequency of the band marker to 1 GHz.

Related Commands

`:View<x>:Marker:Band:Width`

:View<x>:Marker:Band:Left (?)

Sets or queries the frequency of the left edge of the band marker (the two vertical bar cursors).

Syntax

```
:View<x>:Marker:Band:Left <value>

:View<x>:Marker:Band:Left?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:Band:Left 900M`
sets the frequency of the left edge of the band marker to 900 MHz.

Related Commands

`:View<x>:Marker:Band:Right`

:View<x>:Marker:Band:Right (?)

Sets or queries the frequency of the right edge of the band marker (the two vertical bar cursors).

Syntax

```
:View<x>:Marker:Band:Right <value>

:View<x>:Marker:Band:Right?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:Band:Right 1.1G`
sets the frequency of the right edge of the band marker to 1.1 GHz.

Related Commands

`:View<x>:Marker:Band:Center`, `:View<x>:Marker:Band:Left`

:View<x>:Marker:Band:Width (?)

Sets or queries the bandwidth of the band marker (the two vertical bar cursors).

Syntax

`:View<x>:Marker:Band:Width <value>`

`:View<x>:Marker:Band:Width?`

Arguments

`<value>::=<NR3>` ranges 0 to full-scale of the horizontal axis.

Examples

`:View1:Marker:Band:Width 50M`
sets the bandwidth of the band marker to 50 MHz.

Related Commands

`:View<x>:Marker:Band:Center`

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

`:View<x>:Marker:DeltaMarker { On | Off }`

`:View<x>:Marker:DeltaMarker?`

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`,
`:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
```

```
:View<x>:Marker:DeltaX?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

```
:View1:Marker:DeltaX 1G
```

positions the delta marker at 1 GHz.

Related Commands

`:View<x>:Marker:DeltaY?`

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaY?
```

Returns

`<NR3>`

Examples

```
:View1:Marker:DeltaY?
```

might return –100, indicating that the delta marker is at –100 dB.

Related Commands

`:View<x>:Marker:DeltaX`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction. Note that the adjacent peak is farther than the distance specified with the `:View<x>:Marker:SearchSeparation` command.

Syntax `:View<x>:Marker:Peak { 1 | -1 }`

Arguments 1 moves the marker to the adjacent peak on the left of the marker.

-1 moves the marker to the adjacent peak on the right of the marker.

Examples `:View1:Marker:Peak -1`
moves the marker to the adjacent peak on the right of the marker.

Related Commands `:View<x>:Marker:SearchMax`, `:View<x>:Marker:SearchMin`, `:View<x>:Marker:SearchSeparation`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

Syntax `:View<x>:Marker:ResetDelta`

Arguments None

Examples `:View1:Marker:ResetDelta`
moves the delta marker to the main marker position.

Related Commands `:View<x>:Marker:DeltaMarker`

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

Syntax
`:View<x>:Marker:SearchMax`

Arguments
None

Examples
`:View1:Marker:SearchMax`
positions the marker on the highest signal on screen.

Related Commands
`:View<x>:Marker:SearchMin`

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

Syntax
`:View<x>:Marker:SearchMin`

Arguments
None

Examples
`:View1:Marker:SearchMin`
positions the marker on the lowest signal on screen.

Related Commands
`:View<x>:Marker:SearchMax`

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

Syntax
`:View<x>:Marker:SearchSeparation <value>`

Arguments
`<value>::=<NR2>` specifies the minimum horizontal distance to separate two peaks. The range is 1 to 10 % relative to full-scale.

Examples

`:View1:Marker:SearchSeparation 10`
sets the minimum horizontal distance for peak separation to 10 % relative to full-scale.

Related Commands

`:View<x>:Marker:Peak`

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

Syntax

`:View<x>:Marker:ToggleDelta`

Arguments

None

Examples

`:View1:Marker:ToggleDelta`
changes the primary marker and delta marker positions each other.

:View<x>:Marker:Trace (?)

Selects or queries the trace on which the marker is placed.

Syntax

`:View<x>:Marker:Trace { Trace1 | Trace2 }`

`:View<x>:Marker:Trace?`

Arguments

`Trace1` places the marker on the waveform specified with the `:View<x>:Source` command.

`Trace2` places the marker on the waveform specified with the `:View<x>:Trace2:Source` command.

Examples

`:View1:Marker:Trace Trace1`
places the marker on the Trace 1.

Related Commands

`:View<x>:Source`, `:View<x>:Trace2:Source`

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:X 1.5G`
positions the marker at 1.5 GHz.

Related Commands

`:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax

```
:View<x>:Marker:Y?
```

Returns

`<NR3>`

Examples

`:View1:Marker:Y?`
might return `-30`, indicating that the marker is at –30 dBm.

Related Commands

`:View<x>:Marker:X`

:View<x>:MaskVisible (?)

Determines whether or not to display the trigger mask on the analyzer screen.

Syntax

```
:View<x>:MaskVisible { On | Off }
```

```
:View<x>:MaskVisible?
```

Arguments

`On` displays the trigger mask.

`Off` hides the trigger mask.

Examples

`:View1:MaskVisible On`
displays the trigger mask.

:View<x>:Measurement (?)

Selects or queries the measurement item. The measurement starts with the data acquisition. Query the results with the `:View<x>:Result<y>?` command.

Syntax

```
:View<x>:Measurement { Off | Noise | Power | C/N | C/No | ACP |
OBW }
```

```
:View<x>:Measurement?
```

Arguments

`Off` turns the measurement off.

`Noise` selects the noise measurement.

`Power` selects the power measurement.

`C/N` selects the carrier-to-noise ratio (C/N) measurement.

`C/No` selects the carrier-to-noise density ratio (C/No) measurement.

`ACP` selects the adjacent channel leakage power (ACP) measurement.

`OBW` selects the occupied bandwidth (OBW) measurement.

Examples

`:View1:Measurement Noise`
selects the noise measurement.

Related Commands

`:View<x>:Measurement:ACP:BW`, `:View<x>:Measurement:ACP:SP`, `:View<x>:Measurement:OBW`, `:View<x>:Result<y>?`

:View<x>:Measurement:ACP:BW (?)

Sets or queries the bandwidth for the ACP measurement.

Syntax

`:View<x>:Measurement:ACP:BW <value>`

`:View<x>:Measurement:ACP:BW?`

Arguments

`<value>::=<NR3>` ranges 0 to full-scale.

Examples

`:View1:Measurement:ACP:BW 1M`
sets the bandwidth for the ACP measurement to 1 MHz.

Related Commands

`:View<x>:Measurement`

:View<x>:Measurement:ACP:Lock (?)

Determines whether to fix the band power marker position or to makes the band power marker move with the primary (□) marker.

Syntax

`:View<x>:Measurement:ACP:Lock { On | Off }`

`:View<x>:Measurement:ACP:Lock?`

Arguments

`On` fixes the band power marker position.

`Off` makes the band power marker move with the primary (□) marker.

Examples

`:View1:Measurement:ACP:Lock On`
fixes the band power marker position.

Related Commands

`:View<x>:Measurement:ACP:Marker`

:View<x>:Measurement:ACP:Marker (?)

Selects or queries the band-marker position for the ACP measurement.

Syntax

```
:View<x>:Measurement:ACP:Marker { Upper | Center | Lower }
```

```
:View<x>:Measurement:ACP:Marker?
```

Arguments

`Upper` places the band marker to the next higher channel.

`Center` places the band marker to the current channel.

`Lower` places the band marker to the next lower channel.

Examples

`:View1:Measurement:ACP:Marker Upper`
places the band marker to the next higher channel.

Related Commands

`:View<x>:Measurement`

:View<x>:Measurement:ACP:SP (?)

Sets or queries the frequency interval between the adjacent channels for the ACP measurement.

Syntax

```
:View<x>:Measurement:ACP:SP <value>
```

```
:View<x>:Measurement:ACP:SP?
```

Arguments

`<value>::=<NR3>` ranges 0 to full-scale.

Examples

`:View1:Measurement:ACP:SP 2M`
sets the frequency interval between the adjacent channels to 2 MHz for the ACP measurement.

Related Commands

`:View<x>:Measurement`

:View<x>:Measurement:OBW (?)

Sets or queries the occupied bandwidth.

Syntax

`:View<x>:Measurement:OBW <value>`

`:View<x>:Measurement:OBW?`

Arguments

`<value>::=<NR2>` ranges 90 to 99.8 %.

Examples

`:View1:Measurement:OBW 99.8`
sets the occupied bandwidth to 99.8 %.

Related Commands

`:View<x>:Measurement`

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

Syntax

`:View<x>:Position <value>`

`:View<x>:Position?`

Arguments

`<value>::=<NR2>` ranges 0 to 100 %. It specifies the display frame position relative to the block size. 0 % specifies that the first frame in a block is displayed. 100 % represents the last frame.

Examples

`:View1:Position 100`
displays the last frame in a block.

:View<x>:RBW (?)

Sets or queries the resolution bandwidth of the RBW filter selected with the `:View<x>:RBW:Calculation` command.

Syntax

`:View<x>:RBW <value>`

`:View<x>:RBW?`

Arguments

`<value>::=<NR3>` sets the resolution bandwidth.
The range is 4.5 bins to 3 MHz.

Examples

`:View1:RBW 30k`
sets the resolution bandwidth to 30 kHz.

Related Commands

`:View<x>:RBW:Alpha`, `:View<x>:RBW:Calculation`

:View<x>:RBW:Alpha (?)

Sets or queries the shape factor of the RBW filter selected with the :View<x>:RBW:Calculation command.

Syntax

`:View<x>:RBW:Alpha <value>`

`:View<x>:RBW:Alpha?`

Arguments

`<value>::=<NR3>` sets the shape factor. The range is 0.0001 to 1.

Examples

`:View1:RBW:Alpha 1`
sets the shape factor of the RBW filter to 1.

Related Commands

`:View<x>:RBW`, `:View<x>:RBW:Calculation`

:View<x>:RBW:Calculation (?)

Simulates Resolution Bandwidth (RBW) for compatibility with data measured by a conventional scanning RF spectrum analyzer. This command also selects the filter.

Syntax

`:View<x>:RBW:Calculation { Off | Rect | Gaussian | RootNyquist }`

`:View<x>:RBW:Calculation?`

Arguments

`Off` does not simulate RBW.

`Rect` selects the rectangular filter.

`Gaussian` selects the Gaussian filter.

`RootNyquist` selects the Root Nyquist filter.

Examples

`:View1:RBW:Calculation Rect`
selects the rectangular filter.

Related Commands

`:View<x>:RBW`, `:View<x>:RBW:Alpha`

:View<x>:Result<y>? (Query Only)

Queries the measurement results. Selects the measurement with the `:View<x>:Measurement` command.

Syntax

`:View<x>:Result<y>?`

Returns

`<NR3>`

`Result1` returns the result of the measurement specified with the `:View<x>:Measurement` command. For the ACP measurement, it returns the lower measurement result.

`Result2` returns the ACP upper measurement result.

Examples

`:View1:Measurement OBW;Result1?`
might return `57.36E+03`, indicating that the occupied bandwidth is 57.36 kHz.

Related Commands

`:View<x>:Measurement`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

Syntax `:View<x>:Scale:AutoYScale`

Arguments None

Examples `:View1:Scale:AutoYScale`
adjusts the scaling of the vertical axis automatically.

Related Commands `:View<x>:Format`, `:View<x>:Scale:YScale`

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

Syntax `:View<x>:Scale:FullYScale`

Arguments None

Examples `:View1:Scale:FullYScale`
sets the vertical scale to the default full-scale.

Related Commands `:View<x>:Scale:YScale`

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

Syntax

```
:View<x>:Scale:HoldYScale { On | Off }
```

```
:View<x>:Scale:HoldYScale?
```

Arguments

`On` holds the vertical scale setting when you change the input source.

`Off` resets the vertical scale to full-scale.

Examples

`:View1:Scale:HoldYScale On`
holds the vertical scale setting when you change the input source.

Related Commands

`:View<x>:Scale:YScale`, `:View<x>:Scale:YStart`, `:View<x>:Source`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:XScale <value>
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples

`:View1:Scale:XScale 2.5M`
sets the horizontal axis full-scale to 2.5 MHz.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

Syntax

```
:View<x>:Scale:XStart <value>
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples

`:View1:Scale:XStart 800M`
sets the value represented by the left edge of the horizontal axis to 800 MHz.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:XStartZero (?)

Determines whether to put the time at the left edge of each frame to zero.

Syntax

```
:View<x>:Scale:XStartZero { On | Off }
```

```
:View<x>:Scale:XStartZero?
```

Arguments

`On` puts the time at the left edge of each frame to zero.

`Off` uses the normal time axis.

The difference between On and Off is illustrated in Figure 4–4 below.

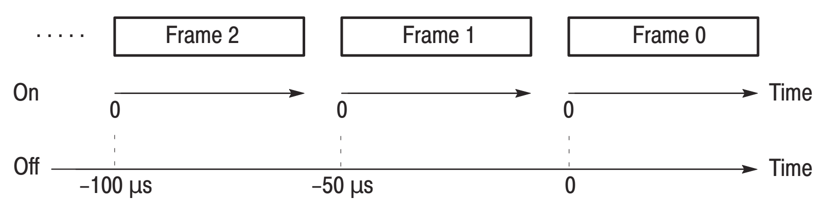


Figure 4–4: :View<x>:Scale:XStartZero On and Off (example)

Examples

`:View1:Scale:XStartZero On`
puts the time at the left edge of each frame to zero.

Related Commands

`:View<x>:Scale:XStart`

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax

`:View<x>:Scale:YScale <value>`

`:View<x>:Scale:YScale?`

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 20`
sets the vertical axis full-scale to 20 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax

`:View<x>:Scale:YStart <value>`

`:View<x>:Scale:YStart?`

Arguments

`<value>::=<NR3>` ranges from the reference level – 200 dB to the reference level + 100 dB.

Examples

`:View1:Scale:YStart -120`
sets the value represented by the bottom edge of the vertical axis to –120 dB.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Average | Zoom | D1D2 |
D3D4 | D5D6 | D7D8 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
<file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Average` specifies the averaged data as the source. When you select this item, you have to set the average type and the number of averages using the `:View<x>:Average:Type` and the `:View<x>:Average:Times` commands, respectively.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

`<file_name>::=<string>` specifies the file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the view source.

Related Commands

`:Config:Mode`, `:View<x>:Average:Times`, `:View<x>:Average:Type`

:View<x>:Trace2:Compression (?)

Selects or queries the Trace 2 display data compression method i.e. how to take or discard each acquired data point for a corresponding pixel on the screen because the number of horizontal pixels is usually less than that of data points.

Syntax

```
:View<x>:Trace2:Compression { Sample | MinMax | Max | Min }
```

```
:View<x>:Trace2:Compression?
```

Arguments

`Sample` takes the acquired data points at regular intervals to obtain a waveform display.

`MinMax` takes the minimum and the maximum data points for a corresponding pixel. The minimum and the maximum data points are displayed with a vertical bar on the screen.

`Max` takes the maximum data point for a corresponding pixel.

`Min` takes the minimum data point for a corresponding pixel.

Examples

`:View1:Trace2:Compression Sample`
takes the acquired data points at regular intervals to obtain a waveform display.

Related Commands

`:View<x>:Compression`, `:View<x>:Trace2:Source?`

:View<x>:Trace2:Format (?)

Selects or queries the display format for the Trace 2.

Syntax

```
:View<x>:Trace2:Format { FreqAmpl | FreqPhase | FreqI | FreqQ |
TimeAmpl | TimePhase | TimeI | TimeQ }
```

```
:View<x>:Trace2:Format?
```

Arguments

Defines the parameters for the horizontal and vertical axes as follows:

Argument	Horizontal axis	Vertical axis
FreqAmpl	Frequency (span)	Amplitude
FreqPhase	Frequency (span)	Phase
FreqI	Frequency (span)	I (In–Phase)
FreqQ	Frequency (span)	Q (Quadrature-Phase)
TimeAmpl	Time	Amplitude
TimePhase	Time	Phase
TimeI	Time	I (In–Phase)
TimeQ	Time	Q (Quadrature-phase)

Examples

```
:View1:Trace2:Format FreqAmpl
```

shows the Trace 2 waveform with frequency along the horizontal axis and amplitude along the vertical axis.

Related Commands

```
:View<x>:Trace2:Source
```

:View<x>:Trace2:Source (?)

Specifies or queries the display data source for the Trace 2.

Syntax

```
:View<x>:Trace2:Source { None | Active | Zoom | D1D2 |
D3D4 | D5D6 | D7D8 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
<file_name> }
```

```
:View<x>:Trace2:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:View1:Trace2:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Config:Mode`, `:View<x>:Source`

:View<x>:Trace2:Z (?)

Specifies or queries the frame number to be displayed for the Trace 2.

Syntax

```
:View<x>:Trace2:Z <value>
```

```
:View<x>:Trace2:Z?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples

`:View1:Trace2:Z 199`
specifies that the frame 199 displays for the Trace 2.

Related Commands

`:View<x>:Z`

:View<x>:Version? (Query Only)

Queries the version of the Waveform View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

Syntax

`:View<x>:Z <value>`

`:View<x>:Z?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Z 199`
specifies that the frame 199 displays.

Related Commands

`:View<x>:Trace2:Z`

Analog View Commands

When you select `Analog` in the `Config:View<x>` command, use the commands in this section to set the details for the Analog view.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
D8 | <file_name> }
```

Arguments

`Clipboard` stores the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D8` select the data register 1 to 8, respectively.

`<file_name>::=<string>` is "*.TXT" (ASCII text file). The extension is set to ".TXT" automatically.

Examples

`:View1:CopyTo Clipboard`
stores the display data to the clipboard.

:View<x>:Format (?)

Selects or queries the signal modulation type.

Syntax

```
:View<x>:Format { AM | PM | FM }
```

```
:View<x>:Format?
```

Arguments

Specifies the modulation type:

Argument	Modulation type	Horizontal axis	Vertical axis
AM	Amplitude modulation	Time	Modulating factor
PM	Phase modulation	Time	Phase
FM	Frequency modulation	Time	Frequency

Examples

`:View1:Format AM`
selects the AM modulation.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX?`, `:View<x>:Marker:DeltaY?`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
```

```
:View<x>:Marker:DeltaX?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:DeltaX 3.4m`
positions the delta marker at 3.4 ms.

Related Commands

`:View<x>:Marker:DeltaY?`

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

Syntax `:View<x>:Marker:DeltaY?`

Returns `<NR3>`

Examples `:View1:Marker:DeltaY?`
might return –100, indicating that the delta marker is positioned at –100 dB.

Related Commands `:View<x>:Marker:DeltaX?`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:X -50u`
positions the marker at –50 μs.

Related Commands

`:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax

```
:View<x>:Marker:Y?
```

Returns

```
<NR3>
```

Examples

```
:View1:Marker:Y?
```

might return 30, indicating that the marker is at 30 % when the modulation type is AM.

Related Commands

```
:View<x>:Marker:X
```

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:XScale <value>
:View<x>:Scale:XScale?
```

Arguments `<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples `:View1:Scale:XScale 20u`
sets the horizontal axis full-scale to 20 μs.

Related Commands `:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

Syntax

```
:View<x>:Scale:XStart <value>
:View<x>:Scale:XStart?
```

Arguments `<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples `:View1:Scale:XStart -50u`
sets the value represented by the left edge of the horizontal axis to –50 μs.

Related Commands `:View<x>:Scale:XScale`

:View<x>:Scale:XStartZero (?)

Determines whether to put the time at the left edge of each frame to zero.

The command usage is the same as that of the :View<x>:Scale:XStartZero command in the Waveform view. Refer to page 4–150.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 20`
sets the vertical axis full-scale to 20 % when the modulation type is AM.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax

```
:View<x>:Scale:YStart <value>
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the reference level – 200 dB to the reference level + 100 dB.

Examples

`:View1:Scale:YStart -10`
sets the value represented by the bottom edge of the vertical axis to –10 % when the modulation type is AM.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Zoom | D1D2 | D3D4 | D5D6 |
D7D8 | <file_name> }
```

```
:View<x>:Source?
```

Arguments

None specifies no source. The display area in the view will be emptied.

Active specifies the currently acquired data as the source.

Zoom specifies the zoomed data as the source.

D1D2 to D7D8 specify the register pair D1D2 to D7D8 as the source, respectively.

<file_name>::=<string> specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

```
:View1:Source Active
```

specifies the currently acquired data as the view source.

Related Commands

:Config:Mode

:View<x>:Version? (Query Only)

Queries the version of the Analog View program.

Syntax

```
:View<x>:Version?
```

Returns

<NR2>

Examples

```
:View1:Version?
```

might return 1.1.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

Syntax

```
:View<x>:Z <value>
```

```
:View<x>:Z?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Z 199`
specifies that the frame 199 displays.

FSK View Commands

When you select FSK in the Config:View<x> command, use the commands in this section to set the details for the FSK view.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
D8 | <file_name> }
```

Arguments

`Clipboard` stores the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D8` selects the data register 1 to 8, respectively.

`<file_name>::=<string>` is "*.TXT" (ASCII text file). The extension is set to ".TXT" automatically.

Examples

`:View1:CopyTo Clipboard`
stores the display data to the clipboard.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`,
`:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
:View<x>:Marker:DeltaX?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal (time) axis.

Examples

`:View1:Marker:DeltaX 1.2m`
positions the delta marker at 1.2 ms.

Related Commands

`:View<x>:Marker:DeltaY?`

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaY?
```

Returns

`<NR3>`

Examples

`:View1:Marker:DeltaY?`
might return `850.5M`, indicating that the delta marker is positioned at 850.5 MHz.

Related Commands

`:View<x>:Marker:DeltaX`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal (time) axis.

Examples

`:View1:Marker:X 3.4`
positions the marker at 3.4 ms.

Related Commands

`:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax

```
:View<x>:Marker:Y?
```

Returns

`<NR3>`

Examples

`:View1:Marker:Y?`
might return `543.2M`, indicating that the marker is positioned at 543.2 MHz.

Related Commands

`:View<x>:Marker:X`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical (frequency) axis automatically to best display the data.

Syntax

```
:View<x>:Scale:AutoScale
```

Arguments None

Examples `:View1:Scale:AutoScale`
adjusts the scaling of the vertical axis automatically.

Related Commands `:View<x>:Scale:XScale`, `:View<x>:Scale:YScale`

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax `:View<x>:Scale:XScale <value>`

`:View<x>:Scale:XScale?`

Arguments `<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples `:View1:Scale:XScale 20u`
sets the horizontal axis full-scale to 20 μs.

Related Commands `:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal (time) axis.

Syntax

```
:View<x>:Scale:XStart <value>
```

```
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples

`:View1:Scale:XStart -50u`
sets the value represented by the left edge of the horizontal axis to –50 μs.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:XStartZero (?)

Determines whether to put the time at the left edge of each frame to zero.

The command usage is the same as that of the :View<x>:Scale:XStartZero command in the Waveform view. Refer to page 4–150.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
```

```
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 10M`
sets the vertical axis full-scale to 10 MHz.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax

```
:View<x>:Scale:YStart <value>
```

```
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the vertical axis of the acquired data.

Examples

`:View1:Scale:YStart -5M`
sets the value represented by the bottom edge of the vertical axis to –5 MHz.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Zoom | D1D2 | D3D4 | D5D6 |
D7D8 | <file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the view source.

Related Commands

`:Config:Mode`

:View<x>:Version? (Query Only)

Queries the version of the FSK View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

Syntax

`:View<x>:Z <value>`

`:View<x>:Z?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Z 199`
specifies that the frame 199 displays.

Spectrogram View Commands

When you select `Spectrogram` in the `Config:View<x>` command, use the commands in this section to control the Spectrogram view.

:View<x>:Compression (?)

Selects or queries the display data compression method. It specifies how to take or discard each acquired data point for a corresponding pixel on the screen because the number of horizontal pixels is usually less than that of data points.

Syntax

```
:View<x>:Compression { Sample | Max | Min }
```

```
:View<x>:Compression?
```

Arguments

`Sample` takes the acquired data points at regular intervals to obtain a waveform display.

`Max` takes the maximum data point for a corresponding pixel.

`Min` takes the minimum data point for a corresponding pixel.

Examples

`:View1:Compression Sample`
takes the acquired data points at regular intervals to obtain a spectrogram display.

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { FreqAmpl | FreqPhase }
```

```
:View<x>:Format?
```

Arguments

Defines the parameters associated with the horizontal axis, vertical axis, and colors as follows:

Argument	Horizontal axis	Vertical (Z) axis	Color
FreqAmpl	Frequency	Frame number	Amplitude
FreqPhase	Frequency	Frame number	Phase

Examples

`:View1:Format FreqAmpl`
shows frame-by-frame time-series spectrum along the vertical axis. In each spectrum, the horizontal axis represents frequency and the color represents amplitude.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

`:View<x>:Marker:DeltaMarker { On | Off }`

`:View<x>:Marker:DeltaMarker?`

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaT? (Query Only)

Queries the delta-marker position on the Z (frame number) axis as the time.

Syntax

`:View<x>:Marker:DeltaT?`

Returns

`<NR3>`

Examples

`:View1:Marker:DeltaT?`
might return 0.0096, indicating that the delta marker is positioned at 0.0096 s.

Related Commands

`:View<x>:Marker:DeltaMarker?`, `:View<x>:Marker:DeltaZ`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

`:View<x>:Marker:DeltaX <value>`

`:View<x>:Marker:DeltaX?`

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the horizontal (frequency) axis.

Examples

`:View1:Marker:DeltaX 1.2G`
positions the delta marker at 1.2 GHz.

Related Commands

`:View<x>:Marker:DeltaY?`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaY? (Query Only)

Queries the delta-marker position on the Y (color) axis.

Syntax

`:View<x>:Marker:DeltaY?`

Returns

`<NR3>`

Examples

`:View1:Marker:DeltaY?`
might return –12.3, indicating that the delta marker is positioned at –12.3 dBm.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaZ (?)

Sets or queries the delta-marker position on the Z (frame number) axis.

Syntax

`:View<x>:Marker:DeltaZ <value>`

`:View<x>:Marker:DeltaZ?`

Arguments

`<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples

`:View1:Marker:DeltaZ 199`
positions the delta marker at 199 of the frame number.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`,
`:View<x>:Marker:ResetDelta`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:T? (Query Only)

Queries the marker position on the Z (frame number) axis as the time.

Syntax `:View<x>:Marker:T?`

Returns `<NR3>`

Examples `:View1:Marker:T?`
might return 0.005827, indicating that the delta marker is positioned at 5.827 ms.

Related Commands `:View<x>:Marker:Z`

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax `:View<x>:Marker:X <value>`

`:View<x>:Marker:X?`

Arguments `<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal (frequency) axis.

Examples `:View1:Marker:X 1.2G`
positions the marker at 1.2 GHz.

Related Commands `:View<x>:Marker:Y?`, `:View<x>:Marker:Z`

:View<x>:Marker:Y? (Query Only)

Queries the marker position on the Y (color) axis.

Syntax

`:View<x>:Marker:Y?`

Returns

`<NR3>`

Examples

`:View1:Marker:Y?`
might return –12.3, indicating that the marker is positioned at –12.3 dBm when the Y axis represents amplitude.

Related Commands

`:View<x>:Marker:X`, `:View<x>:Marker:Z`

:View<x>:Marker:Z (?)

Sets or queries the marker position on the vertical (frame number) axis.

Syntax

`:View<x>:Marker:Z <value>`

`:View<x>:Marker:Z?`

Arguments

`<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples

`:View1:Marker:Z 199`
positions the marker at 199 of the frame number.

Related Commands

`:View<x>:Marker:X`, `:View<x>:Marker:Y?`

:View<x>:Monochrome (?)

Determines whether to display spectrogram in monochrome.

Syntax

`:View<x>:Monochrome { On | Off }`

`:View<x>:Monochrome?`

Arguments

`On` displays spectrogram in monochrome.

`Off` displays spectrogram in color.

Examples

`:View1:Monochrome On`
displays spectrogram in monochrome.

:View<x>:NumberColors (?)

Selects or queries the number of display colors.

Syntax

`:View<x>:NumberColors { 10 | 100 }`

`:View<x>:NumberColors?`

Arguments

10 selects 10-color display (same as the system software version 1.6 or before).

100 selects 100-color display (default).

Examples

`:View1:NumberColors 100`
selects 100 color display.

Related Commands

`:View<x>:Scale:YScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the Y (color) axis automatically to best display the data.

Syntax
`:View<x>:Scale:AutoYScale`

Arguments
None

Examples
`:View1:Scale:AutoYScale`
adjusts the scaling of the Y axis automatically.

Related Commands
`:View<x>:Scale:YScale`

:View<x>:Scale:FullYScale (No Query Form)

Sets the Y (color) axis scale to the default full-scale.

Syntax
`:View<x>:Scale:FullYScale`

Arguments
None

Examples
`:View1:Scale:FullYScale`
sets the Y scale to the default full-scale.

Related Commands
`:View<x>:Scale:YScale`

:View<x>:Scale:HoldYScale (?)

Determines whether to retain or reset the Y (color) axis scale when you change the input source.

Syntax

```
:View<x>:Scale:HoldYScale { On | Off }
:View<x>:Scale:HoldYScale?
```

Arguments

`On` retains the Y scale setting when you change the input source.

`Off` resets the Y scale to full-scale.

Examples

`:View1:Scale:HoldYScale On`
holds the Y scale setting when you change the input source.

Related Commands

`:View<x>:Scale:YScale`, `:View<x>:Scale:YStart`, `:View<x>:Source`

:View<x>:Scale:TimeScale (?)

Determines whether the time scale is displayed on screen.

Syntax

```
:View<x>:Scale:TimeScale { Off | On }
:View<x>:Scale:TimeScale?
```

Arguments

`Off` does not display the time scale.

`On` displays the time scale.

Examples

`:View1:Scale:TimeScale On`
displays the time scale.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal (frequency) axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:XScale <value>
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples

`:View1:Scale:XScale 2.5M`
sets the horizontal axis full-scale to 2.5 MHz.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal (frequency) axis.

Syntax

```
:View<x>:Scale:XStart <value>
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples

`:View1:Scale:XStart 543.2M`
sets the value represented by the left edge of the horizontal axis to 543.2 MHz.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the Y (color) axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
```

```
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 2`
sets the Y axis full-scale to 2 dB when the axis represents the amplitude.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Y (color) axis.

Syntax

```
:View<x>:Scale:YStart <value>
```

```
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the Y axis of the acquired data.

Examples

`:View1:Scale:YStart -12.5`
sets the value represented by the bottom edge of the Y axis to –12.5 dBm.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:ZScale (?)

Sets or queries the frame display interval. The spectrogram is displayed every specified number of frames.

Syntax

```
:View<x>:Scale:ZScale <value>
```

```
:View<x>:Scale:ZScale?
```

Arguments

`<value>::=<NR1>` ranges 1 to 32.
1 means every frame is displayed. 32 means every 32th frame is displayed.

Examples

`:View1:Scale:ZScale 8`
displays the spectrogram every 8th frame.

Related Commands

`:View<x>:Scale:ZStart`

:View<x>:Scale:ZStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Z (frame number) axis, i.e. the first frame to be displayed.

Syntax

```
:View<x>:Scale:ZStart <value>
```

```
:View<x>:Scale:ZStart?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Scale:ZStart 20`
sets the value represented by the bottom edge of the Z axis to 20.

Related Commands

`:View<x>:Scale:ZScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Zoom | D1 | D2 | D3 | D4 | D5 |
D6 | D7 | D8 | <file_name> }
```

```
:View<x>:Source?
```

Arguments

None specifies no source. The display area in the view will be emptied.

Active specifies the currently acquired data as the source.

Zoom specifies the zoomed data as the source.

D1 to D8 specify the register D1 to D8 as the source, respectively.

<file_name>::=<string> specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

:View1:Source Active
specifies the currently acquired data as the view source.

Related Commands

:Config:Mode

:View<x>:Version? (Query Only)

Queries the version of the Spectrogram View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

:View<x>:ZGap (?)

Sets or queries the display interval between two adjacent frames on the Z (frame number) axis.

Syntax `:View<x>:ZGap <value>`

`:View<x>:ZGap?`

Arguments `<value>::=<NR1>` ranges 1 to 10 pixels.

Examples `:View1:ZGap 10`
sets the display interval between two adjacent frames on the Z axis to 10 pixels.

Waterfall View Commands

When you select `Waterfall` in the `:Config:View<x>` command, use the commands in this section to control the Waterfall view.

:View<x>:Compression (?)

Selects or queries the display data compression method.

The command usage is the same as that of the :View<x>:Compression command in the Waveform view. Refer to page 4–124.

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { FreqAmpl | FreqPhase | FreqI | FreqQ | TimeAmpl
| TimePhase | TimeI | TimeQ }
```

```
:View<x>:Format?
```

Arguments

Defines the parameters associated with the horizontal and vertical axes as follows:

Argument	Horizontal axis	Vertical axis
FreqAmpl	Frequency (span)	Amplitude
FreqPhase	Frequency (span)	Phase
FreqI	Frequency (span)	I (In–Phase)
FreqQ	Frequency (span)	Q (Quadrature-Phase)
TimeAmpl	Time	Amplitude
TimePhase	Time	Phase
TimeI	Time	I (In–Phase)
TimeQ	Time	Q (Quadrature-phase)

Examples

```
:View1:Format FreqAmpl
```
shows the waveform, with the frequency along the horizontal axis, and the amplitude along the vertical axis.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaT? (Query Only)

Queries the delta marker position on the Z (frame number) axis as the time.

Syntax

```
:View<x>:Marker:DeltaT?
```

Returns

`<NR3>`

Examples

`:View1:Marker:DeltaT?`
might return 0.0096, indicating that the delta marker is positioned at 0.0096 s.

Related Commands

`:View<x>:Marker:DeltaMarker`, `:View<x>:Marker:DeltaZ`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
```

```
:View<x>:Marker:DeltaX?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:DeltaX 1.2G`
positions the delta marker at 1.2 GHz.

Related Commands

`:View<x>:Marker:DeltaY?`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaY? (Query Only)

Queries the delta-marker position on the vertical axis.

Syntax

```
:View<x>:Marker:DeltaY?
```

Returns

`<NR3>`

Examples

`:View1:Marker:DeltaY?`
might return –12.3, indicating that the delta marker is positioned at –12.3 dBm.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaZ`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaZ (?)

Sets or queries the delta-marker position on the Z (frame number) axis.

Syntax

```
:View<x>:Marker:DeltaZ <value>
```

```
:View<x>:Marker:DeltaZ?
```

Arguments `<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples `:View1:Marker:DeltaZ 199`
positions the delta marker at 199 of the frame number.

Related Commands `:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`,
`:View<x>:Marker:ResetDelta`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:T? (Query Only)

Queries the marker position on the Z (frame number) axis as the time.

Syntax

`:View<x>:Marker:T?`

Returns

`<NR3>`

Examples

`:View1:Marker:T?`
might return 0.0096, indicating that the marker is at 0.0096 s.

Related Commands

`:View<x>:Marker:Z`

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

`:View<x>:Marker:X <value>`

`:View<x>:Marker:X?`

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:X 1.2G`
positions the marker at 1.2 GHz.

Related Commands

`:View<x>:Marker:Y?`, `:View<x>:Marker:Z`

:View<x>:Marker:Y? (Query Only)

Queries the marker position on the vertical axis.

Syntax `:View<x>:Marker:Y?`

Returns `<NR3>`

Examples `:View1:Marker:Y?`
might return –12.3, indicating that the marker is positioned at –12.3 dBm.

Related Commands `:View<x>:Marker:X`, `:View<x>:Marker:Z`

:View<x>:Marker:Z (?)

Sets or queries the marker position on the Z (frame number) axis.

Syntax `:View<x>:Marker:Z <value>`

`:View<x>:Marker:Z?`

Arguments `<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples `:View1:Marker:Z 199`
positions the marker at 199 of the frame number.

Related Commands `:View<x>:Marker:X`, `:View<x>:Marker:Y?`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:XScale <value>
```

```
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples

`:View1:Scale:XScale 2.5M`
sets the horizontal axis full-scale to 2.5 MHz.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

Syntax

```
:View<x>:Scale:XStart <value>
```

```
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples
:View1:Scale:XStart 543.2M
sets the value represented by the left edge of the horizontal axis to 543.2 MHz.

Related Commands
:View<x>:Scale:XScale

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax
:View<x>:Scale:YScale <value>

:View<x>:Scale:YScale?

Arguments
<value>::=<NR3> ranges full-scale/100 to full-scale of the acquired data.

Examples
:View1:Scale:YScale 2
sets the vertical axis full-scale to 2 dB.

Related Commands
:View<x>:Scale:XScale, :View<x>:Scale:YStart

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax
:View<x>:Scale:YStart <value>

:View<x>:Scale:YStart?

Arguments
<value>::=<NR3> ranges from the minimum to the maximum value on the vertical axis of the acquired data.

Examples
:View1:Scale:YStart -12.5
sets the value represented by the bottom edge of the vertical axis to –12.5 dBm.

Related Commands
:View<x>:Scale:YScale

:View<x>:Scale:ZScale (?)

Sets or queries the frame display interval. The waveform is displayed every specified number of frames.

Syntax

```
:View<x>:Scale:ZScale <value>

:View<x>:Scale:ZScale?
```

Arguments

`<value>::=<NR1>` ranges 1 to 32.
1 means every frame is displayed. 32 means every 32th frame is displayed.

Examples

`:View1:Scale:ZScale 8`
displays the waveform every 8th frame.

Related Commands

`:View<x>:Scale:ZStart`

:View<x>:Scale:ZStart (?)

Specifies or queries the first frame to be displayed.

Syntax

```
:View<x>:Scale:ZStart <value>

:View<x>:Scale:ZStart?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples

`:View1:Scale:ZStart 20`
displays the waveform starting from the frame number 20.

Related Commands

`:View<x>:Scale:ZScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Zoom | D1 | D2 | D3 | D4 | D5 |
D6 | D7 | D8 | <file_name> }
```

```
:View<x>:Source?
```

Arguments

None specifies no source. The display area in the view will be emptied.

Active specifies the currently acquired data as the source.

Zoom specifies the zoomed data as the source.

D1 to D8 specify the data register D1 to D7 as the source, respectively.

<file_name>::=<string> specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

:View1:Source Active
specifies the currently acquired data as the view source.

Related Commands

:Config:Mode

:View<x>:Version? (Query Only)

Queries the version of the Waterfall View program.

Syntax

```
:View<x>:Version?
```

Returns

<NR2>

Examples

:View1:Version?
might return 1.1.

:View<x>:YHeight (?)

Sets or queries the full-scale of the vertical axis in pixels.

Syntax

```
:View<x>:YHeight <value>
:View<x>:YHeight?
```

Arguments

`<value>::=<NR1>` ranges 1 to 100 pixels.

Examples

`:View1:YHeight 100`
sets the full-scale of the vertical axis to 100 pixels.

Related Commands

`:View<x>:ZGap`

:View<x>:ZGap (?)

Sets or queries the frame interval for the Waterfall display.

Syntax

```
:View<x>:ZGap <value>
:View<x>:ZGap?
```

Arguments

`<value>::=<NR1>` ranges 1 to 100 pixels.

Examples

`:View1:ZGap 100`
sets the frame interval to 100 pixels.

Related Commands

`:View<x>:YHeight`

Polar View Commands

When you select `Polar` in the `:Config:View<x>` command, use the commands in this section to control the Polar view.

For information on the Polar view, refer to the user manual.

:View<x>:AlphaBT (?)

Sets or queries the α/BT value.

Syntax

`:View<x>:AlphaBT <value>`

`:View<x>:AlphaBT?`

Arguments

`<value>::=<NR2>` ranges 0.0001 to 1.

Examples

`:View1:AlphaBT 1`
sets the α/BT value to 1.

:View<x>:AutoCarrier (?)

Determines whether to search the carrier automatically for each frame.

Syntax

`:View<x>:AutoCarrier { On | Off }`

`:View<x>:AutoCarrier?`

Arguments

`On` searches the carrier automatically for each frame, and displays the frequency error in reference to the center frequency on screen at `Freq Error`.

`Off` sets the carrier frequency to the value set with the `:View<x>:Carrier` command.

Examples

`:View1:AutoCarrier On`
searches the carrier automatically and displays the frequency error.

Related Commands

`:View<x>:Carrier`

:View<x>:Burst:BlockSize (?)

Sets or queries the range for searching a burst signal in frames. The start frame is set with the `:View<x>:Z` command. If the number of frames after the start frame does not reach the specified number, the frames before the start frame are also used. See Figure 4–5.

Syntax

```
:View<x>:Burst:BlockSize <value>
```

```
:View<x>:Burst:BlockSize?
```

Arguments

`<value>::=<NR1>` ranges 1 to 20 frames.

Examples

`:View1:Burst:BlockSize 4`
sets the range for the burst search to 4 frames.

Related Commands

`:View<x>:Burst:NumberFrames`, `:View<x>:Burst:Offset`, `:View<x>:Burst:Search`, `:View<x>:Burst:Threshold`, `:View<x>:Z`

:View<x>:Burst:NumberFrames (?)

Sets or queries the range for acquiring burst data. See Figure 4–5.

Syntax

```
:View<x>:Burst:NumberFrames <value>
```

```
:View<x>:Burst:NumberFrames?
```

Arguments

`<value>::=<NR1>` is 1 or 2 frames.

Examples

`:View1:Burst:NumberFrames 2`
sets the range for burst data acquisition to 2 frames.

Related Commands

`:View<x>:Burst:BlockSize`, `:View<x>:Burst:Offset`, `:View<x>:Burst:Search`, `:View<x>:Burst:Threshold`, `:View<x>:Z`

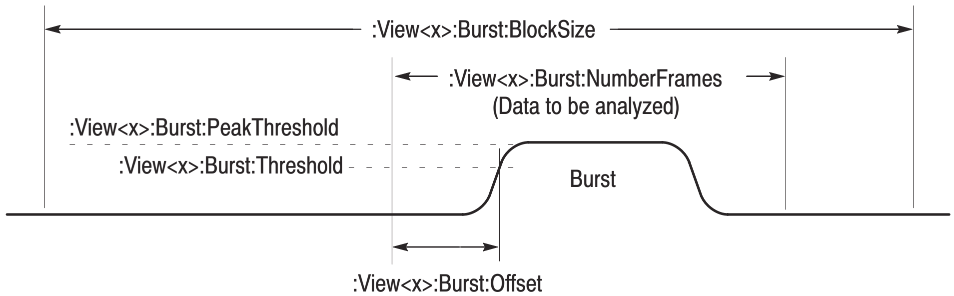


Figure 4–5: Settings for burst analysis

:View<x>:Burst:Offset (?)

Sets or queries the start point for acquiring burst data relative to the rising edge of the burst signal. See Figure 4–5.

Syntax

`:View<x>:Burst:Offset <value>`

`:View<x>:Burst:Offset?`

Arguments

`<value>::=<NR1>` ranges –1024 to +1024 points. The minus values represent the data points before the rising edge.

Examples

`:View1:Burst:Offset -10`
specifies that the analyzer acquires burst data from 10 points before the rising edge.

Related Commands

`:View<x>:Burst:BlockSize`, `:View<x>:Burst:NumberFrames`, `:View<x>:Burst:Search`, `:View<x>:Burst:Threshold`, `:View<x>:Z`

:View<x>:Burst:PeakThreshold (?)

Sets or queries the threshold to determine whether the input signal is a burst. If the peak level of the input signal is higher than the specified level, it is recognized as a burst. See Figure 4–5.

Syntax

`:View<x>:Burst:PeakThreshold <value>`

`:View<x>:Burst:PeakThreshold?`

Arguments

<value>::=<NR1> ranges –100 to +10 dB. The value is the relative level from the reference level.

Examples

:View1:Burst:PeakThreshold -40
sets the threshold to determine whether the input signal is a burst to 40 dB lower than the reference level.

Related Commands

:View<x>:Burst:BlockSize, :View<x>:Burst:NumberFrames, :View<x>:Burst:Offset, :View<x>:Burst:Search, :View<x>:Z

:View<x>:Burst:Search (?)

Determines whether to search a burst signal.

NOTE*. When you search bursts, set the frame position to 0 % with the* :View<x>:Position *command. If you use the default position (100 %), you cannot find the bursts.*

Syntax

:View<x>:Burst:Search { On | Off }

:View<x>:Burst:Search?

Arguments

On searches a burst for analyzing.

Off analyzes the frame specified with the :View<x>:Z command.

Examples

:View1:Burst:Search On
searches a burst for analyzing.

Related Commands

:View<x>:Burst:BlockSize, :View<x>:Burst:NumberFrames, :View<x>:Burst:Offset, :View<x>:Burst:Threshold, :View<x>:Position

:View<x>:Burst:Threshold (?)

Sets or queries the threshold for determining the rising edge of a burst signal. See Figure 4–5.

Syntax

```
:View<x>:Burst:Threshold <value>
```

```
:View<x>:Burst:Threshold?
```

Arguments

`<value>::=<NR1>` ranges –100 to +10 dB. The value is the relative level from the maximum within the burst search range.

Examples

`:View1:Burst:Threshold -20`
sets the threshold for determining the rising edge of a burst signal to –20 dB.

Related Commands

`:View<x>:Burst:BlockSize`, `:View<x>:Burst:NumberFrames`, `:View<x>:Burst:Offset`, `:View<x>:Burst:Search`, `:View<x>:Z`

:View<x>:Carrier (?)

Sets or queries the carrier frequency.

Syntax

```
:View<x>:Carrier <value>
```

```
:View<x>:Carrier?
```

Arguments

`<value>::=<NR3>` ranges 0 Hz to 3 GHz.

Examples

`:View1:Carrier 1.2G`
sets the carrier frequency to 1.2 GHz.

Related Commands

`:View<x>:AutoCarrier`

:View<x>:Display (?)

Selects or queries the display data source.

Syntax

```
:View<x>:Display { Measurement | Reference }
```

```
:View<x>:Display?
```

Arguments

Measurement displays the measurement data.

Reference displays the reference data.

Examples

:View1:Display Measurement
displays the measurement data.

Related Commands

:View<x>:MeasDestination, :View<x>:RefDestination

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { Vector | Constellation }
```

```
:View<x>:Format?
```

Arguments

Vector specifies the Vector format. It displays symbol-to-symbol movements using vector.

Constellation specifies the Constellation format. It displays only symbols.

Examples

:View1:Format Vector
selects the Vector display format.

:View<x>:Marker:A? (Query Only)

Queries the amplitude at the marker position.

Syntax

```
:View<x>:Marker:A?
```

Returns

<NR3>

Examples

:View1:Marker:A?
might return 0.789.

Related Commands

:View<x>:Marker:P?

:View<x>:Marker:DeltaT (?)

Sets or queries the delta-marker position on the time axis.

Syntax

```
:View<x>:Marker:DeltaT <value>
:View<x>:Marker:DeltaT?
```

Arguments `<value>::=<NR3>` ranges the minimum to the maximum value on the time axis.

Examples `:View1:Marker:DeltaT 5.4u`
positions the delta marker at 5.4 μs.

Related Commands `:View<x>:Marker:T`

:View<x>:Marker:P? (Query Only)

Queries the phase at the marker position.

Syntax `:View<x>:Marker:P?`

Returns `<NR3>`

Examples `:View1:Marker:P?`
might return `51.313`, indicating that the marker is positioned at 51.313 degrees.

Related Commands `:View<x>:Marker:A?`

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

Syntax

```
:View<x>:Marker:T <value>
:View<x>:Marker:T?
```

Arguments `<value>::=<NR3>` ranges the minimum to the maximum value on the time axis.

Examples `:View1:Marker:T 5.4u`
positions the marker at 5.4 μs.

Related Commands `:View<x>:Marker:DeltaT?`

:View<x>:Marker:X? (Query Only)

Queries the horizontal position of the marker.

Syntax `:View<x>:Marker:X?`

Returns `<NR3>`
1 represents the full-scale.

Examples `:View1:Marker:X?`
might return 0.345.

Related Commands `:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax `:View<x>:Marker:Y?`

Returns `<NR3>`
1 represents the full-scale.

Examples `:View1:Marker:Y?`
might return 0.765.

Related Commands `:View<x>:Marker:X?`

:View<x>:Mask:Frequency (?)

Determines whether the frequency-domain mask is enabled. Specify the mask area with either the View<x>:Mask:Canter and Width or the View<x>:Mask:Left and Right. See Figure 4–6.

Syntax

```
:View<x>:Mask:Frequency { Off | On }
```

```
:View<x>:Mask:Frequency?
```

Arguments

`Off` disables the frequency-domain mask.

`On` enables the frequency-domain mask.

Examples

`:View1:Mask:Frequency On`
enables the frequency-domain mask.

Related Commands

`:View<x>:Mask:Frequency:Center`, `:View<x>:Mask:Frequency:Left`, `:View<x>:Mask:Frequency:Right`, `:View<x>:Mask:Frequency:Width`

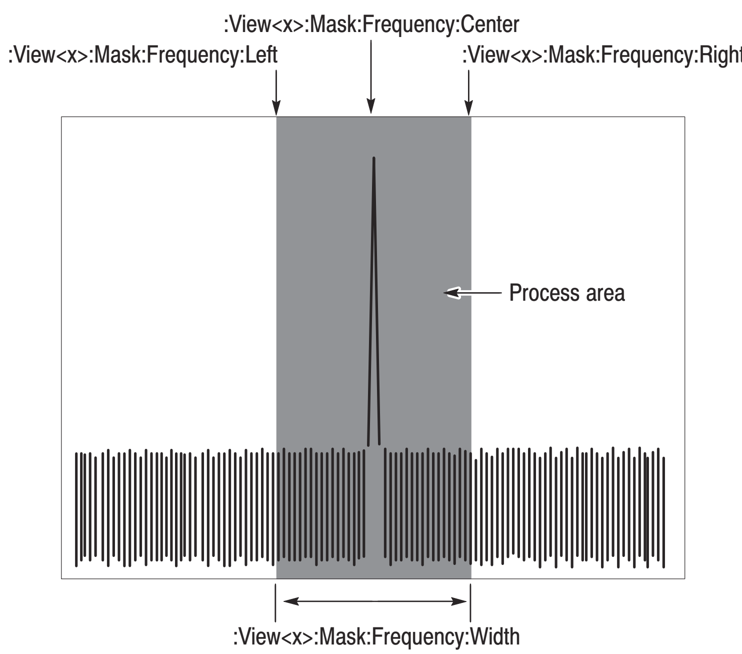


Figure 4–6: Frequency-domain mask setting

:View<x>:Mask:Frequency:Center (?)

Sets or queries the center frequency of the mask area. Use this command with the View<x>:Mask:Width. See Figure 4–6. The data out of the specified area is excluded from the analysis.

Syntax

`:View<x>:Mask:Frequency:Center <value>`

`:View<x>:Mask:Frequency:Center?`

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the frequency axis.

Examples

`:View1:Mask:Frequency:Center 100M`
sets the center frequency of the mask area to 100 MHz.

Related Commands

`:View<x>:Mask:Frequency`, `:View<x>:Mask:Frequency:Width`

:View<x>:Mask:Frequency:Left (?)

Sets or queries the left frequency of the mask area. Use this command with the View<x>:Mask:Right. See Figure 4–6. The data out of the specified area is excluded from the analysis.

Syntax

`:View<x>:Mask:Frequency:Left <value>`

`:View<x>:Mask:Frequency:Left?`

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the frequency axis.

Examples

`:View1:Mask:Frequency:Left 100M`
sets the left frequency of the mask area to 100 MHz.

Related Commands

`:View<x>:Mask:Frequency`, `:View<x>:Mask:Frequency:Right`

:View<x>:Mask:Frequency:Right (?)

Sets or queries the right frequency of the mask area. Use this command with the View<x>:Mask:Left. See Figure 4–6. The data out of the specified area is excluded from the analysis.

Syntax

```
:View<x>:Mask:Frequency:Right <value>

:View<x>:Mask:Frequency:Right?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the frequency axis.

Examples

`:View1:Mask:Frequency:Right 100M`
sets the right frequency of the mask area to 100 MHz.

Related Commands

`:View<x>:Mask:Frequency`, `:View<x>:Mask:Frequency:Left`

:View<x>:Mask:Frequency:Width (?)

Sets or queries the bandwidth of the mask area. Use this command with the View<x>:Mask:Center. See Figure 4–6. The data out of the specified area is excluded from the analysis.

Syntax

```
:View<x>:Mask:Frequency:Width <value>

:View<x>:Mask:Frequency:Width?
```

Arguments

`<value>::=<NR3>` ranges 0 to full-scale of the frequency axis.

Examples

`:View1:Mask:Frequency:Width 1M`
sets the bandwidth of the mask area to 1 MHz.

Related Commands

`:View<x>:Mask:Frequency`, `:View<x>:Mask:Frequency:Center`

:View<x>:Mask:MarkerLink (?)

Determines whether the center frequency of the mask area is set with the marker in the other view.

Syntax

```
:View<x>:Mask:MarkerLink { Off | On }
```

```
:View<x>:Mask:MarkerLink?
```

Arguments

`Off` sets the center frequency of the mask area with the :View<x>:Mask:Frequency:Center command.

`On` sets the center frequency of the mask area to the marker frequency in the other view.

Examples

`:View1:Mask:MarkerLink On`
sets the center frequency of the mask area to the marker frequency in the other view.

Related Commands

`:View<x>:Mask:Frequency`, `:View<x>:Mask:Frequency:Center`

:View<x>:MeasDestination (?)

Selects or queries the register pair to store the measurement data.

Syntax

```
:View<x>:MeasDestination { None | D1D2 | D3D4 | D5D6 | D7D8 }
:View<x>:MeasDestination?
```

Arguments

`None` specifies no destination.

`D1D2` to `D7D8` specify a register pair to store the measurement data.

Examples

`:View1:MeasDestination D1D2`
specifies the D1 and D2 register pair for storing data.

Related Commands

`:View<x>:RefDestination`

:View<x>:MeasFilter (?)

Selects or queries the filter to create the measurement data.

Syntax

```
:View<x>:MeasFilter { None | RootRaisedCosine }
:View<x>:MeasFilter?
```

Arguments

`None` selects no filter.

`RootRaisedCosine` selects the root raised-cosine filter.

Examples

`:View1:MeasFilter RootRaisedCosine`
selects the root raised-cosine filter to create the measurement data.

Related Commands

`:View<x>:RefFilter`

:View<x>:Modulation (?)

Selects or queries the modulation type.

Syntax

```
:View<x>:Modulation { PI4QPSK | BPSK | QPSK | PSK8 | QAM16 |
QAM64 | QAM256 | GMSK | GFSK }
```

```
:View<x>:Modulation?
```

Arguments

PI4QPSK selects 1/4 π Shift QPSK (Quadrature Phase Shift Keying) modulation.

BPSK selects BPSK (Binary Phase Shift Keying) modulation.

QPSK selects QPSK (Quadrature Phase Shift Keying) modulation.

PSK8 selects 8PSK (Phase Shift Keying) modulation.

QAM16 selects 16QAM (Quadrature Amplitude Modulation).

QAM64 selects 64QAM (Quadrature Amplitude Modulation).

QAM256 selects 256QAM (Quadrature Amplitude Modulation).

GMSK selects GMSK (Gaussian-filtered Minimum Shift Keying) modulation.

GFSK selects GFSK (Gaussian-filtered Frequency Shift Keying) modulation.

Examples

```
:View1:Modulation PI4QPSK
```
selects the 1/4 π Shift QPSK modulation.

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

Syntax

```
:View<x>:Position <value>
```

```
:View<x>:Position?
```

Arguments

<value>::=<NR2> ranges 0 to 100 %. It specifies the display frame position relative to the block size. 0 % specifies that the first frame in a block is displayed. 100 % represents the last frame.

Examples

```
:View1:Position 100
```
displays the last frame in a block.

:View<x>:RefDestination (?)

Selects or queries the register pair to store the reference data.

Syntax

```
:View<x>:RefDestination { None | D1D2 | D3D4 | D5D6 | D7D8 }
```

```
:View<x>:RefDestination?
```

Arguments

`None` specifies no destination.

`D1D2` to `D7D8` specify the register pair to store the reference data.

Examples

`:View1:RefDestination D1D2`
selects the D1 and D2 register pair to store the reference data.

Related Commands

`:View<x>:MeasDestination`

:View<x>:RefFilter (?)

Selects or queries the filter to create the reference data.

Syntax

```
:View<x>:RefFilter { None | RaisedCosine | Gaussian }
```

```
:View<x>:RefFilter?
```

Arguments

`None` selects no filter.

`RaisedCosine` selects the raised cosine filter.

`Gaussian` selects the Gaussian filter.

Examples

`:View1:RefFilter RaisedCosine`
selects the raised cosine filter to create the reference data.

Related Commands

`:View<x>:MeasFilter`

:View<x>:Result<y>? (Query Only)

Queries the measurement result.

Syntax

`:View<x>:Result<y>?`

Returns

`<NR3>`

`Result1` returns the carrier frequency error in reference to the center frequency.

`Result2` returns the origin offset.

`Result3` returns the full-scale in volts.

Examples

`:View1:Result1?`
might return `15.23`, indicating that the carrier frequency error is 15.23 Hz.

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

`:View<x>:Source { None | Active | Zoom | <file_name> }`

`:View<x>:Source?`

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Config:Mode`

:View<x>:Standard:CDPD (No Query Form)

Configures the modulating system according to the CDPD (Cellular Digital Packet Data) standard.

Syntax `:View<x>:Standard:CDPD`

Arguments None

Examples `:View1:Standard:CDPD`
configures the modulating system according to the CDPD standard.

:View<x>:Standard:GSM (No Query Form)

Configures the modulating system according to the GSM (Global System for Mobile Communication) standard.

Syntax `:View<x>:Standard:GSM`

Arguments None

Examples `:View1:Standard:GSM`
configures the modulating system according to the GSM standard.

:View<x>:Standard:NADC (No Query Form)

Configures the modulating system according to the NADC (North American Digital Cellular) standard.

Syntax `:View<x>:Standard:NADC`

Arguments None

Examples `:View1:Standard:NADC`
configures the modulating system according to the NADC standard.

:View<x>:Standard:PDC (No Query Form)

Configures the modulating system according to the PDC (Personal Digital Cellular System) standard.

Syntax `:View<x>:Standard:PDC`

Arguments None

Examples `:View1:Standard:PDC`
configures the modulating system according to the PDC standard.

:View<x>:Standard:PHS (No Query Form)

Configures the modulating system according to the PHS (Personal Handy Phone System) standard.

Syntax `:View<x>:Standard:PHS`

Arguments None

Examples `:View1:Standard:PHS`
configures the modulating system according to the PHS standard.

:View<x>:Standard:TETRA (No Query Form)

Configures the modulating system according to the TETRA (Trans-European Trunked Radio) standard.

Syntax `:View<x>:Standard:TETRA`

Arguments None

Examples `:View1:Standard:TETRA`
configures the modulating system according to the TETRA standard.

:View<x>:SymbolRate (?)

Sets or queries the symbol rate for the digital modulation.

Syntax

`:View<x>:SymbolRate <value>`

`:View<x>:SymbolRate?`

Arguments

`<value>::=<NR3>` ranges 1/s to 30 M/s.

Examples

`:View1:SymbolRate 8M`
sets the symbol rate to 8 M/s.

:View<x>:Version?

Queries the version of the Polar View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

Syntax

`:View<x>:Z <value>`

`:View<x>:Z?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Z 199`
specifies that the frame 199 displays.

Eye Diagram View Commands

When you select `EyeDiagram` in the `:Config:View<x>` command, use the commands in this section to control the Eye Diagram view.

For information on the eye diagram, refer to the user manual.

:View<x>:EyeLength (?)

Sets or queries the number of symbols to be displayed, i.e. the horizontal scale.

Syntax

`:View<x>:EyeLength <value>`

`:View<x>:EyeLength?`

Arguments

`<value>::=<NR1>` ranges 1 to 16.

The default length is set to two symbols.

Examples

`:View1:EyeLength 4`
displays four symbols in the eye diagram.

Related Commands

`:View<x>:Format`

:View<x>:Format (?)

Selects or queries the parameter of the vertical axis. To specify the horizontal axis, use the `:View<x>:EyeLength` command.

Syntax

`:View<x>:Format { I | Q | Trellis }`

`:View<x>:Format?`

Arguments

`I` displays the eye diagram with the I data along the vertical axis.

`Q` displays the eye diagram with the Q data along the vertical axis.

`Trellis` displays the eye diagram with the phase along the vertical axis.

Examples

`:View1:Format I`
displays the eye diagram with the I data along the vertical axis.

Related Commands

`:View<x>:EyeLength`

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

Syntax

`:View<x>:Marker:T <value>`

`:View<x>:Marker:T?`

Arguments

`<value>::=<NR3>` ranges the minimum to the maximum value on the time axis.

Examples

`:View1:Marker:T 5.5u`
positions the marker at 5.5 μs.

Related Commands

`:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax

`:View<x>:Marker:Y?`

Returns

`<NR3>`

Examples

`:View1:Marker:Y?`
might return `0.000005`, indicating that the marker is positioned at 5 μs.

Related Commands

`:View<x>:Marker:T`

:View<x>:Source (?)

Selects or queries the data source for the eye diagram.

Syntax

`:View<x>:Source { Measurement | Reference }`

`:View<x>:Source?`

Arguments

`Measurement` selects the measurement data as the view source. The data is stored in the register pair specified with the `:View<x>:MeasDestination` in the Polar View or the CDMA Polar View command group.

`Reference` selects the reference data as the data source. The data is stored in the register pair specified with the `:View<x>:RefDestination` in the Polar View or the CDMA Polar View command group.

Examples

`:View1:Source Measurement`
selects the measurement data as the view source.

Related Commands

`:View<x>:MeasDestination` and `:View<x>:RefDestination`
in the Polar View or the CDMA Polar View command group

:View<x>:Version?

Queries the version of the Eye Diagram View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

Symbol Table View Commands

When you select SymbolTable in the :Config:View<x> command, use the commands in this section to control the Symbol Table view.

For information on the symbol table, refer to the user manual.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | <file_name> }
```

Arguments

`Clipboard` copies the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D4` are the data register 1 to 4, respectively.

`<file_name>::=<string>` is "*.TXT" (ASCII text file). The extension is set to ".TXT" automatically.

Examples

`:View1:CopyTo Clipboard`
stores the display data to the clipboard.

:View<x>:Marker:Data? (Query Only)

Queries the value on the marker position.

Syntax

```
:View<x>:Marker:Data?
```

Returns

`<NR1>`

Examples

`:View1:Marker:Data?`
might return 3.

Related Commands

`:View<x>:Marker:Symbol?`, `:View<x>:Marker:T?`, `:View<x>:Symbol`

:View<x>:Marker:Symbol? (Query Only)

Queries the marker position on the symbol table. This command is the same as the `:View<x>:Symbol?` command, and exists for compatibility.

Syntax `:View<x>:Marker:Symbol?`

Returns `<NR1>`

Examples `:View1:Marker:Symbol?`
might return 110, indicating that the marker is positioned at the symbol number 110.

Related Commands `:View<x>:Marker:Data?`, `:View<x>:Marker:T?`, `:View<x>:Symbol?`

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

Syntax `:View<x>:Marker:T <value>`

`:View<x>:Marker:T?`

Arguments `<value>::=<NR3>` ranges the minimum to the maximum value on the time axis.

Examples `:View1:Marker:T 5u`
positions the marker at 5 μs.

Related Commands `:View<x>:Marker:Data?`, `:View<x>:Marker:Symbol?`, `:View<x>:Symbol`

:View<x>:Radix (?)

Selects or queries the display data format.

Syntax

`:View<x>:Radix { Hex | Oct | Bin }`

`:View<x>:Radix?`

Arguments

`Hex` selects the hexadecimal format.

`Oct` selects the octal format.

`Bin` selects the binary format.

Examples

`:View1:Radix Hex`
selects the hexadecimal format to display data.

Related Commands

`:View<x>:Marker:Data?`

:View<x>:Rotate (?)

Sets or queries the numeric value start position. It is unavailable for the 1/4 π QPSK and GMSK modulating system.

Syntax

`:View<x>:Rotate <value>`

`:View<x>:Rotate?`

Arguments

`<value>::=<NR1>` ranges 0 to 3.

Examples

`:View1:Rotate 3`
sets the numeric value start position to 3.

Related Commands

`:View<x>:Marker:Data?`

:View<x>:Source (?)

Selects or queries the data source for the symbol table.

Syntax

`:View<x>:Source { Measurement | Reference }`

`:View<x>:Source?`

Arguments

`Measurement` selects the measurement data as the view source. The data is stored in the register pair specified with the `:View<x>:MeasDestination` in the Polar View or the CDMA Polar View command group.

`Reference` selects the reference data as the view source. The data is stored in the register pair specified with the `:View<x>:RefDestination` in the Polar View or the CDMA Polar View command group.

Examples

`:View1:Source Measurement`
selects the measurement data as the view source.

Related Commands

`:View<x>:MeasDestination` and `:View<x>:RefDestination` in the Polar View or the CDMA Polar View command group

:View<x>:Symbol (?)

Sets or queries the symbol location where the marker is placed.

Syntax

`:View<x>:Symbol <value>`

`:View<x>:Symbol?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of symbols – 1.

Examples

`:View1:Symbol 10`
places the marker on the symbol number 10.

Related Commands

`:View<x>:Marker:Symbol?`, `:View<x>:Marker:T`

:View<x>:Version? (Query Only)

Queries the version of the Symbol Table View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

EVM View Commands

When you select `EVM` in the `:Config:View<x>` command, use the commands in this section to control the EVM (Error Vector Magnitude) view.

For information on the EVM view, refer to the user manual.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
D8 | <file_name> }
```

Arguments

`Clipboard` stores the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D8` select the data register 1 to 8, respectively.

`<file_name>::=<string>` is "*.TXT" (ASCII text file). The extension is set to ".TXT" automatically.

Examples

`:View1:CopyTo Clipboard`
stores the display data to the clipboard.

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { EVM | MagError | PhaseError }
```

```
:View<x>:Format?
```

Arguments

Defines the parameters associated with the horizontal and vertical axes as follows:

Argument	Display	Horizontal axis	Vertical axis
EVM	Error vector magnitude	Time	Percentage of EVM
MagError	Amplitude error	Time	Percentage of amplitude error
PhaseError	Phase error	Time	Percentage of phase error

Examples

`:View1:Format EVM`
shows the waveform with time along the horizontal axis and percentage of EVM along the vertical axis.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`,
`:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
```

```
:View<x>:Marker:DeltaX?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:DeltaX 5u`
positions the delta marker at 5 μs.

Related Commands

`:View<x>:Marker:DeltaY?`

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

Syntax
`:View<x>:Marker:DeltaY?`

Returns
`<NR3>`

Examples
`:View1:Marker:DeltaY?`
might return `1.78`.

Related Commands
`:View<x>:Marker:DeltaX`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples

`:View1:Marker:X 5u`
positions the marker at 5 μs.

Related Commands

`:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax

```
:View<x>:Marker:Y?
```

Returns

`<NR3>` in percent (%).

Examples

`:View1:Marker:Y?`
might return `1.543`, indicating that the marker is positioned at 1.543 %.

Related Commands

`:View<x>:Marker:X`

:View<x>:Mask:Left (?)

Sets or queries the left edge of the EVM calculation range in the symbol number. See Figure 4–7.

Syntax

`:View<x>:Mask:Left <value>`

`:View<x>:Mask:Left?`

Arguments

`<value>::=<NR1>` ranges 0 to 2,147,483,647 symbols.

Examples

`:View1:Mask:Left 5`
sets the left edge of the EVM calculation range to the 5th symbol.

Related Commands

`:View<x>:Mask:Right`, `:View<x>:Mask:Type`

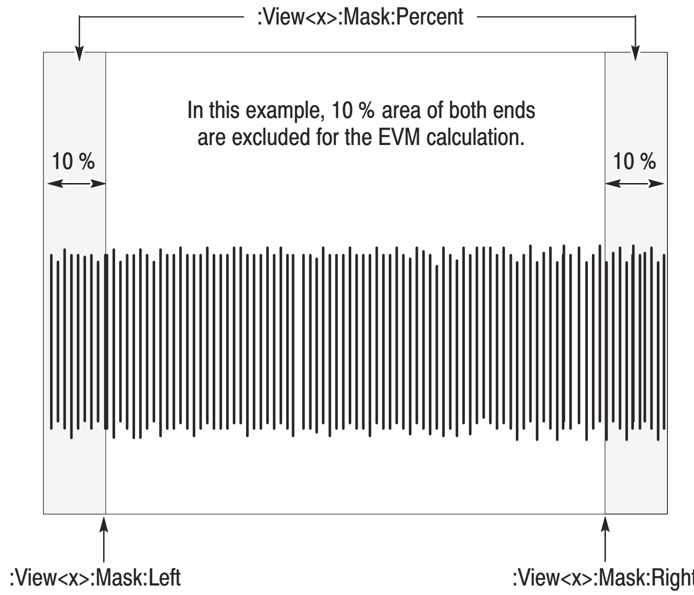


Figure 4–7: Setting the mask for the EVM calculation

***NOTE**. If you set the mask with the* `:View<x>:Mask:Percent` *command, select* `Percent` *with the* `:View<x>:Mask:Type` *command. If* `:View<x>:Mask:Left` *and* `:View<x>:Mask:Right`*, select* `Symbol`*. Refer to* `:View<x>:Mask:Type`.

:View<x>:Mask:Percent (?)

Sets or queries the area on display to exclude from calculating EVM.
See Figure 4–7.

Syntax

`:View<x>:Mask:Percent <value>`

`:View<x>:Mask:Percent?`

Arguments

`<value>::=<NR3>` ranges from 0 to 50 %. Both right and left ends are excluded. 0 % excludes no area. 50 % excludes the whole area.

Examples

`:View1:Mask:Percent 10`
excludes 10 % area from both right and left edges for the EVM calculation.

Related Commands

`:View<x>:Mask:Type`

:View<x>:Mask:Right (?)

Sets or queries the right edge of the EVM calculation range in the symbol number. See Figure 4–7.

Syntax

`:View<x>:Mask:Right <value>`

`:View<x>:Mask:Right?`

Arguments

`<value>::=<NR1>` ranges 0 to 2,147,483,647 symbols.

Examples

`:View1:Mask:Right 105`
sets the right edge of the EVM calculation range to the 105th symbol.

Related Commands

`:View<x>:Mask:Left`, `:View<x>:Mask:Type`

:View<x>:Mask:Type (?)

Sets or queries how to set the mask.

Syntax

`:View<x>:Mask:Type { Percent | Symbol }`

`:View<x>:Mask:Type?`

Arguments `Percent` sets the mask with the `:View<x>:Mask:Percent` command.

`Symbol` sets the mask with the `:View<x>:Mask:Left` and `:View<x>:Mask:Right` commands.

Examples `:View1:Mask:Type Percent`
sets the mask with the `:View<x>:Mask:Percent` command.

Related Commands `:View<x>:Mask:Percent`, `:View<x>:Mask:Left`, `:View<x>:Mask:Right`

:View<x>:Result<y>? (Query Only)

Queries the measurement results.

Syntax `:View<x>:Result<y>?`

Returns `<NR3>`

`Result1` returns the EVM in percent rms.

`Result2` returns the magnitude error in percent rms.

`Result3` returns the phase error in degree.

`Result4` returns the ϱ (Rho).

Examples `:View1:Result1?`
might return `2.913`, indicating that the EVM is 2.913 % rms.

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:XScale <value>
```

```
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples

`:View1:Scale:XScale 640u`
sets the horizontal axis full-scale to 640 μs.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

Syntax

```
:View<x>:Scale:XStart <value>
```

```
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples

`:View1:Scale:XStart 160u`
sets the value represented by the left edge of the horizontal axis to 160 μs.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>

:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 50`
sets the vertical axis full-scale to 50 %.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax

```
:View<x>:Scale:YStart <value>

:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum value on the vertical axis of the acquired data.

Examples

`:View1:Scale:YStart 0`
sets the value represented by the bottom edge of the vertical axis to 0 %.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Symbol (?)

Positions the marker with the symbol number.

Syntax

`:View<x>:Symbol <value>`

`:View<x>:Symbol?`

Arguments

`<value>::=<NR1>`

Examples

`:View1:Symbol 10`
puts the marker on the 10^{th} symbol.

:View<x>:Version? (Query Only)

Queries the version of the EVM View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

CDMA Waveform View Commands

When you select `CDMAWaveform` in the `:Config:View<x>` command, use the commands in this section to control the CDMA Waveform view.

:View<x>:Average:Times (?)

Sets or queries the number of frames that make up an averaged waveform.

Syntax

```
:View<x>:Average:Times <value>
```

```
:View<x>:Average:Times?
```

Arguments

`<value>::=<NR1>` is the number of frames for averaging, from 1 to 1000.

Examples

```
:View1:Average:Times 1000
```
specifies that an averaged waveform will show the result of combining 1000 frames.

Related Commands

`:View<x>:Average:Type`

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode.

The command usage is the same as that of the :View<x>:Average:Type command in the Waveform view. Refer to page 4–124.

:View<x>:CDMA:Channel (?)

Selects or queries the channel.

Syntax

```
:View<x>:CDMA:Channel <value>
```

```
:View<x>:CDMA:Channel?
```

Arguments

`<value>::=<NR1>` specifies the channel number.

For the IS-95 standard, the range is 1 to 777. Channel 1 and 777 correspond to 825.03 and 848.31 MHz, respectively. The frequency difference between two adjacent channels is 0.03 MHz.

For the T-53 standard, the range is 1 to 1199. Channel 1 and 1199 correspond to 915.0125 and 888.9875 MHz, respectively. The frequency difference between two adjacent channels is 0.0125 MHz.

Examples `:View1:CDMA:Channel 777`
sets the channel number to 777.

Related Commands `:Setup:CDMA:Channel`

:View<x>:CDMA:Standard (?)

Selects or queries the CDMA standard.

Syntax `:View<x>:CDMA:Standard { IS95 | T53 }`

`:View<x>:CDMA:Standard?`

Arguments `IS95` selects the IS-95 standard.

`T53` selects the T-53 standard.

Examples `:View1:CDMA:Standard IS95`
selects the IS-95 standard.

:View<x>:Compression (?)

Selects or queries the display data compression method.

The command usage is the same as that of the :View<x>:Compression command in the Waveform view. Refer to page 4–124.

:View<x>:CopyFrom (No Query Form)

Loads the display data from the text file.

The command usage is the same as that of the :View<x>:CopyFrom command in the Waveform view. Refer to page 4–125.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register.

The command usage is the same as that of the :View<x>:CopyTo command in the Waveform view. Refer to page 4–125.

:View<x>:Format (?)

Sets or queries the waveform display format.

The command usage is the same as that of the :View<x>:Format command in the Waveform view. Refer to page 4–129.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

The command usage is the same as that of the :View<x>:Marker:DeltaMarker command in the Waveform view. Refer to page 4–135.

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaX command in the Waveform view. Refer to page 4–136.

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaY? command in the Waveform view. Refer to page 4–136.

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X command in the Waveform view. Refer to page 4–140.

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

The command usage is the same as that of the :View<x>:Marker:Y? command in the Waveform view. Refer to page 4–140.

:View<x>:Mask:RBW1M:Frequency (?)

Sets or queries the frequency of the reference line at 1 MHz of resolution bandwidth (See Figure 4–8).

Syntax

```
:View<x>:Mask:RBW1M:Frequency <value>
:View<x>:Mask:RBW1M:Frequency?
```

Arguments `<value>::=<NR3>` ranges 0 to 25 MHz.

Examples `:View1:Mask:RBW1M:Frequency 25M`
sets the frequency of the reference line to 25 MHz at 1 MHz of RBW.

Related Commands `:View<x>:Mask:RBW1M:Level`

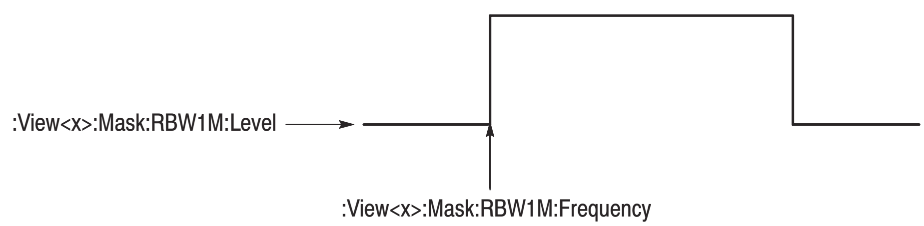


Figure 4–8: Setting the reference line at 1 MHz of RBW

:View<x>:Mask:RBW1M:Level (?)

Sets or queries the level of the reference line at 1 MHz of resolution bandwidth (See Figure 4–8).

Syntax

```
:View<x>:Mask:RBW1M:Level <value>
:View<x>:Mask:RBW1M:Level?
```

Arguments `<value>::=<NR3>` ranges –100 dBm to 30dBm.

Examples `:View1:Mask:RBW1M:Level 30`
sets the level of the reference line to 30 dBm at 1 MHz of RBW.

Related Commands `:View<x>:Mask:RBW1M:Frequency`

:View<x>:Mask:RBW30k:Frequency1 (?)

Sets or queries the frequency of the inner reference line at 30 kHz of resolution bandwidth (See Figure 4–9).

Syntax

```
:View<x>:Mask:RBW30k:Frequency1 <value>
:View<x>:Mask:RBW30k:Frequency1?
```

Arguments `<value>::=<NR3>` ranges 0 to 25 MHz.

Examples

`:View1:Mask:RBW30k:Frequency1 25M`
sets the frequency of the inner reference line to 25 MHz at 30 kHz of RBW.

Related Commands `:View<x>:Mask:RBW30k:Frequency2`

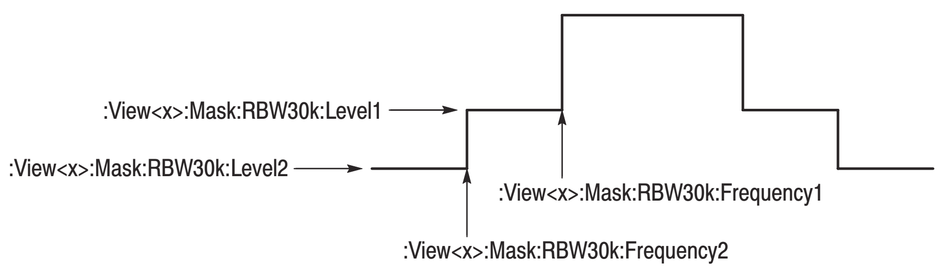


Figure 4–9: Setting the reference line at 30 kHz of RBW

:View<x>:Mask:RBW30k:Frequency2 (?)

Sets or queries the frequency of the outer reference line at 30 kHz of resolution bandwidth (See Figure 4–9).

Syntax

```
:View<x>:Mask:RBW30k:Frequency2 <value>
:View<x>:Mask:RBW30k:Frequency2?
```

Arguments `<value>::=<NR3>` ranges 0 to 25 MHz.

Examples
`:View1:Mask:RBW30k:Frequency2 20M`
sets the frequency of the outer reference line to 20 MHz at 30 kHz of RBW.

Related Commands
`:View<x>:Mask:RBW30k:Frequency1`

:View<x>:Mask:RBW30k:Level1 (?)

Sets or queries the level of the inner reference line at 30 kHz of resolution bandwidth (See Figure 4–9).

Syntax
`:View<x>:Mask:RBW30k:Level1 <value>`

`:View<x>:Mask:RBW30k:Level1?`

Arguments
`<value>::=<NR3>` ranges –100 dB to 0 dB.

Examples
`:View1:Mask:RBW30k:Level1 -42`
sets the level of the inner reference line to –42 dB at 30 kHz of RBW.

Related Commands
`:View<x>:Mask:RBW30k:Level2`

:View<x>:Mask:RBW30k:Level2 (?)

Sets or queries the level of the outer reference line at 30 kHz of resolution bandwidth (See Figure 4–9).

Syntax
`:View<x>:Mask:RBW30k:Level2 <value>`

`:View<x>:Mask:RBW30k:Level2?`

Arguments
`<value>::=<NR3>` ranges –100 dB to 0 dB.

Examples
`:View1:Mask:RBW30k:Level2 -54`
sets the level of the outer reference line to –54 dB at 30 kHz of RBW.

Related Commands
`:View<x>:Mask:RBW30k:Level1`

:View<x>:Measurement (?)

Selects or queries the measurement function. The measurement starts with the data acquisition. Query the results with the `:View<x>:Result<y>?` command.

Syntax

```
:View<x>:Measurement { Off | Power | Spurious }
:View<x>:Measurement?
```

Arguments

`Off` turns the measurement off.

`Power` selects the power measurement.

`Spurious` selects the spurious measurement.

Examples

`:View1:Measurement Power`
selects the power measurement.

Related Commands

`:View<x>:Measurement:OBW`, `:View<x>:Result<y>?`

:View<x>:Measurement:OBW (?)

Sets or queries the occupied bandwidth.

Syntax

```
:View<x>:Measurement:OBW <value>
:View<x>:Measurement:OBW?
```

Arguments

`<value>::=<NR2>` ranges 90 to 99.8 %.

Examples

`:View1:Measurement:OBW 99.8`
sets the occupied bandwidth to 99.8 %.

Related Commands

`:View<x>:Measurement`

:View<x>:Measurement:Separation (?)

Sets or queries the frequency resolution for spurious search to distinguish two adjacent peaks. The resolution is represented by the ratio (%) to the span.

Syntax

```
:View<x>:Measurement:Separation <value>

:View<x>:Measurement:Separation?
```

Arguments

`<value>::=<NR2>` ranges 0 to 100 %. 100 % means that the resolution is equal to the span.

Examples

`:View1:Measurement:Separation 100`
sets the spurious resolution to 100 % (= span).

Related Commands

`:View<x>:Measurement`, `:View<x>:Measurement:SpuriousSearch`

:View<x>:Measurement:SortedBy (?)

Sets or queries the sort key for the spurious signals.

Syntax

```
:View<x>:Measurement:SortedBy { Level | Frequency }

:View<x>:Measurement:SortedBy?
```

Arguments

`Level` specifies that the spurious signals are sorted by level.

`Frequency` specifies that the spurious signals are sorted by frequency.

Examples

`:View1:Measurement:SortedBy Level`
specifies that the spurious signals are sorted by level.

Related Commands

`:View<x>:Measurement`, `:View<x>:Measurement:SpuriousSearch`

:View<x>:Measurement:SpuriousSearch (?)

Determines whether or not to search spurious for each frame.

Syntax

```
:View<x>:Measurement:SpuriousSearch { On | Off }
:View<x>:Measurement:SpuriousSearch?
```

Arguments

`On` turns the spurious search on.

`Off` turns the spurious search off.

Examples

`:View1:Measurement:SpuriousSearch On`
turns the spurious search on.

Related Commands

`:View<x>:Measurement:Threshold`

:View<x>:Measurement:Threshold (?)

Sets or queries the threshold level to detect spurious.

Syntax

```
:View<x>:Measurement:Threshold <value>
:View<x>:Measurement:Threshold?
```

Arguments

`<value>::=<NR1>` ranges –150 to 30 dB.

Examples

`:View1:Measurement:Threshold 30`
sets the threshold level to 30 dB for the spurious search.

Related Commands

`:View<x>:Measurement:SpuriousSearch`

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

The command usage is the same as that of the :View<x>:Position command in the Waveform view. Refer to page 4–144.

:View<x>:RBW (?)

Sets or queries the resolution bandwidth.

Syntax

```
:View<x>:RBW { 3M | 1M | 300k | 100k | 30k | Off }
```

```
:View<x>:RBW?
```

Arguments

`1M` sets the resolution bandwidth to 1 MHz.

`30k` sets the resolution bandwidth to 30 kHz.

`Off` displays the raw data of each bin.

Examples

`:View1:RBW 1M`
sets the resolution bandwidth to 1 MHz.

:View<x>:Result<y>? (Query Only)

Queries the measurement results. Select the measurement function with the `:View<x>:Measurement` command.

Syntax `:View<x>:Result<y>?`

Returns `<NR3>`

<y>=1 returns the power.

<y>=2 returns the occupied bandwidth.

<y>=3, 6, 9, 12, 15, 18, 21, or 24 returns the result of spurious test; Pass or Fail. Each number corresponds to each of eight spurious signals. "Pass" indicates that the waveform is within the specified limits. "Fail" indicates that the waveform exceeds the limits.

<y>=4, 7, 10, 13, 16, 19, 22, or 25 returns each spurious level of eight spurious signals.

<y>=5, 8, 11, 14, 17, 20, 23, or 26 returns each spurious frequency of eight spurious signals.

Examples `:View1:Result5?`
might return `825000000`, indicating that the first spurious frequency is 825 MHz.

Related Commands `:View<x>:Measurement`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYcale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the Waveform view. Refer to page 4–149.

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

The command usage is the same as that of the :View<x>:Scale:XStart command in the Waveform view. Refer to page 4–149.

:View<x>:Scale:XStartZero (?)

Determines whether to put the time at the left edge of each frame to zero.

The command usage is the same as that of the :View<x>:Scale:XStartZero command in the Waveform view. Refer to page 4–150.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:YScale command in the Waveform view. Refer to page 4–151.

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

The command usage is the same as that of the :View<x>:Scale:YStart command in the Waveform view. Refer to page 4–151.

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Average | Zoom | D1D2 |
D3D4 | D5D6 | D7D8 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
<file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Average` specifies the averaged data as the source. When you select this item, you have to set the average type and the number of averages using the `:View<x>:Average:Type` and the `:View<x>:Average:Times` commands, respectively.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

`<file_name>::=<string>` specifies the file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the view source.

Related Commands

`:Config:Mode`, `:View<x>:Average:Times`, `:View<x>:Average:Type`

:View<x>:Version? (Query Only)

Queries the version of the CDMA Waveform View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

The command usage is the same as that of the :View<x>:Z command in the Waveform view. Refer to page 4–156.

CDMA Polar View Commands

When you select `CDMAPolar` in the `:Config:View<x>` command, use the commands in this section to control the CDMA Polar view.

For information on the Polar view, refer to the user manual.

:View<x>:AlphaBT (?)

Sets or queries the α/BT value.

The command usage is the same as that of the :View<x>:AlphaBT command in the Polar view. Refer to page 4–208.

:View<x>:AutoCarrier (?)

Determines whether to search the carrier automatically for each frame.

The command usage is the same as that of the :View<x>:AutoCarrier command in the Polar view. Refer to page 4–208.

:View<x>:Burst:BlockSize (?)

Sets or queries the range for searching a burst signal in frames.

The command usage is the same as that of the :View<x>:Burst:BlockSize command in the Polar view. Refer to page 4–209.

:View<x>:Burst:NumberFrames (?)

Sets or queries the range for acquiring burst data.

The command usage is the same as that of the :View<x>:Burst:NumberFrames command in the Polar view. Refer to page 4–209.

:View<x>:Burst:Offset (?)

Sets or queries the start point for acquiring burst data relative to the rising edge of the burst signal.

The command usage is the same as that of the :View<x>:Burst:Offset command in the Polar view. Refer to page 4–210.

:View<x>:Burst:PeakThreshold (?)

Sets or queries the threshold to determine whether the input signal is a burst.

The command usage is the same as that of the :View<x>:Burst:PeakThreshold command in the Polar view. Refer to page 4–210.

:View<x>:Burst:Search (?)

Determines whether to search a burst signal.

The command usage is the same as that of the :View<x>:Burst:Search command in the Polar view. Refer to page 4–211.

:View<x>:Burst:Threshold (?)

Sets or queries the threshold for determining the rising edge of a burst signal.

The command usage is the same as that of the :View<x>:Burst:Threshold command in the Polar view. Refer to page 4–212.

:View<x>:Carrier (?)

Sets or queries the carrier frequency.

The command usage is the same as that of the :View<x>:Carrier command in the Polar view. Refer to page 4–212.

:View<x>:Display (?)

Selects or queries the display data source.

The command usage is the same as that of the :View<x>:Display command in the Polar view. Refer to page 4–212.

:View<x>:Format (?)

Selects or queries the display format.

The command usage is the same as that of the :View<x>:Format command in the Polar view. Refer to page 4–213.

:View<x>:Marker:A? (Query Only)

Queries the amplitude at the marker position.

The command usage is the same as that of the :View<x>:Marker:A? command in the Polar view. Refer to page 4–213.

:View<x>:Marker:DeltaT (?)

Sets or queries the delta-marker position on the time axis.

The command usage is the same as that of the :View<x>:Marker:DeltaT command in the Polar view. Refer to page 4–214.

:View<x>:Marker:P? (Query Only)

Queries the phase at the marker position.

The command usage is the same as that of the :View<x>:Marker:P? command in the Polar view. Refer to page 4–214.

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

The command usage is the same as that of the :View<x>:Marker:T command in the Polar view. Refer to page 4–214.

:View<x>:Marker:X? (Query Only)

Queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X? command in the Polar view. Refer to page 4–215.

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

The command usage is the same as that of the :View<x>:Marker:Y? command in the Polar view. Refer to page 4–215.

:View<x>:Mask:Frequency (?)

Determines whether the frequency-domain mask is enabled.

The command usage is the same as that of the :View<x>:Mask:Frequency command in the Polar view. Refer to page 4–216.

:View<x>:Mask:Frequency:Center (?)

Sets or queries the center frequency of the mask area.

The command usage is the same as that of the :View<x>:Mask:Frequency:Center command in the Polar view. Refer to page 4–217.

:View<x>:Mask:Frequency:Left (?)

Sets or queries the left frequency of the mask area.

The command usage is the same as that of the :View<x>:Mask:Frequency:Left command in the Polar view. Refer to page 4–217.

:View<x>:Mask:Frequency:Right (?)

Sets or queries the right frequency of the mask area.

The command usage is the same as that of the :View<x>:Mask:Frequency:Right command in the Polar view. Refer to page 4–218.

:View<x>:Mask:Frequency:Width (?)

Sets or queries the bandwidth of the mask area.

The command usage is the same as that of the :View<x>:Mask:Frequency:Width command in the Polar view. Refer to page 4–218.

:View<x>:Mask:MarkerLink (?)

Determines whether the center frequency of the mask area is set with the marker in the other view.

The command usage is the same as that of the :View<x>:Mask:MarkerLink command in the Polar view. Refer to page 4–219.

:View<x>:MeasDestination (?)

Selects or queries the register pair to store the measurement data.

The command usage is the same as that of the :View<x>:MeasDestination command in the Polar view. Refer to page 4–220.

:View<x>:MeasFilter (?)

Selects or queries the filter to create the measurement data.

The command usage is the same as that of the :View<x>:MeasFilter command in the Polar view. Refer to page 4–220.

:View<x>:Modulation (?)

Selects or queries the digital modulation format.

Syntax

```
:View<x>:Modulation { PI4QPSK | BPSK | QPSK | PSK8 | QAM16 |
QAM64 | QAM256 | GMSK | GFSK | CDMAOQPSK }
```

```
:View<x>:Modulation?
```

Arguments

`PI4QPSK` selects 1/4 π Shift QPSK (Quadrature Phase Shift Keying) modulation.

`BPSK` selects BPSK (Binary Phase Shift Keying) modulation.

`QPSK` selects QPSK (Quadrature Phase Shift Keying) modulation.

`PSK8` selects 8PSK (Phase Shift Keying) modulation.

`QAM16` selects 16QAM (Quadrature Amplitude Modulation).

`QAM64` selects 64QAM (Quadrature Amplitude Modulation).

`QAM256` selects 256QAM (Quadrature Amplitude Modulation).

`GMSK` selects GMSK (Gaussian-filtered Minimum Shift Keying) modulation.

`GFSK` selects GFSK (Gaussian-filtered Frequency Shift Keying) modulation.

`CDMAOQPSK` selects CDMA OQPSK (Offset QPSK) used for the IS-95 standard.

Examples

`:View1:Modulation PI4QPSK`
selects the 1/4 π Shift QPSK modulation.

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

The command usage is the same as that of the :View<x>:Position command in the Polar view. Refer to page 4–221.

:View<x>:RefDestination (?)

Selects or queries the register pair to store the reference data.

The command usage is the same as that of the :View<x>:RefDestination command in the Polar view. Refer to page 4–222.

:View<x>:RefFilter (?)

Selects or queries the filter to create the reference data.

Syntax

```
:View<x>:RefFilter { None | RaisedCosine | Gaussian | IS95}
```

```
:View<x>:RefFilter?
```

Arguments

`None` selects no filter.

`RaisedCosine` selects the raised cosine filter.

`Gaussian` selects the Gaussian filter.

`IS95` selects the filter specified in the IS-95 standard.

Examples

`:View1:RefFilter RaisedCosine`
selects the raised cosine filter to create the reference data.

Related Commands

`:View<x>:MeasFilter`

:View<x>:Result<y>? (Query Only)

Queries the measurement result.

The command usage is the same as that of the :View<x>:Result<y>? command in the Polar view. Refer to page 4–223.

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

The command usage is the same as that of the :View<x>:Source command in the Polar view. Refer to page 4–223.

:View<x>:Standard:CDPD (No Query Form)

Configures the modulating system according to the CDPD (Cellular Digital Packet Data) standard.

The command usage is the same as that of the :View<x>:Standard:CDPD command in the Polar view. Refer to page 4–224.

:View<x>:Standard:GSM (No Query Form)

Configures the modulating system according to the GSM (Global System for Mobile Communication) standard.

The command usage is the same as that of the :View<x>:Standard:GSM command in the Polar view. Refer to page 4–224.

:View<x>:Standard:IS95 (No Query Form)

Configures the modulating system according to the IS-95 standard.

Syntax

```
:View<x>:Standard:IS95
```

Arguments

None

Examples

```
:View1:Standard:IS95
```

configures the modulating system according to the IS-95 standard.

:View<x>:Standard:NADC (No Query Form)

Configures the modulating system according to the NADC (North American Digital Cellular) standard.

The command usage is the same as that of the :View<x>:Standard:NADC command in the Polar view. Refer to page 4–224.

:View<x>:Standard:PDC (No Query Form)

Configures the modulating system according to the PDC (Personal Digital Cellular System) standard.

The command usage is the same as that of the :View<x>:Standard:PDC command in the Polar view. Refer to page 4–225.

:View<x>:Standard:PHS (No Query Form)

Configures the modulating system according to the PHS (Personal Handy Phone System) standard.

The command usage is the same as that of the :View<x>:Standard:PHS command in the Polar view. Refer to page 4–225.

:View<x>:Standard:TETRA (No Query Form)

Configures the modulating system according to the TETRA (Trans-European Trunked Radio) standard.

The command usage is the same as that of the :View<x>:Standard:TETRA command in the Polar view. Refer to page 4–225.

:View<x>:SymbolRate (?)

Sets or queries the symbol rate for the digital modulation.

The command usage is the same as that of the :View<x>:SymbolRate command in the Polar view. Refer to page 4–226.

:View<x>:Version?

Queries the version of the CDMA Polar View program.

Syntax

```
:View<x>:Version?
```

Returns

```
<NR2>
```

Examples

```
:View1:Version?
```

might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

The command usage is the same as that of the :View<x>:Z command in the Polar view. Refer to page 4–226.

CDMA Time View Commands

When you select `CDMATime` in the `:Config:View<x>` command, use the commands in this section to control the CDMA Time view.

:View<x>:Average:Times (?)

Sets or queries the number of frames that make up an averaged waveform.

Syntax

```
:View<x>:Average:Times <value>

:View<x>:Average:Times?
```

Arguments `<value>::=<NR1>` is the number of frames for averaging, from 1 to 1000.

Examples `:View1:Average:Times 1000`
specifies that an averaged waveform will show the result of combining 1000 frames.

:View<x>:Block (?)

Sets or queries the block number.

Syntax

```
:View<x>:Block <value>

:View<x>:Block?
```

Arguments `<value>::=<NR1>` ranges from 0 to the number of acquired blocks – 1.

Examples `:View1:Block 0`
sets the block number to 0.

:View<x>:BreakMeasure (No Query Form)

Stops the Measure function executed with the `:View<x>:Measure` command.

Syntax

```
:View<x>:BreakMeasure
```

Arguments None

Examples `:View1:BreakMeasure`
stops the Measure function.

Related Commands `:View<x>:Measure`

:View<x>:BreakMeasureData (No Query Form)

Stops the Measure Data function executed with the `:View<x>:MeasureData` command.

Syntax
`:View<x>:BreakMeasureData`

Arguments
None

Examples
`:View1:BreakMeasureData`
stops the Measure Data function.

Related Commands
`:View<x>:MeasureData`

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:DeltaX`, `:View<x>:Marker:DeltaY?`, `:View<x>:Marker:ResetDelta`

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaX <value>
```

```
:View<x>:Marker:DeltaX?
```

Arguments `<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples `:View1:Marker:DeltaX 5u`
positions the delta marker at 5 μs.

Related Commands `:View<x>:Marker:DeltaY?`

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

Syntax

```
:View<x>:Marker:DeltaY?
```

Returns `<NR3>`

Examples `:View1:Marker:DeltaY?`
might return `-50.23`, indicating that the delta marker is positioned at –50.23 dBm.

Related Commands `:View<x>:Marker:DeltaX`

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
:View<x>:Marker:X?
```

Arguments `<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis.

Examples `:View1:Marker:X 5u`
positions the marker at 5 μs.

Related Commands `:View<x>:Marker:Y?`

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

Syntax `:View<x>:Marker:Y?`

Returns `<NR3>`

Examples `:View1:Marker:Y?`
might return `-50.23`, indicating that the marker is positioned at –50.23 dBm.

Related Commands `:View<x>:Marker:X`

:View<x>:Mask:OffLeft (?)

Sets or queries the mask time just before the rising edge of waveform (see Figure 4–10).

Syntax

`:View<x>:Mask:OffLeft <value>`

`:View<x>:Mask:OffLeft?`

Arguments

`<value>::=<NR3>` ranges 0 to 1.6 ms.

Examples

`:View1:Mask:OffLeft 0.2m`
sets the mask time just before the rising edge of waveform to 0.2 ms.

Related Commands

`:View<x>:Mask:OffRight`

:View<x>:Mask:OffLevel (?)

Sets or queries the mask level for the off state (see Figure 4–10).

Syntax

`:View<x>:Mask:OffLevel <value>`

`:View<x>:Mask:OffLevel?`

Arguments

`<value>::=<NR3>` ranges –100 dB to 0 dB.

Examples

`:View1:Mask:OffLevel -60`
sets the mask level for the off state to –60 dB.

Related Commands

`:View<x>:Mask:OnLevel`

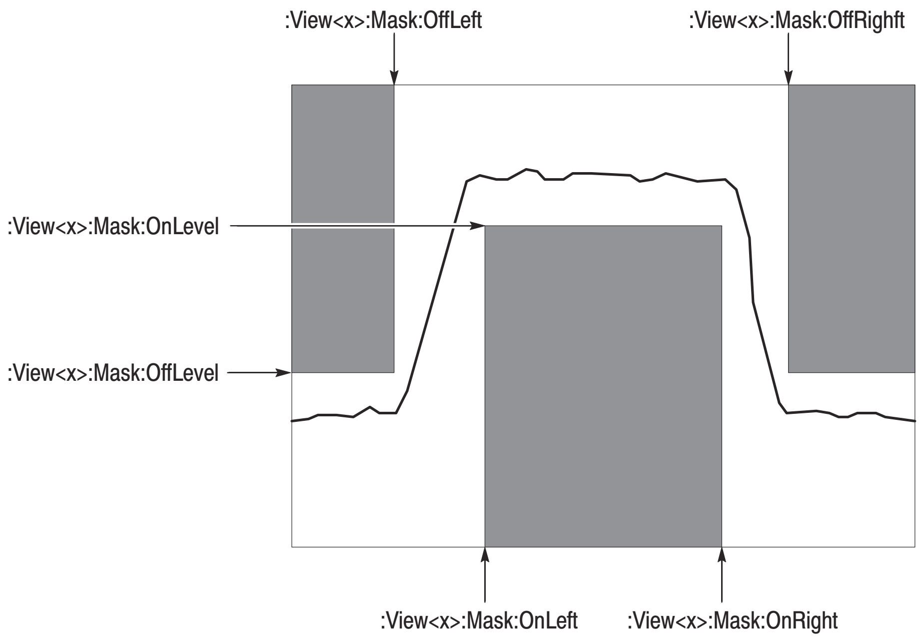


Figure 4–10: Setting the trigger mask

:View<x>:Mask:OffRight (?)

Sets or queries the mask time just after the falling edge of waveform (see Figure 4–10).

Syntax

```
:View<x>:Mask:OffRight <value>
```

```
:View<x>:Mask:OffRight?
```

Arguments

`<value>::=<NR3>` ranges 0 to 1.6 ms.

Examples

```
:View1:Mask:OffRight 1.2m
```

sets the mask time just after the falling edge of waveform to 1.2 ms.

Related Commands

```
:View<x>:Mask:OffLeft
```

:View<x>:Mask:OnLeft (?)

Sets or queries the mask time just after the rising edge of waveform (see Figure 4–10).

Syntax

```
:View<x>:Mask:OnLeft <value>
```

```
:View<x>:Mask:OnLeft?
```

Arguments `<value>::=<NR3>` ranges 0 to 1.6 ms.

Examples `:View1:Mask:OnLeft 0.8m`
sets the mask time just after the rising edge of waveform to 0.8 ms.

Related Commands `:View<x>:Mask:OnLeft`

:View<x>:Mask:OnLevel (?)

Sets or queries the mask level for the on state (see Figure 4–10).

Syntax

```
:View<x>:Mask:OnLevel <value>
```

```
:View<x>:Mask:OnLevel?
```

Arguments `<value>::=<NR3>` ranges –100 dB to 0 dB.

Examples `:View1:Mask:OnLevel -30`
sets the mask level for the on state to –30 dB.

Related Commands `:View<x>:Mask:OffLevel`

:View<x>:Mask:OnRight (?)

Sets or queries the mask time just before the falling edge of waveform (see Figure 4–10).

Syntax

`:View<x>:Mask:OnRight <value>`

`:View<x>:Mask:OnRight?`

Arguments

`<value>::=<NR3>` ranges 0 to 1.6 ms.

Examples

`:View1:Mask:OnRight 1.2m`
sets the mask time just before the falling edge of waveform to 1.2 ms.

Related Commands

`:View<x>:Mask:OnLeft`

:View<x>:Measure (No Query Form)

Measures the CDMA time characteristics of the input signal. The Pass/Fail decision is made by comparing the averaged waveform with the mask settings.

This command is executed in another thread. For synchronizing command execution, refer to page 5–9.

Syntax

`:View<x>:Measure`

Arguments

None

Examples

`:View1:Measure`
measures the input signal and performs the Pass/Fail test.

Related Commands

`:View<x>:Average:Times`, `:View<x>:Mask` command group, `:View<x>:MeasureData`

:View<x>:MeasureData (No Query Form)

Measures the CDMA time characteristics of the waveforms already stored in memory, such as loaded from a file. The Pass/Fail decision is made by comparing the averaged waveform with the mask settings.

This command is executed in another thread. For synchronizing command execution, refer to page 5–9.

Syntax

`:View<x>:MeasureData`

Arguments

None

Examples

`:View1:MeasureData`
measures the stored waveforms and performs the Pass/Fail test.

Related Commands

`:View<x>:Average:Times`, `:View<x>:Mask` command group, `:View<x>:Measure`

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

Syntax

`:View<x>:Position <value>`

`:View<x>:Position?`

Arguments

`<value>::=<NR2>` ranges 0 to 100 %. It specifies the display frame position relative to the block size. 0 % specifies that the first frame in a block is displayed. 100 % represents the last frame.

Examples

`:View1:Position 100`
displays the last frame in a block.

:View<x>:Result<y>? (Query Only)

Queries the measurement result.

Syntax
`:View<x>:Result<y>?`

Returns
`Result1::=<NR3>` returns the number of averages.

`Result2::=<NR3>` returns the average power of the on state.

`Result3::=<NR3>` returns the average power of the off state.

`Result4::=<NR3>` returns the on/off power ratio.

`Result5::={ Pass | Fail }` returns the result of spurious test.

Examples
`:View1:Result2?`
returns `-17.635`, indicating the average power of the on state is –17.635 dB.

Related Commands
`:View<x>:Measure`, `:View<x>:MeasureData`

:View<x>:Scale:FallingEdge (No Query Form)

Expands the falling edge of the waveform on screen.

Syntax
`:View<x>:Scale:FallingEdge`

Arguments
None

Examples
`:View1:Scale:FallingEdge`
expands the falling edge of the waveform on screen.

Related Commands
`:View<x>:Scale:RisingEdge`

:View<x>:Scale:FullScale (No Query Form)

Displays the whole waveform.

Syntax

`:View<x>:Scale:FullScale`

Arguments

None

Examples

`:View1:Scale:FullScale`
displays the whole waveform.

Related Commands

`:View<x>:Scale:RisingEdge`, `:View<x>:Scale:FallingEdge`

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

Syntax

`:View<x>:Scale:HoldYScale { On | Off }`

`:View<x>:Scale:HoldYScale?`

Arguments

`On` holds the vertical scale setting when you change the input source.

`Off` resets the vertical scale to full-scale.

Examples

`:View1:Scale:HoldYScale On`
holds the vertical scale setting when you change the input source.

Related Commands

`:View<x>:Scale:YScale`, `:View<x>:Scale:YStart`, `:View<x>:Source`

:View<x>:Scale:RisingEdge (No Query Form)

Expands the rising edge of the waveform on screen.

Syntax

`:View<x>:Scale:RisingEdge`

Arguments

None

Examples `:View1:Scale:RisingEdge`
expands the rising edge of the waveform on screen.

Related Commands `:View<x>:Scale:FallingEdge`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

Syntax `:View<x>:Scale:XScale <value>`

`:View<x>:Scale:XScale?`

Arguments `<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples `:View1:Scale:XScale 800u`
sets the horizontal axis full-scale to 800 μs.

Related Commands `:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

Syntax `:View<x>:Scale:XStart <value>`

`:View<x>:Scale:XStart?`

Arguments `<value>::=<NR3>` ranges from the minimum to the maximum value on the horizontal axis of the acquired data.

Examples `:View1:Scale:XStart 100u`
sets the value represented by the left edge of the horizontal axis to 100 μs.

Related Commands `:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges full-scale/100 to full-scale of the acquired data.

Examples

`:View1:Scale:YScale 20`
sets the vertical axis full-scale to 20 dB.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

Syntax

```
:View<x>:Scale:YStart <value>
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the reference level – 200 dB to the reference level + 100 dB.

Examples

`:View1:Scale:YStart -120`
sets the value represented by the bottom edge of the vertical axis to –120 dB.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | Active | Zoom | <file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the file as the source. The file name must be "*.IQ" (the IQ format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Config:Mode`

:View<x>:Trace2:TraceVisible (?)

Determines whether or not to display the Trace 2 (averaged waveform).

Syntax

```
:View<x>:Trace2:TraceVisible { On | Off }
```

```
:View<x>:Trace2:TraceVisible?
```

Arguments

`On` displays the Trace 2 (averaged waveform).

`Off` does not display the Trace 2.

Examples

`:View1:Trace2:TraceVisible On`
displays the Trace 2.

Related Commands

`:View<x>:TraceVisible`

:View<x>:TraceVisible (?)

Determines whether or not to display the Trace 1 (acquired waveform).

Syntax

```
:View<x>:TraceVisible { On | Off }
```

```
:View<x>:TraceVisible?
```

Arguments

`On` displays the Trace 1 (acquired waveform).

`Off` does not display the Trace 1.

Examples

`:View1:TraceVisible On`
displays the Trace 1.

Related Commands

`:View<x>:Trace2:TraceVisible`

:View<x>:Version? (Query Only)

Queries the version of the CDMA Time View program.

Syntax

```
:View<x>:Version?
```

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

CodeSpectrogram View Commands

When you select `CodeSpectrogram` in the `Config:View<x>` command, use the commands in this section to control the code-domain power spectrogram view. This view incorporates analysis functions for the cdmaOne standard.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:ResetDelta`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:TotalPower? (Query Only)

Queries the total power of the symbol at the marker position.

Syntax `:View<x>:Marker:TotalPower?`

Returns `<NR3>`

Examples `:View1:Marker:TotalPower?`
might return `-7.212`, indicating that the total power is –7.212 dB.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax `:View<x>:Marker:X <value>`

`:View<x>:Marker:X?`

Arguments `<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis (channel).

Examples `:View1:Marker:X 32`
positions the marker at channel 32.

Related Commands `:View<x>:Marker:Z`, `:View<x>:Scale:XScale`, `:View<x>:Scale:XStart`

:View<x>:Marker:Z (?)

Sets or queries the marker position on the vertical (symbol number) axis.

Syntax

```
:View<x>:Marker:Z <value>
```

```
:View<x>:Marker:Z?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of symbols – 1.

Examples

`:View1:Marker:Z 199`
positions the marker at the symbol number 199.

Related Commands

`:View<x>:Marker:X`, `:View<x>:Scale:ZScale`, `:View<x>:Scale:ZStart`

:View<x>:Monochrome (?)

Determines whether to display a spectrogram in monochrome.

The command usage is the same as that of the :View<x>:Monochrome command in the Spectrogram view. Refer to page 4–187.

:View<x>:NumberColors (?)

Selects or queries the number of display colors.

The command usage is the same as that of the :View<x>:NumberColors command in the Spectrogram view. Refer to page 4–187.

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the Y (color) axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Spectrogram view. Refer to page 4–188.

:View<x>:Scale:FullYScale (No Query Form)

Sets the Y (color) axis scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Spectrogram view. Refer to page 4–188.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel) full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:XScale <value>

:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR1>` ranges 16 to 64 channels.

Examples

`:View1:Scale:XScale 64`
sets the horizontal axis full-scale to 64 channels.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`, `:View<x>:Scale:ZScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel).

Syntax

```
:View<x>:Scale:XStart <value>

:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR1>` ranges from 0 to 64 – (the horizontal axis full-scale) channels.

Examples

`:View1:Scale:XStart 0`
sets the value represented by the left edge of the horizontal axis to channel 0.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the Y (color) axis full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:YScale <value>
```

```
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` is 10, 20, 50, or 100 dB.

Examples

`:View1:Scale:YScale 50`
sets the Y axis full-scale to 50 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`, `:View<x>:Scale:ZScale`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Y (color) axis.

Syntax

```
:View<x>:Scale:YStart <value>
```

```
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from ref level – 200 dB to ref level + 100 dB – horizontal axis full-scale.

Examples

`:View1:Scale:YStart -50`
sets the value represented by the bottom edge of the Y axis to –50 dBm.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:ZScale (?)

Sets or queries the symbol display interval. The spectrogram is displayed every specified number of symbols.

Syntax

```
:View<x>:Scale:ZScale <value>
```

```
:View<x>:Scale:ZScale?
```

Arguments

`<value>::=<NR1>` ranges 1 to 32.
1 means every symbol is displayed. 32 means every 32th symbol is displayed.

Examples

`:View1:Scale:ZScale 8`
displays a spectrogram every 8th symbol.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YScale`, `:View<x>:Scale:ZStart`

:View<x>:Scale:ZStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Z (symbol number) axis, i.e. the first symbol to be displayed.

Syntax

```
:View<x>:Scale:ZStart <value>
```

```
:View<x>:Scale:ZStart?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of symbols – 1.

Examples

`:View1:Scale:ZStart 20`
sets the value represented by the bottom edge of the Z axis to 20.

Related Commands

`:View<x>:Scale:ZScale`

:View<x>:Version? (Query Only)

Queries the version of the CodeSpectrogram View program.

Syntax

```
:View<x>:Version?
```

Returns

<NR2>

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the Y (color) axis.

Syntax

```
:View<x>:YAxis { Relative | Absolute }
```

```
:View<x>:YAxis?
```

Arguments

`Relative` represents relative channel power to the total power along the Y axis.

`Absolute` represents absolute channel power along the Y axis.

Examples

`:View1:YAxis Relative`
represents relative channel power to the total power along the Y axis.

:View<x>:ZGap (?)

Sets or queries the display interval between two adjacent symbols on the Z (symbol number) axis on screen.

Syntax

```
:View<x>:ZGap <value>
```

```
:View<x>:ZGap?
```

Arguments

`<value>::=<NR1>` ranges 1 to 10 pixels.

Examples

`:View1:ZGap 10`
sets the display interval between two adjacent symbols on the Z axis to 10 pixels.

CodePolar View Commands

When you select `CodePolar` in the `:Config:View<x>` command, use the commands in this section to control the polar view. This view incorporates analysis functions for the cdmaOne standard.

For information on the polar view, refer to the user manual.

:View<x>:AlphaBT (?)

Sets or queries the α/BT value.

The command usage is the same as that of the :View<x>:AlphaBT command in the Polar view. Refer to page 4–208.

:View<x>:Analysis:Symbol (?)

Specifies or queries the symbol number to display the constellation.

Syntax

```
:View<x>:Analysis:Symbol <value>
```

```
:View<x>:Analysis:Symbol?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of symbols – 1.

Examples

`:View1:Analysis:Symbol 1`
sets the symbol number to 1.

:View<x>:Analyze (No Query Form)

Performs analysis for all symbols acquired. This command is executed in another thread. For synchronizing command execution, refer to page 5–9.

Syntax

```
:View<x>:Analyze
```

Arguments

None

Examples

`:View1:Analyze`
performs analysis on the background for all symbols acquired.

Related Commands

`:View<x>:BreakAnalyze`

:View<x>:AutoCarrier (?)

Determines whether to search the carrier automatically for each frame.

The command usage is the same as that of the :View<x>:AutoCarrier command in the Polar view. Refer to page 4–208.

:View<x>:BreakAnalyze (No Query Form)

Breaks the analysis executed by the `:View<x>:Analyze` command.

Syntax

```
:View<x>:BreakAnalyze
```

Arguments

None

Examples

```
:View1:Analyze
```
breaks the analysis.

Related Commands

```
:View<x>:Analyze
```

:View<x>:Carrier (?)

Sets or queries the carrier frequency.

The command usage is the same as that of the :View<x>:Carrier command in the Polar view. Refer to page 4–212.

:View<x>:ChipRate (?)

Sets or queries the chip rate.

Syntax

```
:View<x>:ChipRate <value>
```

```
:View<x>:ChipRate?
```

Arguments

`<value>::=<NR3>` ranges 1/s to 30 M/s.

Examples

```
:View1:ChipRate 8M
```
sets the chip rate to 8 M/s.

:View<x>:Display (?)

Selects or queries the display data source.

The command usage is the same as that of the :View<x>:Display command in the Polar view. Refer to page 4–212.

:View<x>:Format (?)

Selects or queries the waveform display format.

The command usage is the same as that of the :View<x>:Format command in the Polar view. Refer to page 4–213.

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

Syntax

```
:View<x>:Marker:T <value>

:View<x>:Marker:T?
```

Arguments `<value>::=<NR3>` ranges 0 to the time length of the acquisition data.

Examples `:View1:Marker:T 5.4u`
positions the marker at 5.4 μs.

:View<x>:MeasFilter (?)

Selects or queries the filter to create the measurement data.

Syntax

```
:View<x>:MeasFilter { None | RootRaisedCosine }
```

```
:View<x>:MeasFilter?
```

Arguments

`None` selects no filter.

`RootRaisedCosine` selects the root raised-cosine filter.

Examples

`:View1:MeasFilter RootRaisedCosine`
selects the root raised-cosine filter to create the measurement data.

Related Commands

`:View<x>:RefFilter`

:View<x>:Modulation (?)

Selects or queries the modulation type.

Syntax

```
:View<x>:Modulation { IS95 | IS95+EQ }
```

```
:View<x>:Modulation?
```

Arguments

`IS95` selects the modulation without the equalizer according to the IS-95 standard.

`IS95+EQ` selects the modulation with the equalizer according to the IS-95 standard.

Examples

`:View1:Modulation IS95+EQ`
selects the modulation with the equalizer according to the IS-95 standard.

:View<x>:RefFilter (?)

Selects or queries the filter to create the reference data.

Syntax

`:View<x>:RefFilter { None | RaisedCosine | Gaussian | IS95 }`

`:View<x>:RefFilter?`

Arguments

`None` selects no filter.

`RaisedCosine` selects the raised cosine filter.

`Gaussian` selects the Gaussian filter.

`IS95` selects the IS-95 filter.

Examples

`:View1:RefFilter RaisedCosine`
selects the raised cosine filter to create the reference data.

Related Commands

`:View<x>:MeasFilter`

:View<x>:Source (?)

Selects or queries the input data source for the view.

Syntax

`:View<x>:Source { None | Active | Zoom | <file_name> }`

`:View<x>:Source?`

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the source.

:View<x>:Standard:IS95 (No Query Form)

Configures the modulating system without the equalizer according to the IS-95 standard.

Syntax `:View<x>:Standard:IS95`

Arguments None

Examples `:View1:Standard:IS95`
configures the modulating system without the equalizer according to the IS-95 standard.

Related Commands `:View<x>:Standard:IS95EQ`

:View<x>:Standard:IS95EQ (No Query Form)

Configures the modulating system with the equalizer according to the IS-95 standard.

Syntax `:View<x>:Standard:IS95EQ`

Arguments None

Examples `:View1:Standard:IS95EQ`
configures the modulating system with the equalizer according to the IS-95 standard.

Related Commands `:View<x>:Standard:IS95`

:View<x>:Version? (Query Only)

Queries the version of the CodePolar View program.

Syntax

```
:View<x>:Version?
```

Returns

```
<NR2>
```

Examples

```
:View1:Version?
```

might return 1.1.

CodePower View Commands

When you select `CodePower` in the `Config:View<x>` command, use the commands in this section to set the details for the code-domain power view. This view incorporates analysis functions for the cdmaOne standard.

:View<x>:Average (?)

Determines whether or not to display the results of averaging on the analyzer screen.

Syntax

:View<x>:Average { On | Off }

:View<x>:Average?

Arguments

On displays the results of averaging. The averaged power is displayed for each channel with a bar graph.

Off hides the results of averaging.

Examples

:View1:Average On
displays the results of averaging.

Related Commands

:View<x>:Average:Execute

:View<x>:Average:AllFrames (No Query Form)

Specifies that all of the acquired symbol data are used for averaging.

Syntax

:View<x>:AllFrames

Arguments

None

Examples

:View1:AllFrames
specifies that all of acquired symbol data are used for averaging.

Related Commands

:View<x>:Average:BeginZ, :View<x>:Average:EndZ

:View<x>:Average:BeginZ (?)

Sets or queries the uppermost symbol in the average range.

Syntax

```
:View<x>:Average:BeginZ <value>
:View<x>:Average:BeginZ?
```

Arguments `<value>::=<NR1>` ranges 0 to the number of symbols – 1.

Examples `:View1:Average:BeginZ 199`
sets the uppermost symbol number to 199.

Related Commands `:View<x>:Average:EndZ`

:View<x>:Average:EndZ (?)

Sets or queries the lowermost symbol in the average range.

Syntax

```
:View<x>:Average:EndZ <value>
:View<x>:Average:EndZ?
```

Arguments `<value>::=<NR1>` ranges 0 to the number of symbols – 1.

Examples `:View1:Average:EndZ 100`
sets the lowermost symbol number to 100.

Related Commands `:View<x>:Average:BeginZ`

:View<x>:Average:Execute (No Query Form)

Performs averaging for each bin for symbols in the specified range.

Syntax

`:View<x>:Average:Execute`

Arguments

None

Examples

`:View1:Average:Execute`
performs averaging for each bin for symbols in the specified range.

Related Commands

`:View<x>:Average:AllFrames`, `:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:MarkerToFrame (No Query Form)

Specifies that the symbols between the main marker and the delta marker are averaged.

Syntax

`:View<x>:MarkerToFrame`

Arguments

None

Examples

`:View<x>:MarkerToFrame`
specifies that the symbols between the main marker and the delta marker are averaged.

Related Commands

`:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:Times (?)

Sets or queries the number of waveform acquisitions that make up an averaged waveform.

Syntax

```
:View<x>:Average:Times <value>
:View<x>:Average:Times?
```

Arguments

`<value>::=<NR1>` ranges 1 to 1,000,000.

Examples

`:View1:Average:Times 1000`
specifies that an averaged waveform will show the result of combining 1000 separately acquired waveforms.

Related Commands

`:View<x>:Average:Type`

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode. For more information about averaging, refer to the user manual.

Syntax

```
:View<x>:Average:Type { RMSExpo | RMS | MaxHold | MinHold }
:View<x>:Average:Type?
```

Arguments

`RMSExpo` averages with the RMS (root mean squared) exponential. This mode weights older acquisition data so that they have a progressively smaller effect on the average.

`RMS` averages with the RMS (root mean squared).

`MaxHold` holds the maximum value for each data point.

`MinHold` holds the minimum value for each data point.

Examples

`:View1:Average:Type RMSExpo`
averages the waveform with the exponential RMS.

Related Commands

`:View<x>:Average:Time`

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:ResetDelta`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

`:View<x>:Marker:X <value>`

`:View<x>:Marker:X?`

Arguments

`<value>::=<NR1>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis (channel).

Examples

`:View1:Marker:X 32`
positions the marker at channel 32.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:XStart`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel) full-scale to display a portion of data.

Syntax

`:View<x>:Scale:XScale <value>`

`:View<x>:Scale:XScale?`

Arguments

`<value>::=<NR1>` ranges 16 to 64 channels.

Examples

`:View1:Scale:XScale 64`
sets the horizontal axis full-scale to 64 channels.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel).

Syntax

`:View<x>:Scale:XStart <value>`

`:View<x>:Scale:XStart?`

Arguments

`<value>::=<NR1>` ranges from 0 to 64 – (the horizontal axis full-scale) channels.

Examples

`:View1:Scale:XStart 0`
sets the value represented by the left edge of the horizontal axis to 0.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis (power) full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:YScale <value>
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges 1 to 100 dB.

Examples

`:View1:Scale:YScale 50`
sets the vertical axis full-scale to 50 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis (power).

Syntax

```
:View<x>:Scale:YStart <value>
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from ref level – 200 dB to ref level + 100 dB – horizontal axis full-scale.

Examples

`:View1:Scale:YStart -50`
sets the value represented by the bottom edge of the vertical axis to –50 dB.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:TotalPower? (Query Only)

Queries the total power of the symbol.

Syntax

```
:View<x>:TotalPower?
```

Returns

<NR3>

Examples

`:View1:TotalPower?`
might return `-7.212`, indicating that the total power is –7.212 dB.

:View<x>:Version? (Query Only)

Queries the version of the CodePower View program.

Syntax

```
:View<x>:Version?
```

Returns

<NR2>

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:XAxis (?)

Specifies or queries the horizontal axis.

Syntax

```
:View<x>:XAxis { Code | Symbol }
:View<x>:XAxis?
```

Arguments

`Code` defines the horizontal axis as code.

`Symbol` defines the horizontal axis as symbol.

Examples

`:View1:XAxis ShortCode`
defines the horizontal axis as code.

Related Commands

`:View<x>:YAxis`

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the vertical axis.

Syntax

`:View<x>:YAxis { Relative | Absolute }`

`:View<x>:YAxis?`

Arguments

`Relative` represents relative channel power to the total power along the vertical axis.

`Absolute` represents absolute channel power along the vertical axis.

Examples

`:View1:YAxis Relative`
represents relative channel power to the total power along the vertical axis.

Related Commands

`:View<x>:XAxis`

:View<x>:Z (?)

Specifies or queries the displayed symbol number.

Syntax

`:View<x>:Z <value>`

`:View<x>:Z?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of symbols – 1.

Examples

`:View1:Z 199`
specifies that the symbol 199 displays.

CodeWSpectrogram View Commands

When you select `CodeWSpectrogram` in the `Config:View<x>` command, use the commands in this section to control the code-domain power spectrogram view. This view incorporates analysis functions for the W-CDMA standard.

:View<x>:Marker:Channel? (Query Only)

Queries the channel number at the marker position.

Syntax

```
:View<x>:Marker:Channel?
```

Returns

`<NR1>`

Examples

`:View1:Marker:Channel?`
might return `26` indicating the channel number.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:ResetDelta`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

Syntax `:View<x>:Marker:ResetDelta`

Arguments None

Examples `:View1:Marker:ResetDelta`

Related Commands `:View<x>:Marker:DeltaMarker`

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

Syntax `:View<x>:Marker:SearchMax`

Arguments None

Examples `:View1:Marker:SearchMax`
positions the marker on the highest signal on screen.

Related Commands `:View<x>:Marker:SearchMin`

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

Syntax `:View<x>:Marker:SearchMin`

Arguments None

Examples `:View1:Marker:SearchMin`
positions the marker on the lowest signal on screen.

Related Commands `:View<x>:Marker:SearchMax`

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

Syntax

```
:View<x>:Marker:SearchSeparation <value>
:View<x>:Marker:SearchSeparation?
```

Arguments

`<value>::=<NR2>` specifies the minimum horizontal distance to separate two peaks. The range is 1 to 10 % relative to full-scale.

Examples

`:View1:Marker:SearchSeparation 10`
sets the minimum horizontal distance for peak separation to 10 % relative to full-scale.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Marker:SymbolRate? (Query Only)

Queries the symbol rate at the marker position.

Syntax

```
:View<x>:Marker:SymbolRate?
```

Returns

`<NR1>`

Examples

`:View1:Marker:SymbolRate?`
might return `1024k`, indicating that the symbol rate is 1024 K/s.

Related Commands

`:View<x>:SymbolRate`

:View<x>:Marker:TimeSlot? (Query Only)

Queries the time slot number at the marker position.

Syntax `:View<x>:Marker:TimeSlot?`

Returns <NR1>

Examples `:View1:Marker:TimeSlot?`
might return 12, indicating that the time slot number is twelve.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

Syntax `:View<x>:Marker:ToggleDelta`

Arguments None

Examples `:View1:Marker:ToggleDelta`
changes the primary marker and delta marker positions each other.

:View<x>:Marker:TotalPower? (Query Only)

Queries the total power of the time slot at the marker position.

Syntax `:View<x>:Marker:TotalPower?`

Returns <NR3>

Examples `:View1:Marker:TotalPower?`
might return -7.212, indicating that the total power is –7.212 dB.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR1>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis (channel).

Examples

`:View1:Marker:X 32`
positions the marker at channel 32.

Related Commands

`:View<x>:Marker:Z`

:View<x>:Marker:Z (?)

Sets or queries the marker position on the vertical (time slot number) axis.

Syntax

```
:View<x>:Marker:Z <value>
```

```
:View<x>:Marker:Z?
```

Arguments

`<value>::=<NR1>` ranges 0 to the number of slots – 1.

Examples

`:View1:Marker:Z 199`
positions the marker at slot 199.

Related Commands

`:View<x>:Marker:X`

:View<x>:Monochrome (?)

Determines whether to display a spectrogram in monochrome.

Syntax

```
:View<x>:Monochrome { On | Off }
```

```
:View<x>:Monochrome?
```

Arguments

`On` displays a spectrogram in monochrome.

`Off` displays a spectrogram in color.

Examples

`:View1:Monochrome On`
displays a spectrogram in monochrome.

:View<x>:NumberColors (?)

Selects or queries the number of display colors.

Syntax

```
:View<x>:NumberColors { 10 | 100 }
```

```
:View<x>:NumberColors?
```

Arguments

10 selects the 10-color display.

100 selects the 100-color display (default).

Examples

`:View1:NumberColors 100`
selects the 100-color display.

Related Commands

`:View<x>:Scale:YScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the Y (color) axis automatically to best display the data.

Syntax `:View<x>:Scale:AutoYScale`

Arguments None

Examples `:View1:Scale:AutoYScale`
adjusts the scaling of the Y axis automatically.

Related Commands `:View<x>:Scale:YScale`

:View<x>:Scale:FullYScale (No Query Form)

Sets the Y (color) axis scale to the default full-scale.

Syntax `:View<x>:Scale:FullYScale`

Arguments None

Examples `:View1:Scale:FullYScale`
sets the Y scale to the default full-scale.

Related Commands `:View<x>:Scale:YScale`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel) full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:XScale <value>
```

```
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges 16 to 1024 channels.

Examples

`:View1:Scale:XScale 64`
sets the horizontal axis full-scale to 64 channels.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`, `:View<x>:Scale:ZScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel).

Syntax

```
:View<x>:Scale:XStart <value>
```

```
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from 0 to 1024 – (the horizontal axis full-scale) channels.

Examples

`:View1:Scale:XStart 0`
sets the value represented by the left edge of the horizontal axis to channel 0.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the Y (color) axis full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
```

```
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` is 10, 20, 50, or 100 dB.

Examples

`:View1:Scale:YScale 50`
sets the Y axis full-scale to 50 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`, `:View<x>:Scale:ZScale`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Y (color) axis.

Syntax

```
:View<x>:Scale:YStart <value>
```

```
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from ref level – 200 dB to ref level + 100 dB – horizontal axis full-scale.

Examples

`:View1:Scale:YStart -50`
sets the value represented by the bottom edge of the Y axis to –50 dBm.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:ZScale (?)

Sets or queries the time slot display interval. The spectrogram is displayed every specified number of slots.

Syntax

```
:View<x>:Scale:ZScale <value>
```

```
:View<x>:Scale:ZScale?
```

Arguments

`<value>::=<NR1>` ranges 1 to 32.
1 means every slot is displayed. 32 means every 32^{th} slot is displayed.

Examples

`:View1:Scale:ZScale 8`
displays the spectrogram every 8^{th} slot.

Related Commands

`:View<x>:Scale:ZStart`

:View<x>:Scale:ZStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Z (slot number) axis, i.e. the first slot to be displayed.

Syntax

```
:View<x>:Scale:ZStart <value>
```

```
:View<x>:Scale:ZStart?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of slots – 1.

Examples

`:View1:Scale:ZStart 20`
sets the value represented by the bottom edge of the Z axis to 20.

Related Commands

`:View<x>:Scale:ZScale`

:View<x>:SymbolRate (?)

Sets or queries the symbol rate to display the code-domain power.

Syntax

`:View<x>:SymbolRate <value>`

`:View<x>:SymbolRate?`

Arguments

`<value>::={ 1024k | 512k | 256k | 128k | 64k | 32k| 16k | Composite }` selects the symbol rate. Composite corresponds to multi-rate.

Examples

`:View1:SymbolRate 1024k`
sets the symbol rate to 1024K.

:View<x>:Version? (Query Only)

Queries the version of the CodeWSpectrogram View program.

Syntax

`:View<x>:Version?`

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the Y (color) axis.

Syntax

`:View<x>:YAxis { Relative | Absolute }`

`:View<x>:YAxis?`

Arguments

`Relative` represents relative channel power to the total power along the Y axis.

`Absolute` represents absolute channel power along the Y axis.

Examples

`:View1:YAxis Relative`
represents relative channel power to the total power along the Y axis.

:View<x>:ZGap (?)

Sets or queries the display interval between two adjacent time slots on the Z (slot number) axis on screen.

Syntax

```
:View<x>:ZGap <value>
```

```
:View<x>:ZGap?
```

Arguments

`<value>::=<NR1>` ranges 1 to 10 pixels.

Examples

```
:View1:ZGap 10
```

sets the display interval between two adjacent slots on the Z axis to 10 pixels.

CodeWPolar View Commands

When you select CodeWPolar in the :Config:View<x> command, use the commands in this section to control the polar view. This view incorporates analysis functions for the W-CDMA standard.

For information on the polar view, refer to the user manual.

:View<x>:AlphaBT (?)

Sets or queries the α/BT value.

Syntax

`:View<x>:AlphaBT <value>`

`:View<x>:AlphaBT?`

Arguments

`<value>::=<NR2>` ranges 0.0001 to 1.

Examples

`:View1:AlphaBT 1`
sets the α/BT value to 1.

:View<x>:Analysis:TimeSlot (?)

Specifies or queries the time slot number to display the constellation.

Syntax

`:View<x>:Analysis:TimeSlot <value>`

`:View<x>:Analysis:TimeSlot?`

Arguments

`<value>::=<NR1>` ranges 0 to the number of time slots –1.

Examples

`:View1:Analysis:TimeSlot 1`
sets the time slot number to 1.

:View<x>:Analyze (No Query Form)

Performs analysis for all time slots acquired. This command is executed in another thread. For synchronizing command execution, refer to page 5–9.

Syntax

`:View<x>:Analyze`

Arguments

None

Examples

`:View1:Analyze`
performs analysis for all slots.

Related Commands

`:View<x>:BreakAnalyze`

:View<x>:AutoCarrier (?)

Determines whether to search the carrier automatically for each time slot.

Syntax

```
:View<x>:AutoCarrier { On | Off }
:View<x>:AutoCarrier?
```

Arguments

`On` searches the carrier automatically for each slot, and displays the frequency error in reference to the center frequency on screen at `Freq Error`.

`Off` sets the carrier frequency to the value specified with the `:View<x>:Carrier` command.

Examples

`:View1:AutoCarrier On`
searches the carrier automatically and displays the frequency error.

Related Commands

`:View<x>:Carrier`

:View<x>:BreakAnalyze (No Query Form)

Breaks the analysis executed by the `:View<x>:Analyze` command.

Syntax

```
:View<x>:BreakAnalyze
```

Arguments

None

Examples

`:View1:BreakAnalyze`
breaks the analysis.

Related Commands

`:View<x>:Analyze`

:View<x>:Carrier (?)

Sets or queries the carrier frequency.

Syntax

```
:View<x>:Carrier <value>
```

```
:View<x>:Carrier?
```

Arguments

`<value>::=<NR3>` ranges 0 Hz to 3 GHz.

Examples

`:View1:Carrier 1.2G`
sets the carrier frequency to 1.2 GHz.

Related Commands

`:View<x>:AutoCarrier`

:View<x>:ChipRate (?)

Sets or queries the chip rate.

Syntax

```
:View<x>:ChipRate <value>
```

```
:View<x>:ChipRate?
```

Arguments

`<value>::=<NR3>` ranges 1/s to 30 M/s.

Examples

`:View1:ChipRate 30M`
sets the chip rate to 30 M/s.

:View<x>:Constellation:SymbolRate (?)

Sets or queries the symbol rate to display symbol constellation.

Syntax

```
:View<x>:Constellation:SymbolRate <value>
:View<x>:Constellation:SymbolRate?
```

Arguments

`<value>::={ 1024k | 512k | 256k | 128k | 64k | 32k| 16k | Composite }` selects the symbol rate. Composite corresponds to multi-rate.

Examples

`:View1:Constellation:SymbolRate 1024k`
sets the symbol rate to 1024K/s.

:View<x>:Display (?)

Selects or queries the display data source.

Syntax

```
:View<x>:Display { Measurement | Reference }
:View<x>:Display?
```

Arguments

`Measurement` displays the measurement data.

`Reference` displays the reference data. Only the symbol constellation can be displayed.

Refer to the user manual on the measurement and reference data.

Examples

`:View1:Display Measurement`
displays the measurement data.

Related Commands

`:View<x>:MeasFilter`, `:View<x>:RefFilter`

:View<x>:Format (?)

Selects or queries the waveform display format.

Syntax

```
:View<x>:Format { Vector | Constellation }
```

```
:View<x>:Format?
```

Arguments

`Vector` selects the Vector format. It displays symbol-to-symbol movements using vector.

`Constellation` selects the Constellation format. It displays only symbols.

Examples

`:View1:Format Vector`
selects the Vector display format.

:View<x>:HideLMSPart (?)

Determines whether to hide LMS in the last part of the data.

Syntax

```
:View<x>:HideLMSPart { Off | On }
```

```
:View<x>:HideLMSPart?
```

Arguments

`Off` displays (does not hide) LMS.

`On` hides (does not display) LMS.

Examples

`:View1:HideLMSPart On`
hides LMS.

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

Syntax

`:View<x>:Marker:T <value>`

`:View<x>:Marker:T?`

Arguments

`<value>::=<NR3>` ranges from 0 to the time length of the acquisition data.

Examples

`:View1:Marker:T 5.4u`
positions the marker at 5.4 μs.

:View<x>:MeasFilter (?)

Selects or queries the filter to create the measurement data.

Syntax

`:View<x>:MeasFilter { None | RootRaisedCosine }`

`:View<x>:MeasFilter?`

Arguments

`None` selects no filter.

`RootRaisedCosine` selects the root raised-cosine filter.

Examples

`:View1:MeasFilter RootRaisedCosine`
selects the root raised-cosine filter to create the measurement data.

Related Commands

`:View<x>:RefFilter`

:View<x>:Modulation (?)

Selects or queries the modulation type.

Syntax

`:View<x>:Modulation W-CDMA`

`:View<x>:Modulation?`

Arguments

`W-CDMA` selects the W-CDMA modulation (this parameter only).

Examples

`:View1:Modulation W-CDMA`
selects the W-CDMA modulation.

:View<x>:RefFilter (?)

Selects or queries the filter to create the reference data.

Syntax

`:View<x>:RefFilter { None | RaisedCosine | Gaussian }`

`:View<x>:RefFilter?`

Arguments

`None` selects no filter.

`RaisedCosine` selects the raised cosine filter.

`Gaussian` selects the Gaussian filter.

Examples

`:View1:RefFilter RaisedCosine`
selects the raised cosine filter to create the reference data.

Related Commands

`:View<x>:MeasFilter`

:View<x>:ShortCode (?)

Sets or queries the short code number to display the symbol constellation.

Syntax

```
:View<x>:ShortCode <value>
```

```
:View<x>:ShortCode?
```

Arguments

`<value>::=<NR1>` ranges 0 to 1023 of the channel number.

Examples

`:View1:ShortCode 255`
sets the short code to channel 255.

Related Commands

`:View<x>:SymbolConstellation`

:View<x>:Source (?)

Selects or queries the display data source.

Syntax

```
:View<x>:Source { None | Active | Zoom | <file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the data file as the source. The file name must be “*.IQ” (the IQ format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the source.

:View<x>:Standard:WCDMA16M (No Query Form)

Configures the modulating system according to the W-CDMA standard with the chip rate of 16 M.

Syntax `:View<x>:Standard:WCDMA16M`

Arguments None

Examples `:View1:Standard:WCDMA16M`
configures the modulating system according to the W-CDMA standard with the chip rate of 16 M.

Related Commands `:View<x>:Standard:WCDMA4M`, `:View<x>:Standard:WCDMA8M`

:View<x>:Standard:WCDMA4M (No Query Form)

Configures the modulating system according to the W-CDMA standard with the chip rate of 4 M.

Syntax `:View<x>:Standard:WCDMA4M`

Arguments None

Examples `:View1:Standard:WCDMA4M`
configures the modulating system according to the W-CDMA standard with the chip rate of 4 M.

Related Commands `:View<x>:Standard:WCDMA16M`, `:View<x>:Standard:WCDMA8M`

:View<x>:Standard:WCDMA8M (No Query Form)

Configures the modulating system according to the W-CDMA standard with the chip rate of 4 M.

Syntax `:View<x>:Standard:WCDMA8M`

Arguments None

Examples `:View1:Standard:WCDMA8M`
configures the modulating system according to the W-CDMA standard with the chip rate of 8 M.

Related Commands `:View<x>:Standard:WCDMA16M`, `:View<x>:Standard:WCDMA4M`

:View<x>:SymbolConstellation (?)

Determines whether or not to display the symbol constellation.

Syntax `:View<x>:SymbolConstellation { On | Off }`

`:View<x>:SymbolConstellation?`

Arguments `On` displays the symbol constellation.

`Off` does not display the symbol constellation.

Examples `:View1:SymbolConstellation On`
displays the symbol constellation.

Related Commands `:View<x>:ShortCode`, `:View<x>:TimeSlot`

:View<x>:TimeSlot (?)

Specifies or queries the time slot number to display the symbol constellation.

Syntax

```
:View<x>:TimeSlot <value>
```

```
:View<x>:TimeSlot?
```

Arguments `<value>::=<NR1>` ranges from 0 to the number of time slots –1.

Examples

```
:View1:TimeSlot 1
```

sets the slot number to 1.

Related Commands `:View<x>:SymbolConstellation`

:View<x>:Version? (Query Only)

Queries the version of the CodeWPolar View program.

Syntax

```
:View<x>:Version?
```

Returns `<NR2>`

Examples

```
:View1:Version?
```

might return `1.1`.

CodeWPower View Commands

When you select `CodeWPower` in the `Config:View<x>` command, use the commands in this section to set the details for the code-domain power view. This view incorporates analysis functions for the W-CDMA standard.

:View<x>:Average (?)

Determines whether or not to display the results of averaging on the analyzer screen.

Syntax

`:View<x>:Average { On | Off }`

`:View<x>:Average?`

Arguments

`On` displays the results of averaging. The averaged power is displayed for each channel or symbol with a bar graph.

`Off` hides the results of averaging.

Examples

`:View1:Average On`
displays the results of averaging.

Related Commands

`:View<x>:Average:Execute`

:View<x>:Average:AllFrames (No Query Form)

Specifies that all of the acquired time slots are used for averaging.

Syntax

`:View<x>:Average:AllFrames`

Arguments

None

Examples

`:View1:Average:AllFrames`
specifies that all of the acquired time slots are used for averaging.

Related Commands

`:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:BeginZ (?)

Specifies or queries the uppermost time slot in the average range.

Syntax

```
:View<x>:Average:BeginZ <value>
```

```
:View<x>:Average:BeginZ?
```

Arguments `<value>::=<NR1>` ranges from 0 to the number of slots – 1.

Examples `:View1:Average:BeginZ 199`
sets the uppermost slot number to 199.

Related Commands `:View<x>:Average:EndZ`

:View<x>:Average:EndZ (?)

Specifies or queries the lowermost time slot in the average range.

Syntax

```
:View<x>:Average:EndZ <value>
```

```
:View<x>:Average:EndZ?
```

Arguments `<value>::=<NR1>` ranges from 0 to the number of slots – 1.

Examples `:View1:Average:EndZ 100`
sets the lowermost slot number to 100.

Related Commands `:View<x>:Average:BeginZ`

:View<x>:Average:Execute (No Query Form)

Performs averaging for each bin for the time slots in the specified range.

Syntax `:View<x>:Average:Execute`

Arguments None

Examples `:View1:Average:Execute`
performs averaging for each bin for the time slots in the specified range.

Related Commands `:View<x>:Average:AllFrames`, `:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:MarkerToFrame (No Query Form)

Specifies that the time slots between the main marker and the delta marker are averaged.

Syntax `:View<x>:Average:MarkerToFrame`

Arguments None

Examples `:View<x>:Average:MarkerToFrame`
specifies that the time slots between the main marker and the delta marker are averaged.

Related Commands `:View<x>:Average:BeginZ`, `:View<x>:Average:EndZ`

:View<x>:Average:Times (?)

Sets or queries the number of waveform acquisitions that make up an averaged waveform.

Syntax

```
:View<x>:Average:Times <value>
```

```
:View<x>:Average:Times?
```

Arguments `<value>::=<NR1>` ranges 1 to 1,000,000.

Examples `:View1:Average:Times 1000`
specifies that an averaged waveform will show the result of combining 1000 waveforms.

Related Commands `:View<x>:Average:Type`

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode. For more information about the averaging, refer to the *User Manua*l.

Syntax

```
:View<x>:Average:Type { RMSExpo | RMS | MaxHold | MinHold }
```

```
:View<x>:Average:Type?
```

Arguments `RMSExpo` averages with the RMS (root mean squared) exponential. This mode weights older acquisition data so that they have a progressively smaller effect on the average.

`RMS` averages with the RMS (root mean squared).

`MaxHold` holds the maximum value for each data point.

`MinHold` holds the minimum value for each data point.

Examples `:View1:Average:Type RMSExpo`
averages waveform with the exponential RMS.

Related Commands `:View<x>:Average:Time`

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

Syntax `:View<x>:Line:DeltaX<n>` where n = 1 to 8

Returns `<NR3>`

Examples `:View1:Line:DeltaX2?`
might return `10M`, indicating that the differnce between the vertical line marker 2 and the vertical line marker 1 is 10 MHz when the horizontal axis represents frequency.

Related Commands `:View<x>:Line:DeltaY?`, `:View<x>:Line:X<n>`

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

Syntax `:View<x>:Line:DeltaY<n>` where n = 1 to 8

Returns `<NR3>`

Examples `:View1:Line:DeltaY2?`
might return 10, indicating that the differnce between the horizontal line marker 2 and the horizontal line marker 1 is 10 dB when the vertical axis represents amplitude.

Related Commands `:View<x>:Line:DeltaX?`, `:View<x>:Line:Y<n>`

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

Syntax
```
:View<x>:Line:X<n> <value>
```
where n = 1 to 8
```
:View<x>:Line:X<n>?
```

Arguments `<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the horizontal axis.

Examples
```
:View1:Line:X1 1G
```
positions the vertical line marker 1 at 1 GHz.

Related Commands `:View<x>:Line:DeltaX?`, `:View<x>:Line:X<n>:Visible`

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

Syntax
```
:View<x>:Line:X<n>:Visible { Off | On }
```
where n = 1 to 8
```
:View<x>:Line:X<n>:Visible?
```

Arguments `Off` does not display the vertical line marker n.

`On` displays the vertical line marker n.

Examples
```
:View1:Line:X1:Visible On
```
displays the vertical line marker 1.

Related Commands `:View<x>:Line:X<n>`

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

Syntax

`:View<x>:Line:Y<n> <value>` where n = 1 to 8

`:View<x>:Line:Y<n>?`

Arguments

`<value>::=<NR3>` ranges from the minimum (bottom) edge to the maximum (top) edge of the vertical axis.

Examples

`:View1:Line:Y1 -10`
positions the horizontal line marker 1 at –10 dBm.

Related Commands

`:View<x>:Line:DeltaY?`, `:View<x>:Line:Y<n>:Visible`

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

Syntax

`:View<x>:Line:Y<n>:Visible { Off | On }` where n = 1 to 8

`:View<x>:Line:Y<n>:Visible?`

Arguments

`Off` does not display the horizontal line marker n.

`On` displays the horizontal line marker n.

Examples

`:View1:Line:X1:Visible On`
displays the horizontal line marker 1.

Related Commands

`:View<x>:Line:Y<n>`

:View<x>:Marker:Channel? (Query Only)

Queries the channel number at the marker position.

Syntax
```
:View<x>:Marker:Channel?
```

Returns
```
<NR1>
```

Examples
```
:View1:Marker:Channel?
```
might return 26 indicating the channel number.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax
```
:View<x>:Marker:DeltaMarker { On | Off }
```
```
:View<x>:Marker:DeltaMarker?
```

Arguments

On turns the delta marker on.

Off turns the delta marker off.

Examples
```
:View1:Marker:DeltaMarker On
```
turns the delta marker on.

Related Commands
```
:View<x>:Marker:ResetDelta
```

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

Syntax

```
:View<x>:Marker:ResetDelta
```

Arguments

None

Examples

```
:View1:Marker:ResetDelta
```

moves the delta marker to the main marker position.

Related Commands

```
:View<x>:Marker:DeltaMarker
```

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

Syntax

```
:View<x>:Marker:SearchMax
```

Arguments

None

Examples

```
:View1:Marker:SearchMax
```

positions the marker on the highest signal on screen.

Related Commands

```
:View<x>:Marker:SearchMin
```

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

Syntax

```
:View<x>:Marker:SearchMin
```

Arguments

None

Examples

`:View1:Marker:SearchMin`
positions the marker on the lowest signal on screen.

Related Commands

`:View<x>:Marker:SearchMax`

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

Syntax

```
:View<x>:Marker:SearchSeparation <value>
```

Arguments

`<value>::=<NR2>` specifies the minimum horizontal distance to separate two peaks. The range is 1 to 10 % relative to full-scale.

Examples

`:View1:Marker:SearchSeparation 10`
sets the minimum horizontal distance for peak separation to 10 % relative to full-scale.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Marker:SymbolRate? (Query Only)

Queries the symbol rate at the marker position.

Syntax `:View<x>:Marker:SymbolRate?`

Returns `<NR1>`

Examples `:View1:Marker:SymbolRate?`
might return 1024k, indicating that the symbol rate is 1024 K/s.

Related Commands `:View<x>:SymbolRate`

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

Syntax `:View<x>:Marker:ToggleDelta`

Arguments None

Examples `:View1:Marker:ToggleDelta`
changes the primary marker and delta marker positions each other.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR1>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis (channel or symbol).

Examples

`:View1:Marker:X 32`
positions the marker at channel 32.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:XStart`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

Syntax

```
:View<x>:Scale:AutoYScale
```

Arguments

None

Examples

`:View1:Scale:AutoYScale`
adjusts the scaling of the vertical axis automatically.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

Syntax

`:View<x>:Scale:FullYScale`

Arguments

None

Examples

`:View1:Scale:FullYScale`
sets the vertical scale to the default full-scale.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel or symbol) full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:XScale <value>
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR1>` depends on the horizontal axis setting:

Horizontal axis	Value
Short code	16, 32, 64, 128, 256, 512, or 1024
Symbol	20, 40, 80, 160, 320, or 640

Examples

`:View1:Scale:XScale 64`
sets the horizontal axis full-scale to 64 channels.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`, `:View<x>:XAxis`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel or symbol).

Syntax

```
:View<x>:Scale:XStart <value>
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR1>` ranges from 0 to 1024 – (the horizontal axis full-scale) channels.

Examples

`:View1:Scale:XStart 0`
sets the value represented by the left edge of the horizontal axis to 0.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis (power) full-scale to display a portion of the data.

Syntax

```
:View<x>:Scale:YScale <value>
:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges 1 to 100 dB.

Examples

`:View1:Scale:YScale 50`
sets the vertical axis full-scale to 50 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis (power).

Syntax

```
:View<x>:Scale:YStart <value>
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from ref level – 200 dB to ref level + 100 dB – horizontal axis full-scale.

Examples

`:View1:Scale:YStart -50`
sets the value represented by the bottom edge of the vertical axis to –50 dB.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:ShortCode (?)

Specifies or queries the short code to display the code-domain power when the horizontal axis represents symbol.

Syntax

`:View<x>:ShortCode <value>`

`:View<x>:ShortCode?`

Arguments

`<value>::=<NR1>` ranges 0 to 1023 of the channel number.

Examples

`:View1:ShortCode 255`
sets the short code to channel 255.

Related Commands

`:View<x>:XAxis`

:View<x>:SymbolRate (?)

Sets or queries the symbol rate to display the code-domain power.

Syntax

`:View<x>:SymbolRate <value>`

`:View<x>:SymbolRate?`

Arguments

`<value>::={ 1024k | 512k | 256k | 128k | 64k | 32k| 16k | Composite }` selects the symbol rate. Composite corresponds to multi-rate.

Examples

`:View1:SymbolRate 1024k`
sets the symbol rate to 1024K.

:View<x>:TimeSlot? (Query Only)

Queries the time slot number.

Syntax
```
:View<x>:TimeSlot?
```

Returns <NR1>

Examples
```
:View1:TimeSlot?
```
might return 12, indicating that the time slot number is twelve.

:View<x>:TotalPower? (Query Only)

Queries the total power of the time slot.

Syntax
```
:View<x>:TotalPower?
```

Returns <NR3>

Examples
```
:View1:TotalPower?
```
might return -7.212, indicating that the total power is –7.212 dB.

:View<x>:Version? (Query Only)

Queries the version of the CodeWPower View program.

Syntax
```
:View<x>:Version?
```

Returns <NR2>

Examples
```
:View1:Version?
```
might return 1.1.

:View<x>:XAxis (?)

Specifies or queries the horizontal axis.

Syntax

```
:View<x>:XAxis { ShortCode | Symbol | TimeSlot }

:View<x>:XAxis?
```

Arguments

`ShortCode` defines the horizontal axis as short code.

`Symbol` defines the horizontal axis as symbol.

`TimeSlot` defines the horizontal axis as time slot.

Examples

`:View1:XAxis ShortCode`
defines the horizontal axis as short code.

Related Commands

`:View<x>:ShortCode`, `:View<x>:YAxis`

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the vertical axis.

Syntax

```
:View<x>:YAxis { Relative | Absolute }

:View<x>:YAxis?
```

Arguments

`Relative` represents relative channel power to the total power along the vertical axis.

`Absolute` represents absolute channel power along the vertical axis.

Examples

`:View1:YAxis Relative`
represents relative channel power to the total power along the vertical axis.

Related Commands

`:View<x>:XAxis`

:View<x>:Z (?)

Specifies or queries the number of the time slot to display.

Syntax

```
:View<x>:Z <value>
```

```
:View<x>:Z?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of slots – 1.

Examples

`:View1:Z 199`
specifies that the slot 199 displays.

3gppACPView Commands

When you select `3gppACPView` in the `Config:View<x>` command, use the commands in this section to set the details for the 3gppACPView.

:View<x>:Average (?)

Determines whether or not to display the results of averaging.

The command usage is the same as that of the :View<x>:Average command in the Waveform view. Refer to page 4–120.

:View<x>:Average:AllFrames (No Query Form)

Specifies that averaging is performed for data in all frames acquired.

The command usage is the same as that of the :View<x>:Average:AllFrames command in the Waveform view. Refer to page 4–120.

:View<x>:Average:BeginZ (?)

Specifies or queries the uppermost frame in the averaging range.

The command usage is the same as that of the :View<x>:Average:BeginZ command in the Waveform view. Refer to page 4–121.

:View<x>:Average:EndZ (?)

Specifies or queries the lowermost frame in the averaging range.

The command usage is the same as that of the :View<x>:Average:EndZ command in the Waveform view. Refer to page 4–121.

:View<x>:Average:Execute (No Query Form)

Starts averaging.

The command usage is the same as that of the :View<x>:Average:Execute command in the Waveform view. Refer to page 4–122.

:View<x>:Average:MarkerToFrame (No Query Form)

Specifies that the frames between the main marker and the delta marker are averaged.

The command usage is the same as that of the :View<x>:Average:MarkerTo-Frame command in the Waveform view. Refer to page 4–122.

:View<x>:Average:Reset (No Query Form)

Restarts averaging.

Syntax

```
:View<x>:Average:Reset
```

Arguments None

Examples

```
:View1:Average:Reset
```

restarts averaging.

:View<x>:Average:Times (?)

Sets or queries the number of frames that make up an averaged waveform.

The command usage is the same as that of the :View<x>:Average:Times command in the Waveform view. Refer to page 4–123.

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode. For more information about the averaging, refer to the user manual.

The command usage is the same as that of the :View<x>:Average:Type command in the Waveform view. Refer to page 4–124.

:View<x>:CopyFrom (No Query Form)

Loads the display data from the text file.

The command usage is the same as that of the :View<x>:CopyFrom command in the Waveform view. Refer to page 4–125.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

The command usage is the same as that of the :View<x>:CopyTo command in the Waveform view. Refer to page 4–125.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:Band (?)

Determines whether the band marker (the two vertical bar cursors) is displayed.

The command usage is the same as that of the :View<x>:Marker:Band command in the Waveform view. Refer to page 4–133.

:View<x>:Marker:Band:Center (?)

Sets or queries the center frequency of the band marker (the two vertical bar cursors).

The command usage is the same as that of the :View<x>:Marker:Band:Center command in the Waveform view. Refer to page 4–133.

:View<x>:Marker:Band:Left (?)

Sets or queries the frequency of the left edge of the band marker (the two vertical bar cursors).

The command usage is the same as that of the :View<x>:Marker:Band:Left command in the Waveform view. Refer to page 4–134.

:View<x>:Marker:Band:Right (?)

Sets or queries the frequency of the right edge of the band marker (the two vertical bar cursors).

The command usage is the same as that of the :View<x>:Marker:Band:Right command in the Waveform view. Refer to page 4–134.

:View<x>:Marker:Band:Width (?)

Sets or queries the bandwidth of the band marker (the two vertical bar cursors).

The command usage is the same as that of the :View<x>:Marker:Band:Width command in the Waveform view. Refer to page 4–135.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

The command usage is the same as that of the :View<x>:Marker:DeltaMarker command in the Waveform view. Refer to page 4–135.

:View<x>:Marker:DeltaX (?)

Sets or queries the horizontal position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaX command in the Waveform view. Refer to page 4–136.

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaY? command in the Waveform view. Refer to page 4–136.

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X command in the Waveform view. Refer to page 4–140.

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

The command usage is the same as that of the :View<x>:Marker:Y? command in the Waveform view. Refer to page 4–140.

:View<x>:Measurement (?)

Selects or queries the measurement item. The measurement starts with the data acquisition. Query the results with the `:View<x>:Result<y>?` command.

The command usage is the same as that of the :View<x>:Measurement command in the Waveform view. Refer to page 4–141.

:View<x>:Measurement:ACP:BW (?)

Sets or queries the bandwidth for the ACP measurement.

The command usage is the same as that of the :View<x>:Measurement:ACP:BW command in the Waveform view. Refer to page 4–142.

:View<x>:Measurement:ACP:SP (?)

Sets or queries the frequency interval between the adjacent channels for the ACP measurement.

The command usage is the same as that of the :View<x>:Measurement:ACP:SP command in the Waveform view. Refer to page 4–143.

:View<x>:Measurement:Filter (?)

Enables or disables the receive filter that extracts specific channels.

Syntax

```
:View<x>:Measurement:Filter { On | Off }
```

```
:View<x>:Measurement:Filter?
```

Arguments

`On` enables the receive filter.

`Off` disables the receive filter.

Examples

`:View1:Measurement:Filter On`
enables the receive filter.

Related Commands

`:View<x>:Measurement:Filter:Alpha`,
`:View<x>:Measurement:Filter:BW`,
`:View<x>:Measurement:Filter:SP`

:View<x>:Measurement:Filter:Alpha (?)

Sets or queries the shape factor of the receive filter. See Figure 4–11.

Syntax

```
:View<x>:Measurement:Filter:Alpha <value>
:View<x>:Measurement:Filter:Alpha?
```

Arguments `<value>::=<NR2>` sets the shape factor. The range is 0.0001 to 1.

Examples `:View1:Measurement:Filter:Alpha 1`
sets the shape factor to 1.

Related Commands `:View<x>:Measurement:Filter`, `:View<x>:Measurement:Filter:BW`, `:View<x>:Measurement:Filter:SP`

:View<x>:Measurement:Filter:BW (?)

Sets or queries the bandwidth of the receive filter. See Figure 4–11.

Syntax

```
:View<x>:Measurement:Filter:BW <value>
:View<x>:Measurement:Filter:BW?
```

Arguments `<value>::=<NR3>` sets the bandwidth. The range is 0 to full-scale.

Examples `:View1:Measurement:Filter:BW 3M`
sets the bandwidth of the receive filter to 3 MHz.

Related Commands `:View<x>:Measurement:Filter`, `:View<x>:Measurement:Filter:Alpha`, `:View<x>:Measurement:Filter:SP`

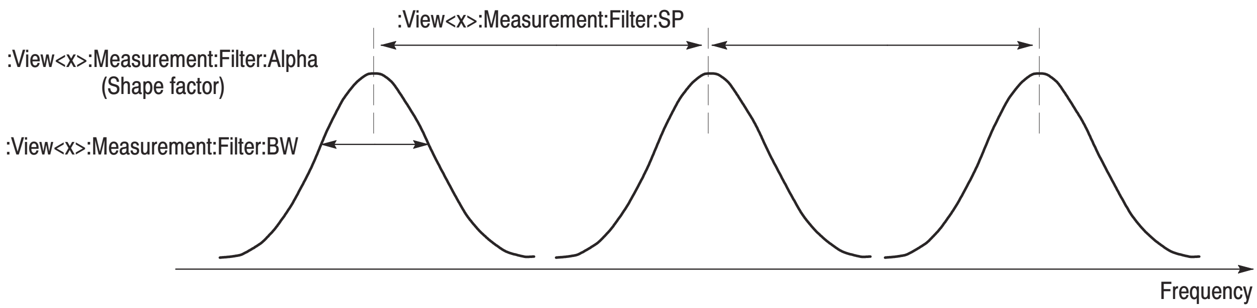


Figure 4–11: Setting the receive filter

:View<x>:Measurement:Filter:SP (?)

Sets or queries the frequency interval between adjacent receive filters. See Figure 4–11.

Syntax

`:View<x>:Measurement:Filter:SP <value>`

`:View<x>:Measurement:Filter:SP?`

Arguments

`<value>::=<NR3>` sets the bandwidth. The range is 0 to full-scale.

Examples

`:View1:Measurement:Filter:BW 3M`
sets the interval between adjacent receive filters to 3 MHz.

Related Commands

`:View<x>:Measurement:Filter`, `:View<x>:Measurement:Filter:Alpha`, `:View<x>:Measurement:Filter:BW`

:View<x>:Measurement:OBW (?)

Sets or queries the occupied bandwidth.

The command usage is the same as that of the :View<x>:Measurement:OBW command in the Waveform view. Refer to page 4–144.

:View<x>:RBW (?)

Sets or queries the resolution bandwidth of the RBW filter selected with the `:View<x>:RBW:Calculation` command.

The command usage is the same as that of the :View<x>:RBW command in the Waveform view. Refer to page 4–145.

:View<x>:RBW:Alpha (?)

Sets or queries the shape factor of the RBW filter selected with the :View<x>:RBW:Calculation command.

The command usage is the same as that of the :View<x>:RBW:Alpha command in the Waveform view. Refer to page 4–145.

:View<x>:RBW:Calculation (?)

Simulates Resolution Bandwidth (RBW) for compatibility with data measured by a conventional scanning RF spectrum analyzer. This command also selects the filter.

The command usage is the same as that of the :View<x>:RBW:Calculation command in the Waveform view. Refer to page 4–146.

:View<x>:Result<y>? (Query Only)

Queries the measurement results. Selects the measurement with the `:View<x>:Measurement` command.

Syntax

```
:View<x>:Result<y>?
```

Returns

`<NR3>`

`Result1` returns the result of the measurement specified with the `:View<x>:Measurement` command. For the ACP measurement, it returns the lower measurement result.

`Result2` returns the ACP upper measurement result.

Examples

```
:View1:Measurement OBW;Result1?
```

might return `57.36E+03`, indicating that the occupied bandwidth is 57.36 kHz.

Related Commands

`:View<x>:Measurement`

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the Waveform view. Refer to page 4–149.

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

The command usage is the same as that of the :View<x>:Scale:XStart command in the Waveform view. Refer to page 4–149.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the Waveform view. Refer to page 4–151.

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

The command usage is the same as that of the :View<x>:Scale:XStart command in the Waveform view. Refer to page 4–151.

:View<x>:Source (?)

Selects or queries the display data source for the specified view.

Syntax

```
:View<x>:Source { None | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

Examples

`:View1:Source D1`
specifies the data register D1 as the view source.

:View<x>:Version? (Query Only)

Queries the version of the 3gppACPView program.

Syntax

```
:View<x>:Version?
```

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

3gppSpectrogram View Commands

Use the 3gppSpectrogram view commands to control the code-domain power spectrogram view when you select `3gppSpectrogram` in the `Config:View<x>` command. This view incorporates analysis functions for the 3GPP standard.

:View<x>:Marker:Channel? (Query Only)

Queries the channel number at the marker position.

The command usage is the same as that of the :View<x>:Marker:Channel? command in the CodeWSpectrogram view. Refer to page 4–326.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

The command usage is the same as that of the :View<x>:Marker:DeltaMarker command in the CodeWSpectrogram view. Refer to page 4–326.

:View<x>:Marker:PSCHPower? (Query Only)

Queries the absolute power of the Primary Synchronization Channel (PSCH) of the time slot at the marker position.

Syntax `:View<x>:Marker:PSCHPower?`

Returns `<NR3>`

Examples `:View1:Marker:PSCHPower?`
might return `7.212`, indicating that the absolute power of the PSCH is 7.212 dB.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the CodeWSpectrogram view. Refer to page 4–327.

:View<x>:Marker:SCG? (Query Only)

Queries the Scrambling Code Group (SCG) of the time slot at the marker position.

Syntax `:View<x>:Marker:SCG?`

Returns `<NR1>`

Examples `:View1:Marker:SCG?`
might return 38 indicating the SCG.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the CodeWSpectrogram view. Refer to page 4–327.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the CodeWSpectrogram view. Refer to page 4–327.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the CodeWSpectrogram view. Refer to page 4–328.

:View<x>:Marker:SSCH? (Query Only)

Queries the Secondary Synchronization Channel (SSCH) of the time slot at the marker position.

Syntax `:View<x>:Marker:SSCH?`

Returns `<NR1>`

Examples `:View1:Marker:SSCH?`
might return `26` indicating the SSCH.

:View<x>:Marker:SSCHPower? (Query Only)

Queries the absolute power of the Secondary Synchronization Channel (SSCH) of the time slot at the marker position.

Syntax `:View<x>:Marker:SSCHPower?`

Returns `<NR3>`

Examples `:View1:Marker:SSCHPower?`
might return `7.212`, indicating that the absolute power of the SSCH is 7.212 dB.

:View<x>:Marker:SymbolRate? (Query Only)

Queries the symbol rate at the marker position.

The command usage is the same as that of the :View<x>:Marker:SymbolRate? command in the CodeWSpectrogram view. Refer to page 4–328.

:View<x>:Marker:TimeSlot? (Query Only)

Queries the time slot number at the marker position.

The command usage is the same as that of the :View<x>:Marker:TimeSlot? command in the CodeWSpectrogram view. Refer to page 4–329.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the CodeWSpectrogram view. Refer to page 4–329.

:View<x>:Marker:TotalPower? (Query Only)

Queries the total power of the time slot at the marker position.

The command usage is the same as that of the :View<x>:Marker:TotalPower? command in the CodeWSpectrogram view. Refer to page 4–329.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X command in the CodeWSpectrogram view. Refer to page 4–330.

:View<x>:Marker:Z (?)

Sets or queries the marker position on the vertical (time slot number) axis.

The command usage is the same as that of the :View<x>:Marker:Z command in the CodeWSpectrogram view. Refer to page 4–330.

:View<x>:Monochrome (?)

Determines whether to display a spectrogram in monochrome.

The command usage is the same as that of the :View<x>:Monochrome command in the CodeWSpectrogram view. Refer to page 4–331.

:View<x>:NumberColors (?)

Selects or queries the number of display colors.

The command usage is the same as that of the :View<x>:NumberColors command in the CodeWSpectrogram view. Refer to page 4–331.

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the Y (color) axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the CodeWSpectrogram view. Refer to page 4–332.

:View<x>:Scale:FullYScale (No Query Form)

Sets the Y (color) axis scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the CodeWSpectrogram view. Refer to page 4–332.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel) full-scale to display a portion of data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the CodeWSpectrogram view. Refer to page 4–333.

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel).

The command usage is the same as that of the :View<x>:Scale:XStart command in the CodeWSpectrogram view. Refer to page 4–333.

:View<x>:Scale:YScale (?)

Sets or queries the Y (color) axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:YScale command in the CodeWSpectrogram view. Refer to page 4–334.

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Y (color) axis.

The command usage is the same as that of the :View<x>:Scale:YStart command in the CodeWSpectrogram view. Refer to page 4–334.

:View<x>:Scale:ZScale (?)

Sets or queries the time slot display interval. The spectrogram is displayed every specified number of slots.

The command usage is the same as that of the :View<x>:Scale:ZScale command in the CodeWSpectrogram view. Refer to page 4–335.

:View<x>:Scale:ZStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the Z (slot number) axis, i.e. the first slot to be displayed.

The command usage is the same as that of the :View<x>:Scale:ZStart command in the CodeWSpectrogram view. Refer to page 4–335.

:View<x>:SymbolRate (?)

Sets or queries the symbol rate to display the code-domain power.

The command usage is the same as that of the :View<x>:SymbolRate command in the CodeWSpectrogram view. Refer to page 4–336.

:View<x>:Version? (Query Only)

Queries the version of the 3gppSpectrogram View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the Y (color) axis.

The command usage is the same as that of the :View<x>:YAxis command in the CodeWSpectrogram view. Refer to page 4–336.

:View<x>:ZGap (?)

Sets or queries the display interval between two adjacent time slots on the Z (slot number) axis on screen.

The command usage is the same as that of the :View<x>:ZGap command in the CodeWSpectrogram view. Refer to page 4–337.

3gppPolar View Commands

When you select 3gppPolar in the :Config:View<x> command, use the commands in this section to control the polar view. This view incorporates analysis functions for the 3GPP standard.

:View<x>:AlphaBT (?)

Sets or queries the α/BT value.

The command usage is the same as that of the :View<x>:AlphaBT command in the CodeWPolar view. Refer to page 4–340.

:View<x>:Analysis:TimeSlot (?)

Specifies or queries the time slot number to display the constellation.

The command usage is the same as that of the :View<x>:Analysis:TimeSlot command in the CodeWPolar view. Refer to page 4–340.

:View<x>:Analyze (No Query Form)

Performs analysis on the background for all time slots acquired.

The command usage is the same as that of the :View<x>:Analyze command in the CodeWPolar view. Refer to page 4–340.

:View<x>:AutoCarrier (?)

Determines whether to search the carrier automatically for each time slot.

The command usage is the same as that of the :View<x>:AutoCarrier command in the CodeWPolar view. Refer to page 4–341.

:View<x>:BreakAnalyze (No Query Form)

Breaks the analysis executed by the `:View<x>:Analyze` command.

The command usage is the same as that of the :View<x>:AutoCarrier command in the CodeWPolar view. Refer to page 4–341.

:View<x>:Carrier (?)

Sets or queries the carrier frequency.

The command usage is the same as that of the :View<x>:Carrier command in the CodeWPolar view. Refer to page 4–342.

:View<x>:ChipRate (?)

Sets or queries the chip rate.

The command usage is the same as that of the :View<x>:ChipRate command in the CodeWPolar view. Refer to page 4–342.

:View<x>:Constellation:SymbolRate (?)

Sets or queries the symbol rate to display symbol constellation.

Syntax

```
:View<x>:Constellation:SymbolRate <value>

:View<x>:Constellation:SymbolRate?
```

Arguments

`<value>::={ 960k | 480k | 240k | 120k | 60k | 30k| 15k | 7.5k | Composite }` selects the symbol rate. Composite corresponds to multi-rate.

Examples

`:View1:Constellation:SymbolRate 960k`
sets the symbol rate to 960K/s.

:View<x>:Display (?)

Selects or queries the display data source.

The command usage is the same as that of the :View<x>:Display command in the CodeWPolar view. Refer to page 4–343.

:View<x>:Downlink:ScramblingCode (?)

Specifies or queries the scrambling code. The specified scrambling code is effective when `:View<x>:Downlink:ScramblingCodeSearch Off` is selected.

Syntax

```
:View<x>:Downlink:ScramblingCode <value>
```

```
:View<x>:Downlink:ScramblingCode?
```

Arguments

`<value>::=<NR1>` sets the scrambling code. The range is 0 to 24575.

Examples

```
:View1:Downlink:ScramblingCode 24575
```

sets the scrambling code to 24575.

Related Commands

`:View<x>:Downlink:ScramblingCodeSearch`

:View<x>:Downlink:ScramblingCodeSearch (?)

Determines whether or not to search for the scrambling code to analyze the down-link signal.

Syntax

```
:View<x>:Downlink:ScramblingCodeSearch { On | Off }

:View<x>:Downlink:ScramblingCodeSearch?
```

Arguments

`On` searches for the scrambling code to analyze the down-link signal.

`Off` uses the scrambling code set with the `:View<x>:Downlink:ScramblingCode` command instead of searching for the scrambling code.

***NOTE**. The analyzer detects the three channels of P-SCH, S-SCH, and PCPICH to establish the synchronization and correct the frequency and phase for the 3GPP down-link signal analysis. If these channel levels are too low to be detected, the analyzer cannot make measurement correctly. This error occurs when one of these channel levels is less than about 1/10th the sum of other channel levels. In this case, select* `:View<x>:Downlink:ScramblingCodeSearch Off` *and specify the scrambling code with the* `:View<x>:Downlink:ScramblingCode` *command.*

Examples

`:View1:Downlink:ScramblingCodeSearch On`
searches for the scrambling code to analyze the down-link signal.

Related Commands

`:View<x>:Downlink:ScramblingCode`

:View<x>:Format (?)

Selects or queries the waveform display format.

The command usage is the same as that of the :View<x>:Format command in the CodeWPolar view. Refer to page 4–344.

:View<x>:HideSCHPart (?)

Determines whether to hide SCH at the head of data.

Syntax

```
:View<x>:HideSCHPart { Off | On }
```

```
:View<x>:HideSCHPart?
```

Arguments

`Off` displays (does not hide) SCH.

`On` hides (does not display) SCH.

Examples

`:View1:HideSCHPart On`
hides SCH.

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

The command usage is the same as that of the :View<x>:Marker:T command in the CodeWPolar view. Refer to page 4–345.

:View<x>:MeasFilter (?)

Selects or queries the filter to create the measurement data.

The command usage is the same as that of the :View<x>:MeasFilter command in the CodeWPolar view. Refer to page 4–345.

:View<x>:Modulation (?)

Selects or queries the modulation type.

The command usage is the same as that of the :View<x>:Modulation command in the CodeWPolar view. Refer to page 4–346.

:View<x>:RefFilter (?)

Selects or queries the filter to create the reference data.

The command usage is the same as that of the :View<x>:RefFilter command in the CodeWPolar view. Refer to page 4–346.

:View<x>:ShortCode (?)

Sets or queries the short code number to display the symbol constellation.

The command usage is the same as that of the :View<x>:ShortCode command in the CodeWPolar view. Refer to page 4–347.

:View<x>:Source (?)

Selects or queries the display data source.

The command usage is the same as that of the :View<x>:Source command in the CodeWPolar view. Refer to page 4–347.

:View<x>:Standard:WCDMA (No Query Form)

Configures the modulating system according to the W-CDMA standard with the chip rate of 3.84 M/s.

Syntax

```
:View<x>:Standard:WCDMA
```

Arguments

None

Examples

```
:View1:Standard:WCDMA
```

configures the modulating system according to the W-CDMA standard with the chip rate of 3.84 M/s.

:View<x>:SymbolConstellation (?)

Determines whether or not to display the symbol constellation.

The command usage is the same as that of the :View<x>:SymbolConstellation command in the CodeWPolar view. Refer to page 4–349.

:View<x>:TimeSlot (?)

Specifies or queries the time slot number to display the symbol constellation.

The command usage is the same as that of the :View<x>:TimeSlot command in the CodeWPolar view. Refer to page 4–350.

:View<x>:Version? (Query Only)

Queries the version of the 3gppPolar View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

3gppPower View Commands

Use the 3gppPower view commands to control the code-domain power view when you select `3gppPower` in the `Config:View<x>` command. This view incorporates analysis functions for the 3GPP standard.

:View<x>:Average (?)

Determines whether or not to display the results of averaging on the analyzer screen.

The command usage is the same as that of the :View<x>:Average command in the CodeWPower view. Refer to page 4–352.

:View<x>:Average:AllFrames (No Query Form)

Specifies that all of the acquired time slots are used for averaging.

The command usage is the same as that of the :View<x>:Average:AllFrames command in the CodeWPower view. Refer to page 4–352.

:View<x>:Average:BeginZ (?)

Specifies or queries the uppermost time slot in the average range.

The command usage is the same as that of the :View<x>:Average:BeginZ command in the CodeWPower view. Refer to page 4–353.

:View<x>:Average:EndZ (?)

Specifies or queries the lowermost time slot in the average range.

The command usage is the same as that of the :View<x>:Average:EndZ command in the CodeWPower view. Refer to page 4–353.

:View<x>:Average:Execute (No Query Form)

Performs averaging for each bin for the time slots in the specified range.

The command usage is the same as that of the :View<x>:Average:Execute command in the CodeWPower view. Refer to page 4–354.

:View<x>:Average:MarkerToFrame (No Query Form)

Specifies that the time slots between the main marker and the delta marker are averaged.

The command usage is the same as that of the :View<x>:Average:MarkerToFrame command in the CodeWPower view. Refer to page 4–354.

:View<x>:Average:Times (?)

Sets or queries the number of waveform acquisitions that make up an averaged waveform.

The command usage is the same as that of the :View<x>:Average:Times command in the CodeWPower view. Refer to page 4–355.

:View<x>:Average:Type (?)

Selects or queries the average type. You can also select the peak hold mode. For more information about the averaging, refer to the *User Manua*l.

The command usage is the same as that of the :View<x>:Average:Type command in the CodeWPower view. Refer to page 4–355.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n>? command in the CodeWPower view. Refer to page 4–356.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n>? command in the CodeWPower view. Refer to page 4–356.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the CodeWPower view. Refer to page 4–357.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the CodeWPower view. Refer to page 4–357.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the CodeWPower view. Refer to page 4–358.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the CodeWPower view. Refer to page 4–358.

:View<x>:Marker:Channel? (Query Only)

Queries the channel number at the marker position.

The command usage is the same as that of the :View<x>:Marker:Channel? command in the CodeWPower view. Refer to page 4–359.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

The command usage is the same as that of the :View<x>:Marker:DeltaMarker command in the CodeWPower view. Refer to page 4–359.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the CodeWPower view. Refer to page 4–360.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the CodeWPower view. Refer to page 4–360.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the CodeWPower view. Refer to page 4–361.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the CodeWPower view. Refer to page 4–361.

:View<x>:Marker:SymbolRate? (Query Only)

Queries the symbol rate at the marker position.

The command usage is the same as that of the :View<x>:Marker:SymbolRate? command in the CodeWPower view. Refer to page 4–362.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the CodeWPower view. Refer to page 4–362.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X command in the CodeWPower view. Refer to page 4–363.

:View<x>:PSCHPower? (Query Only)

Queries the absolute power of the Primary Synchronization Channel (PSCH) of the time slot.

Syntax `:View<x>:PSCHPower?`

Returns `<NR3>`

Examples `:View1:PSCHPower?`
might return `7.212`, indicating that the absolute power of the PSCH is 7.212 dB.

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the CodeWPower view. Refer to page 4–363.

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the CodeWPower view. Refer to page 4–364.

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (channel or symbol) full-scale to display a portion of data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the CodeWPower view. Refer to page 4–365.

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (channel or symbol).

The command usage is the same as that of the :View<x>:Scale:XStart command in the CodeWPower view. Refer to page 4–365.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis (power) full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:YScale command in the CodeWPower view. Refer to page 4–366.

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis (power).

The command usage is the same as that of the :View<x>:Scale:YStart command in the CodeWPower view. Refer to page 4–366.

:View<x>:SCG? (Query Only)

Queries the Scrambling Code Group (SCG) of the time slot.

Syntax
```
:View<x>:SCG?
```

Returns
```
<NR1>
```

Examples
```
:View1:SCG?
```
might return 38 indicating the SCG.

:View<x>:ShortCode (?)

Specifies or queries the short code to display the code-domain power when the horizontal axis represents symbol.

The command usage is the same as that of the :View<x>:ShortCode command in the CodeWPower view. Refer to page 4–367.

:View<x>:SSCH? (Query Only)

Queries the Secondary Synchronization Channel (SSCH) of the time slot.

Syntax `:View<x>:SSCH?`

Returns `<NR1>`

Examples `:View1:SSCH?`
might return `26` indicating the SSCH.

:View<x>:SSCHPower? (Query Only)

Queries the absolute power of the Secondary Synchronization Channel (SSCH) of the time slot.

Syntax `:View<x>:SSCHPower?`

Returns `<NR3>`

Examples `:View1:SSCHPower?`
might return `7.212`, indicating that the absolute power of the SSCH is 7.212 dB.

:View<x>:SymbolRate (?)

Sets or queries the symbol rate to display the code-domain power.

The command usage is the same as that of the :View<x>:SymbolRate command in the CodeWPower view. Refer to page 4–367.

:View<x>:TimeSlot? (Query Only)

Queries the time slot number.

The command usage is the same as that of the :View<x>:TimeSlot? command in the CodeWPower view. Refer to page 4–368.

:View<x>:TotalPower? (Query Only)

Queries the total power of the time slot.

The command usage is the same as that of the :View<x>:TotalPower? command in the CodeWPower view. Refer to page 4–368.

:View<x>:Version? (Query Only)

Queries the version of the 3gppPower View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

:View<x>:XAxis (?)

Specifies or queries the horizontal axis.

The command usage is the same as that of the :View<x>:XAxis command in the CodeWPower view. Refer to page 4–369.

:View<x>:YAxis (?)

Determines whether to represent relative or absolute channel power along the vertical axis.

The command usage is the same as that of the :View<x>:YAxis command in the CodeWPower view. Refer to page 4–369.

:View<x>:Z (?)

Specifies or queries the displayed time slot number.

The command usage is the same as that of the :View<x>:Z command in the CodeWPower view. Refer to page 4–370.

GSM View Commands

When you select `opGSM1` in the `:Config:Mode` command or `GSM` in the `Config:View<x>` command, use the commands in this section to set the details for the GSM view. This view incorporates analysis functions for the GSM standard. For details on GSM measurement, refer to the *User Manual*.

***NOTE**. The GSM view is designed to work only in View 4 (View D) for the GSM analysis. When you perform GSM measurement, select* opGSM1 *with the* `:Config:Mode` *command or define View 4 as* GSM.

:View<x>:Analyze (No Query Form)

Measures burst data acquired on the memory or loaded from a file. To quit the analysis, execute the `:Config:Roll` or `:Config:Block` command.

Syntax

`:View<x>:Analyze`

Arguments

None

Examples

`:View4:Analyze`
measures the data acquired on the memory or loaded from a file.

Related Commands

`:Config:Block`, `:Config:Roll`

:View<x>:Average (?)

Sets or queries the number of bursts to acquire.
Use the `:View<x>:Measure` command to start data acquisition.

Syntax

`:View<x>:Average <value>`

`:View<x>:Average?`

Arguments

`<value>::=<NR1>` ranges 1 to 4000.

Examples

`:View4:Average 20`
specifies that the analyzer acquires 20 bursts.

Related Commands

`:View<x>:Measure`

:View<x>:Burst (?)

Sets or queries the number of the burst to analyze. The measurement result is shown on screen.

Syntax

```
:View<x>:Burst <value>
:View<x>:Burst?
```

Arguments

`<value>::=<NR1>` ranges 1 to 4000.

Examples

`:View4:Burst 20`
analyzes the 20th burst and display the result on screen.

Related Commands

`:View<x>:Analyze`

:View<x>:Measure (No Query Form)

Starts to acquire and process burst data.
Use the `:Config:Roll` or `:Config:Block` command to stop data acquisition.

Syntax

```
:View<x>:Measure
```

Arguments

None

Examples

`:View4:Measure`
starts to acquire and process burst data.

Related Commands

`:Config:Block`, `:Config:Roll`

:View<x>:Script (?)

Selects or queries the GSM measurement item.
For details on the GSM measurement, refer to the *User Manual*.

Syntax

```
:View<x>:Script { Mod.Accuracy | OutputPower | PowerVSTime |
Spectrum(MOD) | Spectrum(SW) }
```

```
:View<x>:Script?
```

Arguments

`Mod.Accuracy` selects the modulation accuracy measurement.

`OutputPower` selects the mean carrier power measurement.

`PowerVSTime` selects the power vs. time measurement.

`Spectrum(MOD)` selects the ACP measurement for a continuous modulation spectrum.

`Spectrum(SW)` selects the ACP measurement for a switching transient spectrum.

Examples

`:View4:Script Mod.Accuracy`
selects the modulation accuracy measurement.

GSMPolar View Commands

When you select `opGSM1` in the `:Config:Mode` command or `GSM` in the `Config:View<x>` command, use the commands in this section to control the GSMPolar view. This view incorporates analysis functions for the GSM (Global System for Mobile Communication) standard. For details on GSM measurement, refer to the *User Manual*.

:View<x>:Burst:AnalysisLength (?)

Sets or queries the process range in bits for the measurement according to the GSM standard. See Figure 4–12.

Syntax

```
:View<x>:Burst:AnalysisLength <value>
```

```
:View<x>:Burst:AnalysisLength?
```

Arguments

`<value>::=<NR1>` ranges 1 to 8191 bits.

Examples

`:View1:Burst:AnalysisLength 8191`
sets the process range to 8192 bits for the GSM measurement.

Related Commands

`:View<x>:Burst:SyncWordLength`, `:View<x>:Burst:SyncWordPosition`

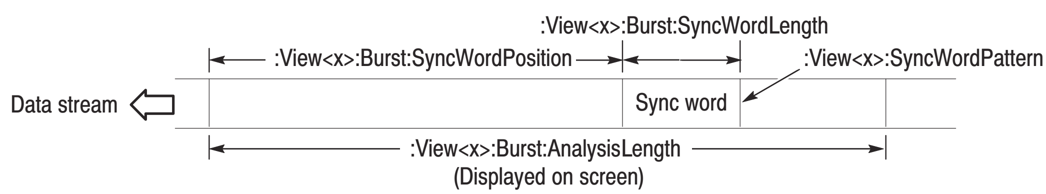


Figure 4–12: Setting the GSM measurement parameters

:View<x>:Burst:SyncWordLength (?)

Sets or queries the number of bits of the sync word for the measurement according to the GSM standard. See Figure 4–12.

This command overwrites the value set with `<length>` of `:View<x>:Burst:SyncWordPattern` command.

Syntax

```
:View<x>:Burst:SyncWordLength <value>
```

```
:View<x>:Burst:SyncWordLength?
```

Arguments

`<value>::=<NR1>` ranges 1 to 8191 bits.

Examples

`:View1:Burst:SyncWordLength 26`
sets the number of bits of the sync word to 26.

Related Commands

`:View<x>:Burst:SyncWordPattern`

:View<x>:Burst:SyncWordPattern (?)

Sets or queries the sync word for the measurement according to the GSM standard.

Syntax

```
:View<x>:Burst:SyncWordPattern { TSC0 | TSC1 | TSC2 | TSC3 | TSC4
| TSC5 | TSC6 | TSC7 | User | <pattern>,<length> }
```

```
:View<x>:Burst:SyncWordPattern?
```

Arguments

`TSC0` to `TSC7` have these hexadecimal values under the GSM standard:

TSC0: 25C225C	TSC4: 1AE41AC
TSC1: 2DDE2DC	TSC5: 4EB04E8
TSC2: 43BA438	TSC6: A7D7A7C
TSC3: 47B4478	TSC7: EF12EF0

The sync word length is set to 26 bits for TSC0 to TSC7.

`User` uses the sync word that you have set just before.

`<pattern>` sets the sync word in hexadecimal numbers.
Setting the word shows “User” in the Sync Word field of the GSMPolar view.

`<length>` sets the sync word length in bits.

- If you omit `<length>`, the value is set to 4 × (the number of characters in `<pattern>`).
- If you set the length greater than the sync word set with `<pattern>`, the remaining lower bits are set to 0.
- If you set the length less than the sync word set with `<pattern>`, the remaining lower bits are ignored.

`:View<x>:Burst:SyncWordPattern?` returns `User` if you set the sync word with `<pattern>`.

Examples

`:View1:Burst:SyncWordPattern TSC7`
sets the sync word to TSC7.

`:View1:Burst:SyncWordPattern User`
uses the sync word that you have set just before.

`:View1:Burst:SyncWordPattern 34AB14,22`
sets the sync word to 34AB14 (hexadecimal) and the word length to 22 bits.

Related Commands

`:View<x>:Burst:SyncWordLength`

:View<x>:Burst:SyncWordPosition (?)

Sets or queries the sync word position in waveform display for the measurement according to the GSM standard. See Figure 4–12 on page 4–416.

Syntax

```
:View<x>:Burst:SyncWordPosition <value>
```

```
:View<x>:Burst:SyncWordPosition?
```

Arguments

`<value>::=<NR1>` ranges 0 to 8191 bits.

Examples

`:View1:Burst:SyncWordPosition 8191`
sets the sync word position to 8191 bits.

Related Commands

`:View<x>:Burst:AnalysisLength`, `:View<x>:Burst:SyncWordLength`, `:View<x>:Burst:SyncWordPattern`

:View<x>:Marker:P? (Query Only)

Queries the phase at the marker position.

The command usage is the same as that of the :View<x>:Marker:P? command in the Polar view. Refer to page 4–214.

:View<x>:Marker:T (?)

Sets or queries the marker position on the time axis.

The command usage is the same as that of the :View<x>:Marker:T command in the Polar view. Refer to page 4–214.

:View<x>:Marker:X? (Query Only)

Queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X? command in the Polar view. Refer to page 4–215.

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

The command usage is the same as that of the :View<x>:Marker:Y? command in the Polar view. Refer to page 4–215.

:View<x>:Result<y>? (Query Only)

Queries the measurement result.

The command usage is the same as that of the :View<x>:Result<y>? command in the Polar view. Refer to page 4–223.

:View<x>:Version? (Query Only)

Queries the version of the GSMPolar View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

GSMMask View Commands

The GSMMask view is displayed on View 3 (View C) when you select `PowerVSTime` with the `:View<x>:Script` command in the the GSM view (refer to page 4–411). Use the commands in this section to set the details for the GSMMask view. This view incorporates analysis functions for the GSM standard. For details on GSM measurement, refer to the *User Manual*.

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

The command usage is the same as that of the :View<x>:CopyTo command in the Waveform view. Refer to page 4–125.

:View<x>:Line:DeltaX<n>? (Query Only)

Queries the difference between the vertical line marker n (= 1 to 8) and the vertical line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaX<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:DeltaY<n>? (Query Only)

Queries the difference between the horizontal line marker n (= 1 to 8) and the horizontal line marker 1.

The command usage is the same as that of the :View<x>:Line:DeltaY<n> command in the Waveform view. Refer to page 4–130.

:View<x>:Line:X<n> (?)

Sets or queries the position of the vertical line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:X<n> command in the Waveform view. Refer to page 4–131.

:View<x>:Line:X<n>:Visible (?)

Determines whether the vertical line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:X<n>:Visible command in the Waveform view. Refer to page 4–131.

:View<x>:Line:Y<n> (?)

Sets or queries the position of the horizontal line marker n (= 1 to 8).

The command usage is the same as that of the :View<x>:Line:Y<n> command in the Waveform view. Refer to page 4–132.

:View<x>:Line:Y<n>:Visible (?)

Determines whether the horizontal line marker n (= 1 to 8) is displayed.

The command usage is the same as that of the :View<x>:Line:Y<n>:Visible command in the Waveform view. Refer to page 4–132.

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

The command usage is the same as that of the :View<x>:Marker:DeltaMarker command in the Analog view. Refer to page 4–160.

:View<x>:Marker:DeltaX? (Query Only)

Queries the horizontal position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaX? command in the Analog view. Refer to page 4–160.

:View<x>:Marker:DeltaY? (Query Only)

Queries the vertical position of the delta marker.

The command usage is the same as that of the :View<x>:Marker:DeltaY? command in the Analog view. Refer to page 4–161.

:View<x>:Marker:Peak (No Query Form)

Moves the marker to the adjacent peak in the specified direction.

The command usage is the same as that of the :View<x>:Marker:Peak command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:ToggleDelta (No Query Form)

Changes the primary marker and delta marker positions each other.

The command usage is the same as that of the :View<x>:Marker:ToggleDelta command in the Waveform view. Refer to page 4–139.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

The command usage is the same as that of the :View<x>:Marker:X command in the Analog view. Refer to page 4–162.

:View<x>:Marker:Y? (Query Only)

Queries the vertical position of the marker.

The command usage is the same as that of the :View<x>:Marker:Y? query in the Analog view. Refer to page 4–163.

:View<x>:Result1? (Query Only)

Queries the Power vs. Time Pass/Fail test result.

Syntax `:View<x>:Result1?`

Arguments None

Returns `Pass` or `Fail`.

Examples `:View3:Result1?`
might return `Pass`, indicating that the Power vs. Time Pass/Fail test passed.

:View<x>:Scale:AutoYScale (No Query Form)

Adjusts the scaling of the vertical axis automatically to best display the data.

The command usage is the same as that of the :View<x>:Scale:AutoYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:FallingEdge (No Query Form)

Expands the falling edge of the waveform horizontally on screen.

Syntax `:View<x>:Scale:FallingEdge`

Arguments None

Examples `:View3:Scale:FallingEdge`
expands the falling edge of the waveform horizontally on screen.

Related Commands `:View<x>:Scale:RisingEdge`

:View<x>:Scale:FullYScale (No Query Form)

Sets the vertical scale to the default full-scale.

The command usage is the same as that of the :View<x>:Scale:FullYScale command in the Waveform view. Refer to page 4–147.

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:RisingEdge (No Query Form)

Expands the rising edge of the waveform horizontally on screen.

Syntax

```
:View<x>:Scale:RisingEdge
```

Arguments

None

Examples

```
:View3:Scale:RisingEdge
```

expands the rising edge of the waveform horizontally on screen.

Related Commands

```
:View<x>:Scale:FallingEdge
```

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:XScale command in the Analog view. Refer to page 4–164.

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis.

The command usage is the same as that of the :View<x>:Scale:XStart command in the Analog view. Refer to page 4–164.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis full-scale to display a portion of the data.

The command usage is the same as that of the :View<x>:Scale:YScale command in the Analog view. Refer to page 4–165.

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis.

The command usage is the same as that of the :View<x>:Scale:YStart command in the Analog view. Refer to page 4–165.

:View<x>:Version? (Query Only)

Queries the version of the GSMMask View program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

CCDF Commands

When you select CCDF in the Config:View<x> command, use the commands in this section to set details for the CCDF (Complementary Cumulative Distribution Function) measurement. The final results are displayed on CCDFView, which is controlled with the CCDFView commands (refer to page 4–441).

For details on the CCDF measurement, refer to the user manual.

:View<x>:AllFrames (No Query Form)

Specifies that CCDF is calculated for data in all frames acquired.

Syntax `:View<x>:AllFrames`

Arguments None

Examples `:View1:AllFrames`
specifies that CCDF is calculated for data in all frames.

Related Commands `:View<x>:BeginZ`, `:View<x>:EndZ`

:View<x>:Average:Reset (No Query Form)

Stops the current CCDF measurement and restarts the process.

Syntax `:View<x>:Average:Reset`

Arguments None

Examples `:View1:Average:Reset`
stops the current CCDF measurement and restarts the process.

Related Commands `:View<x>:Execute`

:View<x>:BeginZ (?)

Specifies or queries the uppermost frame in the CCDF calculation range.

Syntax

```
:View<x>:BeginZ <value>
:View<x>:BeginZ?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:BeginZ 199`
sets the uppermost frame number to 199.

Related Commands

`:View<x>:EndZ`

:View<x>:Destination (?)

Selects or queries the destination to output the CCDF calculation results.

Syntax

```
:View<x>:Destination { D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }
:View<x>:Destination?
```

Arguments

`D1` to `D8` selects one of the data registers to output the CCDF calculation results.

Examples

`:View1:Destination D1`
selects the D1 data register.

:View<x>:EndZ (?)

Specifies or queries the lowermost frame in the CCDF calculation range.

Syntax

```
:View<x>:EndZ <value>
```

```
:View<x>:EndZ?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:EndZ 100`
sets the lowermost frame number to 100.

Related Commands

`:View<x>:BeginZ`

:View<x>:Execute (No Query Form)

Performs the CCDF calculation.

Syntax

```
:View<x>:Execute
```

Arguments

None

Examples

`:View1:Execute`
performs the CCDF calculation.

Related Commands

`:View<x>:Average:Reset`

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:ResetDelta`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal (time) axis.

Examples

`:View1:Marker:X 20u`
positions the marker at –20 μs.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:XStart`, `:View<x>:Scale:XStartZero`

:View<x>:MarkerToFrame (No Query Form)

Specifies that CCDF is calculated for the range between the main marker and the delta marker.

Syntax

```
:View<x>:MarkerToFrame
```

Arguments

None

Examples

`:View1:MarkerToFrame`
specifies that CCDF is calculated for the range between the main marker and the delta marker.

Related Commands

`:View<x>:AllFrames`, `:View<x>:BeginZ`, `:View<x>:EndZ`

:View<x>:OutputFormat (?)

Selects or queries the display format of the CCDF calculation results.

Syntax

```
:View<x>:OutputFormat { CCDF | Histogram }
```

```
:View<x>:OutputFormat?
```

Arguments

`CCDF` specifies that the CCDF calculation results are displayed with time along the horizontal axis and power along the vertical axis.

`Histogram` transforms the CCDF calculation results to a histogram.

Examples

`:View1:OutputFormat CCDF`
selects the CCDF format.

:View<x>:Position (?)

Specifies or queries which frame in a block to be displayed.

The command usage is the same as that of the :View<x>:Marker:Position command in the Waveform view. Refer to page 4–144.

:View<x>:Resolution (?)

Sets or queries the amplitude resolution of a histogram when the output format is set to Histogram with the `:View<x>:OutputFormat` command.

Syntax

```
:View<x>:Resolution <value>
```

```
:View<x>:Resolution?
```

Arguments

`<value>::=<NR3>` ranges 0.01 to 10 dB.

Examples

`:View1:Resolution 10`
sets the resolution of a histogram to 10 dB.

Related Commands

`:View<x>:OutputFormat`

:View<x>:Result<y>? (Query Only)

Queries the measurement results.

Syntax
`:View<x>:Result<y>?`

Returns
`<NR3>`

`Result1` queries the crest factor (the maximum – average of amplitude).

`Result2` queries the maximum amplitude.

`Result3` queries the average amplitude.

Examples
`:View1:Result1?`
might return `11.916`, indicating that the crest factor is 11.916 dB.

`:View1:Result2?`
might return `-1.446`, indicating that the maximum amplitude is –1.446 dBm.

`:View1:Result3?`
might return `-13.363`, indicating that the average amplitude is –13.363 dBm.

:View<x>:Scale:AutoScale (No Query Form)

Resets the vertical axis (amplitude) to full-scale.

Syntax
`:View<x>:Scale:AutoScale`

Arguments
None

Examples
`:View1:Scale:AutoScale`
resets the vertical axis to full-scale.

Related Commands
`:View<x>:Scale:YScale`

:View<x>:Scale:HoldYScale (?)

Determines whether to hold or reset the vertical scale when you change the input source.

The command usage is the same as that of the :View<x>:Scale:HoldYScale command in the Waveform view. Refer to page 4–148.

:View<x>:Scale:Origin (No Query Form)

Resets the horizontal and vertical axis scale to the default setting.

Syntax

```
:View<x>:Scale:Origin
```

Arguments

None

Examples

```
:View1:Scale:Origin
```

resets the horizontal and vertical axis scale to the default setting.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YScale`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal (time) axis full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:XScale <value>
```

```
:View<x>:Scale:XScale?
```

Arguments

`<value>::=<NR3>` ranges from (time length of a frame)/256 to (time length of a frame).

Examples

```
:View1:Scale:XScale 50u
```

sets the horizontal axis full-scale to 50 μs.

Related Commands

`:View<x>:Scale:XStart`, `:View<x>:Scale:YScale`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal (time) axis.

Syntax

```
:View<x>:Scale:XStart <value>

:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum time of acquired data – the horizontal axis full-scale setting.

Examples

`:View1:Scale:XStart -50`
sets the value represented by the left edge of the horizontal axis to –50 μs.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:XStartZero`

:View<x>:Scale:XStartZero (?)

Determines whether to put the time at the left edge of each frame to zero.

The command usage is the same as that of the :View<x>:Scale:XStartZero command in the Waveform view. Refer to page 4–150.

:View<x>:Scale:YScale (?)

Sets or queries the vertical axis (amplitude) full-scale to display a portion of data.

Syntax

```
:View<x>:Scale:YScale <value>

:View<x>:Scale:YScale?
```

Arguments

`<value>::=<NR3>` ranges from full-scale/100 to full-scale of acquired data.

Examples

`:View1:Scale:YScale 100`
sets the vertical axis full-scale to 100 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:YStart`

:View<x>:Scale:YStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis (amplitude).

Syntax

```
:View<x>:Scale:YStart <value>
```

```
:View<x>:Scale:YStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum of acquired data to the maximum – full-scale setting.

Examples

`:View1:Scale:YStart -90`
sets the value represented by the bottom edge of the vertical axis to –90 dBm.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Source (?)

Selects or queries the data source for the view.

Syntax

```
:View<x>:Source { None | Active | Zoom | <file_name> }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the file as the source. The file name must be "*.IQ" (the IQ format).

Examples

`:View1:Source Active`
specifies the currently acquired data as the view source.

:View<x>:Version? (Query Only)

Queries the version of the CCDF program.

Syntax

```
:View<x>:Version?
```

Returns

`<NR2>`

Examples

`:View1:Version?`
might return `1.1`.

:View<x>:Z (?)

Specifies or queries the displayed frame number.

Syntax

```
:View<x>:Z <value>

:View<x>:Z?
```

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:View1:Z 199`
specifies that the frame 199 displays.

CCDFView Commands

When you select `CCDFView` in the `Config:View<x>` command, use the commands in this section to set details for the CCDF (Complementary Cumulative Distribution Function) view. This view displays the final results of CCDF measurement, which is controlled with the CCDF commands (refer to page 4–429).

For details on the CCDF measurement, refer to the user manual.

:View<x>:CopyFrom (No Query Form)

Copies display data from a data register or text file to the data register specified with the `:View<x>:Source` command.

Syntax

```
:View<x>:CopyFrom { D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
<file_name> }
```

Arguments

`D1` to `D8` selects one of the data registers.

`<file_name>::=<string>` is "*.TXT" (ASCII text file).

The register or file is the one to which the display data has been stored with the `:View<x>:CopyTo` command. The data of the register or file is copied to the register specified with the `:View<x>:Source` command.

Examples

`:View1:CopyFrom D2`
copies display data from the data register D2.

Related Commands

`:View<x>:CopyTo`, `:View<x>:Source`

:View<x>:CopyTo (No Query Form)

Stores the display data to a file or data register. This command is effective only for the data acquired in the Vector mode.

Syntax

```
:View<x>:CopyTo { Clipboard | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
D8 | <file_name> }
```

Arguments

`Clipboard` copies the display data to the Windows clipboard in the ASCII text format. You can use the data in your application.

`D1` to `D8` are the data register 1 to 8, respectively.

`<file_name>::=<string>` is "*.TXT" (ASCII text file). The extension is set to ".TXT" automatically.

Examples

`:View1:CopyTo Clipboard`
stores the display data to the clipboard.

Related Commands

`:View<x>:CopyFrom`

:View<x>:Marker:DeltaMarker (?)

Turns the delta marker on or off.

Syntax

```
:View<x>:Marker:DeltaMarker { On | Off }
```

```
:View<x>:Marker:DeltaMarker?
```

Arguments

`On` turns the delta marker on.

`Off` turns the delta marker off.

Examples

`:View1:Marker:DeltaMarker On`
turns the delta marker on.

Related Commands

`:View<x>:Marker:ResetDelta`

:View<x>:Marker:ResetDelta (No Query Form)

Moves the delta marker to the main marker position.

The command usage is the same as that of the :View<x>:Marker:ResetDelta command in the Waveform view. Refer to page 4–137.

:View<x>:Marker:SearchMax (No Query Form)

Positions the marker on the highest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMax command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchMin (No Query Form)

Positions the marker on the lowest signal on screen.

The command usage is the same as that of the :View<x>:Marker:SearchMin command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:SearchSeparation (?)

Sets or queries the minimum horizontal distance to separate two peaks.

The command usage is the same as that of the :View<x>:Marker:SearchSeparation command in the Waveform view. Refer to page 4–138.

:View<x>:Marker:X (?)

Sets or queries the horizontal position of the marker.

Syntax

```
:View<x>:Marker:X <value>
```

```
:View<x>:Marker:X?
```

Arguments

`<value>::=<NR3>` ranges from the minimum (left) edge to the maximum (right) edge of the the horizontal axis (amplitude).

Examples

`:View1:Marker:X 5`
positions the marker at 5 dB.

Related Commands

`:View<x>:Scale:XScale`, `:View<x>:Scale:XStart`

:View<x>:Scale:AutoScale (No Query Form)

Adjusts scaling of the vertical axis automatically to best display the data.

Syntax

```
:View<x>:Scale:AutoScale
```

Arguments

None

Examples

`:View1:Scale:AutoScale`
adjusts scaling of the vertical axis automatically to best display the data.

Related Commands

`:View<x>:Scale:YScale`

:View<x>:Scale:LYStart (?)

Sets or queries the value represented by the minimum (bottom) edge of the vertical axis (CCDF).

Syntax

```
:View<x>:Scale:LYStart <value>
```

```
:View<x>:Scale:LYStart?
```

Arguments

`<value>::=<NR3>` ranges $1/10^8$ to 100 %.

The value must be smaller than the one set with the `:View<x>:Scale:LYStop` command.

Examples

`:View1:Scale:LYStart 1m`
sets the value represented by the bottom edge of the vertical axis to 1/1000 %.

Related Commands

`:View<x>:Scale:LYStop`

:View<x>:Scale:LYStop (?)

Sets or queries the value represented by the maximum (top) edge of the vertical axis (CCDF).

Syntax

```
:View<x>:Scale:LYStop <value>
```

```
:View<x>:Scale:LYStop?
```

Arguments

`<value>::=<NR3>` ranges $1/10^8$ to 100 %.

The value must be greater than the one set with the `:View<x>:Scale:LYStart` command.

Examples

`:View1:Scale:LYStop 100`
sets the value represented by the top edge of the vertical axis to 100 %.

Related Commands

`:View<x>:Scale:LYStart`

:View<x>:Scale:Origin (No Query Form)

Resets the horizontal and vertical scale to the default setting.

Syntax
```
:View<x>:Scale:Origin
```

Arguments None

Examples
```
:View1:Scale:Origin
```
resets the horizontal and vertical scale to the default setting.

Related Commands `:View<x>:Scale:LYStart`, `:View<x>:Scale:LYStop`, `:View<x>:Scale:XScale`

:View<x>:Scale:XScale (?)

Sets or queries the horizontal axis (amplitude) full-scale to display a portion of data.

Syntax
```
:View<x>:Scale:XScale <value>
```
```
:View<x>:Scale:XScale?
```

Arguments `<value>::=<NR3>` ranges full-scale/256 to full-scale of the acquired data.

Examples
```
:View1:Scale:XScale 10
```
sets the horizontal axis full-scale to 10 dB.

Related Commands `:View<x>:Scale:LYStart`, `:View<x>:Scale:LYStop`, `:View<x>:Scale:XStart`

:View<x>:Scale:XStart (?)

Sets or queries the value represented by the minimum (left) edge of the horizontal axis (amplitude).

Syntax

```
:View<x>:Scale:XStart <value>
```

```
:View<x>:Scale:XStart?
```

Arguments

`<value>::=<NR3>` ranges from the minimum to the maximum amplitude of acquired data – the horizontal axis full-scale setting.

Examples

`:View1:Scale:XStart 0`
sets the value represented by the left edge of the horizontal axis to 0 dB.

Related Commands

`:View<x>:Scale:XScale`

:View<x>:Source (?)

Selects or queries the data source for the view.

Syntax

```
:View<x>:Source { None | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }
```

```
:View<x>:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

Examples

`:View1:Source D1`
specifies the data register D1 as the view source.

:View<x>:Version? (Query Only)

Queries the version of the CCDFView program.

Syntax `:View<x>:Version?`

Returns `<NR2>`

Examples `:View1:Version?`
might return `1.1`.

Self Gain-Calibration Commands

The `Util1` command group calibrates the amplifier gain based on an internal signal generator. This routine should be run when you boot the analyzer or when UNCAL (uncalibrated) is displayed during operation.

Allow the analyzer to warm up for 20 minutes before you begin the calibration procedure. The warm-up period allows electrical performance of the analyzer to stabilize.

During normal operation, when the ambient temperature changes by more than ±5° C from the temperature at the previous calibration, UNCAL is displayed in red in the hardware status display area. Run the self gain-calibration.

***NOTE**. When you run self gain-calibration during signal acquisition, calibration begins after the acquisition is completed.*

:Util1:Gain:Auto (No Query Form)

Determines whether to perform the self gain-calibration automatically when the analyzer is in uncal state. The calibration starts after data acquisition completes.

Syntax `:Util1:Gain:Auto`

Arguments None

Examples `:Util1:Gain:Auto`
executes the self gain-calibration automatically after data acquisition when the analyzer is in uncal state.

Related Commands `:Util1:Gain:Execute`

:Util1:Gain:Execute (No Query Form)

Executes the self gain-calibration.

Syntax `:Util1:Gain:Execute`

Arguments None

Examples `:Util1:Gain:Execute`
executes the self gain-calibration.

Related Commands `:Util1:Result1?`

:Util1:IQOffset:Execute (No Query Form)

Compensates the offset of the I and Q input signals.

***NOTE**. Set the level of the I and Q input signals to zero before executing the command.*

Syntax `:Util1:IQOffset:Execute`

Arguments None

Examples `:Util1:IQOffset:Execute`
compensates the offset of the I and Q input signals.

:Util1:Result1? (Query Only)

Queries the calibration result.

Syntax `:Util1:Result1?`

Returns `Result1::={ Pass | Fail }` indicates the result of the self gain-calibration.

Examples `:Util1:Result1?`
might return `Pass`, indicating the self gain-calibration completes successfully.

Related Commands `:Util1:Gain:Execute`

:Util1:Version? (Query Only)

Queries the version of the self gain-calibration program.

Syntax `:Util1:Version?`

Returns `<NR2>`

Examples `:Util1:Version?`
might return `1.1`.

:Util1:WideIQBalance:Execute (No Query Form)

Balances the DC components of I and Q signals in the Wide IF mode.

Syntax `:Util1:WideIQBalance:Execute`

Arguments None

Examples `:Util1:WideIQBalance:Execute`
balances the DC components of I and Q signals in the Wide IF mode.

:Util1:WideOffset:Request (No Query Form)

Compensates the offset in the Wide IF mode at the next data acquisition. The analyzer turns off the signal input automatically.

Syntax `:Util1:WideOffset:Request`

Arguments None

Examples `:Util1:WideOffset:Request`
compensates the offset in the Wide IF mode at the next data acquisition.

Save/Load Commands

The `Util2` command group allows you to save/load acquisition data to/from the hard disk or floppy disk.

Remember that the data file is composed of the following parts:

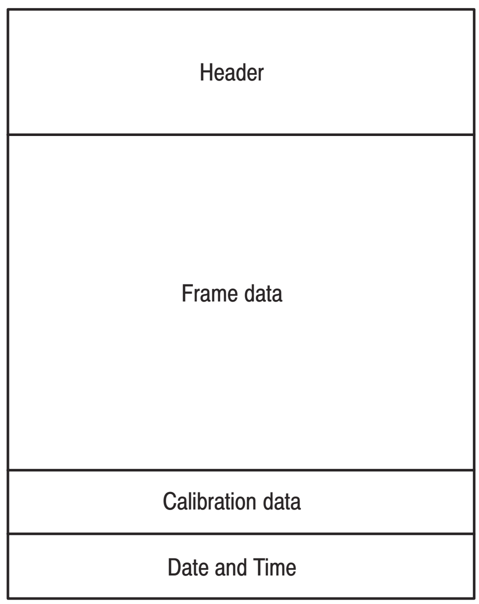


Figure 4–13: Data file structure

:Util2:AllFrames (No Query Form)

Specifies that acquisition data in all frames are saved. Use the `:Util2:Data:SaveAP` or `:Util2:Data:SaveIQ` command to save the data.

Syntax `:Util2:AllFrames`

Arguments None

Examples `:Util2:AllFrames`
specifies that acquisition data in all frames are saved.

Related Commands `:Util2:BeginZ`, `:Util2:Data:SaveAP`, `:Util2:Data:SaveIQ`, `:Util2:EndZ`

:Util2:BeginZ (?)

Specifies or queries the first frame in the save range. This setting is used by the `:Util2:Data:SaveAP` and `:Util2:Data:SaveIQ` commands to save data. Use the `:Util2:EndZ` command to specify the last frame.

Syntax `:Util2:BeginZ <value>`

`:Util2:BeginZ?`

Arguments `<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples `:Util2:BeginZ 199`
specifies that the save starts from the frame 199.

Related Commands `:Util2:Data:SaveAP`, `:Util2:Data:SaveIQ`, `:Util2:EndZ`

:Util2:Buffer:<header> (?)

Sets or queries the file header in the buffer area read from the data source with the `:Util2:Buffer:CopyHeader` command. You can save the header with the `:Util2:Buffer:SaveHeader` command.

Syntax

```
:Util2:Buffer:<header> <value>
:Util2:Buffer:<header>?
```

Arguments The following list shows the `<header>` items, `<value>`, and their meanings.

<header>	<value>	Meaning
Bins	<NRf>	The number of bins
BlockSize	<NRf>	Block size
CenterFrequency	<NRf>	Center frequency
DateTime	<NRf>	Date and time
FFTPoints	<NRf>	The number of FFT points
FFTWindow	<string>	FFT window type
FrameLength	<NRf>	Frame length
FramePadding	<string>	The position of an invalid frame: Before or After
FramePeriod	<NRf>	Frame period
FrameReverse	<string>	Frame sequence: On (reverse) or Off (normal)
FrequencyOffset	<NRf>	Frequency offset
GainOffset	<NRf>	Gain offset
LevelOffset	<NRf>	Level offset
MaxInputLevel	<NRf>	Maximum input level
MultiAddr	<NRf>	The last frame address in the multi-frame mode
MultiFrames	<NRf>	The number of frames in the multi-frame mode
Span	<NRf>	Span
Type	<string>	File type: IQ or AP format
UnitPeriod	<NRf>	Unit frame update period
ValidFrames	<NRf>	The number of all frames in the file

Examples `:Util2:Buffer:MultiFrames 2`
sets the number of frames in the multi-frame mode to two in the file header.

Related Commands `:Util2:Buffer:CopyHeader`, `:Util2:Buffer:SaveHeader`

:Util2:Buffer:CopyHeader (No Query Form)

Copies the file header from the specified data source to the buffer area. You can set the header with the `:Util2:Buffer:<header>` command, and save the header with the `:Util2:Buffer:SaveHeader` command.

Syntax

`:Util2:Buffer:CopyHeader <source>,<beginZ>,<endZ>`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames – 1.

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1, or –1. –1 represents variable record length.

Examples

`:Util2:Buffer:CopyHeader Active,199,0`
copies the file header from the active memory for the frame 199 to 0.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Util2:Buffer:<header>`, `:Util2:Buffer:SaveHeader`

:Util2:Buffer:SaveHeader (No Query Form)

Saves the file header from the buffer memory to a file. You can load the header data with the `:Util2:Buffer:CopyHeader` command to the buffer, and set the header with the `:Util2:Buffer:<header>` command.

Syntax

`:Util2:Buffer:SaveHeader <source>,<destination>`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source for the header.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<destination>::=<file_name>`

`<file_name>::=<string>` specifies the file to save the header data. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util2:Buffer:SaveHeader Active,"SAMPLE1.IQ"`
saves the header for data on active memory to the file SAMPLE1.IQ.

Related Commands

`:Util2:Buffer:<header>`, `:Util2:Buffer:CopyHeader`

:Util2:Data:Load (No Query Form)

Loads the data from the file in the IQ format to the active memory.

Syntax

`:Util2:Data:Load <file_name>`

Arguments

`<file_name>::=<string>` must be "*.IQ".

Examples

`:Util2:Data:Load "SAMPLE1.IQ"`
loads the data from the file SAMPLE1.IQ to the active memory.

Related Commands

`:Util2:Data:Save`

:Util2:Data:Save (No Query Form)

Saves the data from the specified data source to a file. All of the data including header, frame data, calibration data, date and time are saved.

Syntax

`:Util2:Data:Save <source>,<beginZ>,<endZ>,<destination>`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames – 1.

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1.

`<destination>::=<string>` specifies the file to save the data. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util2:Data:Save Active,199,0,"SAMPLE1.IQ"`
saves the data from the active memory for the frame 199 to 0 to the file SAMPLE1.IQ.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Util2:Data:Load`

:Util2:Data:SaveAP (No Query Form)

Saves data to a file in the AP format. All of the data including header, frame data, calibration data, date and time are saved. You can specify the data source with the `:Util2:Source` command, and the frame range with the `:Util2:ALLFrames` or the `:Util2:BeginZ` and `:Util2:EndZ` commands. The data saved in the AP format is unavailable for modulation analysis or zooming.

Syntax `:Util2:Data:SaveAP <file_name>`

Arguments `<file_name>::=<string>` must be “*.AP”.

Examples `:Util2:Data:SaveAP "SAMPLE1.AP"`
saves the data to the file SAMPLE1.AP.

Related Commands `:Util2:ALLFrames`, `:Util2:BeginZ`, `:Util2:EndZ`, `:Util2:Source`, `:Util2:Data:Save`, `:Util2:Data:SaveIQ`

:Util2:Data:Saved (No Query Form)

Informs the other programs of Utility, Configuration, Setup, and View that a file has been updated.

Syntax `:Util2:Data:Saved [<file_name>]`

Arguments `<file_name>::=<string>` specifies the updated file to be informed. The file name must be “*.IQ” (the IQ format) or “*.AP” (the AP format). If you omit the argument, the analyzer uses the file name specified with other command previously.

Examples `:Util2:Data:Saved "SAMPLE1.IQ"`
informs the other programs that the file SAMPLE1.IQ is updated.

:Util2:Data:SaveDateTime (No Query Form)

Adds the date and time to the file for the specified data source. Use this command only for files with variable record length.

Syntax

`:Util2:Data:SaveDateTime [<source>,]<endZ>[,<destination>]`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source. If you omit the `<source>`, the value specified in the `:Util2:Buffer:SaveHeader` command is used.

> `Active` specifies the currently acquired data as the source.
>
> `Zoom` specifies the zoomed data as the source.
>
> `D1` to `D8` specify the data register D1 to D8, respectively.
>
> `<file_name>::=<string>` specifies the data file as the source. The file name must be “*.IQ” (the IQ format) or “*.AP” (the AP format).

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1.

`<destination>::=<string>` specifies the file to save the data. The file name must be “*.IQ” (the IQ format) or “*.AP” (the AP format). If you omit the `<destination>`, the value specified in the `:Util2:Data:SaveHeader` command is used.

Examples

`:Util2:Data:SaveDateTime Active,0,"SAMPLE1.IQ"`
adds the date and time for data on active memory to the file SAMPLE1.IQ.

Related Commands

`:Util2:Buffer:SaveHeader`, `:Util2:Data:SaveHeader`

:Util2:Data:SaveFlatness (No Query Form)

Adds the calibration data for the specified source to the file.

Syntax

`:Util2:Data:SaveFlatness [<source>],[<destination>]`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source. If you omit `<source>`, the value specified in the `:Util2:Buffer:SaveHeader` command is used.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<destination>::=<string>` specifies the file to save the data. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format). If you omit `<destination>`, the value specified in the `:Util2:Data:SaveHeader` command is used.

Examples

`:Util2:Data:SaveFlatness Active,"SAMPLE1.IQ"`
adds the calibration data for currently acquired data to the file SAMPLE1.IQ.

Related Commands

`:Util2:Buffer:SaveHeader`, `:Util2:Data:SaveHeader`

:Util2:Data:SaveFrame (No Query Form)

Adds frame data for the specified source to the file.

Syntax

```
:Util2:Data:SaveFrame [<source>,]<beginZ>,<endZ>
[,<destination>]
```

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source. If you omit the `<source>`, the value specified in the `:Util2:Buffer:SaveHeader` command is used.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames – 1.

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1.

`<destination>::=<string>` specifies the file to save the data. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format). If you omit the `<destination>`, the value specified in the `:Util2:Data:SaveHeader` command is used.

Examples

`:Util2:Data:SaveFrame Active,199,0,"SAMPLE1.IQ"`
saves the frame 199 to 0 from the active memory to the file SAMPLE1.IQ.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Util2:Buffer:SaveHeader`, `:Util2:Data:SaveHeader`

:Util2:Data:SaveHeader (No Query Form)

Saves the header to a file for the specified data source.

Syntax

`:Util2:Data:SaveHeader <source>,<beginZ>,<endZ>,<destination>`

Arguments

`<source>::={ Active | Zoom | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | <file_name> }` specifies the data source.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames –1.

`<endZ>::=<NR1>` specifies the last frame. It is 0 to the number of frames – 1, or –1. –1 represents variable record length.

`<destination>::=<string>` specifies the file to save the header. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util2:Data:SaveHeader Active,199,0,"SAMPLE1.IQ"`
saves the header for the frame 199 to 0 on the active memory to the file SAMPLE1.IQ.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Util2:Data:SaveFrame`, `:Util2:Data:SaveFlatness`, `:Util2:Data:SaveDateTime`

:Util2:Data:SaveIQ (No Query Form)

Saves data to a file in the IQ format. All of the data including header, frame data, calibration data, date and time are saved. You can specify the data source with the `:Util2:Source` command, and the frame range with the `:Util2:ALLFrames` or the `:Util2:BeginZ` and `:Util2:EndZ` commands.

Syntax

`:Util2:Data:SaveIQ <file_name>`

Arguments

`<file_name>::=<string>` must be "*.IQ".

Examples

`:Util2:Data:SaveIQ "SAMPLE1.IQ"`
saves the data to the file SAMPLE1.IQ.

Related Commands

`:Util2:ALLFrames`, `:Util2:BeginZ`, `:Util2:EndZ`, `:Util2:Source`, `:Util2:Data:SaveAP`, `:Util2:Data:Save`

:Util2:EndZ (?)

Specifies or queries the last frame in the save range. This setting is used by the `:Util2:Data:SaveAP` and `:Util2:Data:SaveIQ` commands to save data. Use the `:Util2:BeginZ` command to specify the first frame.

Syntax

`:Util2:EndZ <value>`

`:Util2:EndZ?`

Arguments

`<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples

`:Util2:EndZ 100`
sets the number of the last frame to 100.

Related Commands

`:Util2:BeginZ`, `:Util2:Data:SaveAP`, `:Util2:Data:SaveIQ`

:Util2:MarkerToFrame (No Query Form)

Specifies that the frames between the main marker and the delta marker are saved. This setting is used by the `:Util2:Data:SaveAP` and `:Util2:Data:SaveIQ` commands to save data.

Syntax

`:Util2:MarkerToFrame`

Arguments

None

Examples

`:Util2:MarkerToFrame`
specifies that the frames between the main marker and the delta marker are saved.

Related Commands

`:Util2:BeginZ`, `:Util2:EndZ`, `:Util2:Data:SaveAP`, `:Util2:Data:SaveIQ`

:Util2:Source (?)

Specifies or queries the data to be saved. Use the `:Util2:Data:SaveAP` or `:Util2:Data:SaveIQ` command to save the data.

Syntax

```
:Util2:Source { None | Active | Zoom | D1 | D2 | D3 | D4 | D5 |
D6 | D7 | D8 | <file_name> }
```

`:Util2:Source?`

Arguments

`None` specifies no source (no operation).

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util2:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Util2:Data:SaveAP`, `:Util2:Data:SaveIQ`

:Util2:Version? (Query Only)

Queries the version of the Save/Load program.

Syntax

`:Util2:Version?`

Returns

`<NR2>`

Examples

`:Util2:Version?`
might return `1.1`.

Average Commands

The `Util3` command group controls averaging for the existing in-memory or -file data in the specified range.

:Util3:AllFrames (No Query Form)

Specifies that data in all frames are processed.

Syntax `:Util3:AllFrames`

Arguments None

Examples `:Util3:AllFrames`
specifies that data in all frames are processed.

Related Commands `:Util3:RMS`, `:Util3:PeakHold`

:Util3:BeginZ (?)

Sets or queries the first frame in the process range.

Syntax `:Util3:BeginZ <value>`

`:Util3:BeginZ?`

Arguments `<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples `:Util2:BeginZ 199`
sets the first frame number to 199.

Related Commands `:Util3:EndZ`, `:Util3:RMS`, `:Util3:PeakHold`

:Util3:Destination (?)

Selects or queries the destination to which the process result is output.

Syntax `:Util3:Destination { D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }`

`:Util3:Destination?`

Arguments `D1` to `D8` selects the data register D1 to D8, respectively.

The default is D1.

Examples `:Util3:Destination D1`
selects the D1 register as the destination.

Related Commands `:Util3:RMS`, `:Util3:PeakHold`

:Util3:EndZ (?)

Sets or queries the last frame in the process range.

Syntax `:Util3:EndZ <value>`

`:Util3:EndZ?`

Arguments `<value>::=<NR1>` ranges 0 to the number of frames – 1.

Examples `:Util3:EndZ 100`
sets the last frame number to 100.

Related Commands `:Util3:BeginZ`, `:Util3:RMS`, `:Util3:PeakHold`

:Util3:MarkerToFrame (No Query Form)

Specifies that the frames between the main marker and the delta marker are processed.

Syntax `:Util3:MarkerToFrame`

Arguments None

Examples `:Util3:MarkerToFrame`
specifies that the frames between the main marker and the delta marker are processed.

Related Commands `:Util3:BeginZ`, `:Util3:EndZ`, `:Util3:RMS`, `:Util3:PeakHold`

:Util3:PeakHold (No Query Form)

Holds peak for each bin in the specified frame range.

Syntax

```
:Util3:PeakHold [[&lt;source&gt;],[&lt;beginZ&gt;],[&lt;endZ&gt;],[&lt;destination&gt;]]
```

Arguments

`<source>::={ Active | Zoom | <file_name> }` specifies the data source. If you specify no argument, the value set with the `:Util3:Source` command is used.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames – 1. If you specify no argument, the value set with the `:Util3:BeginZ` command is used.

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1. If you specify no argument, the value set with the `:Util3:EndZ` command is used.

`<destination>::={ D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }` specifies the register to store the results. If you specify no argument, the value set with the `:Util3:Destination` command is used.

Examples

`:Util3:PeakHold Active,199,0,D1`
holds peak for each bin in the frame 199 to 0, and stores the results in the register D1.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Config:Mode`, `:Util3:BeginZ`, `:Util3:EndZ`, `:Util3:Destination`, `:Util3:Source`

:Util3:RMS (No Query Form)

Calculates RMS (root mean square) for each bin in the specified frame range.

Syntax

`:Util3:RMS [[<source>],[<beginZ>],[<endZ>],[<destination>]]`

Arguments

`<source>::={ Active | Zoom | <file_name> }` specifies the data source. If you specify no argument, the value set with the `:Util3:Source` command is used.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

`<beginZ>::=<NR1>` specifies the first frame. It ranges 0 to the number of frames – 1. If you specify no argument, the value set with the `:Util3:BeginZ` command is used.

`<endZ>::=<NR1>` specifies the last frame. It ranges 0 to the number of frames – 1. If you specify no argument, the value set with the `:Util3:EndZ` command is used.

`<destination>::={ D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }` specifies the register to store the results. If you specify no argument, the value set with the `:Util3:Destination` command is used.

Examples

`:Util3:RMS Active,199,0,D1`
calculates RMS for each bin in the frame 199 to 0, and stores the results in the register D1.

The larger frame number represents the older frame. However, you can specify the frame range by changing `<beginZ>` and `<endZ>`. In this example, both "`199,0`" and "`0,199`" are possible.

Related Commands

`:Config:Mode`, `:Util3:BeginZ`, `:Util3:EndZ`, `:Util3:Destination`, `:Util3:Source`

:Util3:Source (?)

Specifies or queries the data source for averaging or peak hold.

Syntax

```
:Util3:Source { None | Active | Zoom | <file_name> }
```

```
:Util3:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util3:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Config:Mode`, `:Util3:RMS`, `:Util3:PeakHold`

:Util3:Version? (Query Only)

Queries the version of the Average program.

Syntax

```
:Util3:Version?
```

Returns

`<NR2>`

Examples

`:Util3:Version?`
might return `1.1`.

Remote Commands

Use the Remote command group to set up the GPIB and TCP/IP interfaces.

*CLS (No Query Form)

Clears the analyzer status data structures.

The *CLS command clears the following:

- the Status Byte Register (SBR)
- the Service Request Enable Register (SRER)
- the Standard Event Status Register (SESR)
- the Event Status Enable Register (ESER)
- the Error Queue

For details on these registers, refer to page 5–2.

Syntax `*CLS`

Arguments None

Related Commands *ESE, *ESR?, *SRE, *STB?

*ESE (?)

Sets and queries the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (SBR). For a more detailed discussion of the use of these registers, refer to page 5–2.

Syntax

`*ESE <NR1>`

`*ESE?`

Arguments `<NR1>` is a value in the range from 0 through 255. The binary bits of the ESER are set according to this value.

Examples

`*ESE 209`
sets the ESER to binary 11010001, which enables the PON, URQ, EXE, and OPC bits.

`*ESE?`
might return the string `*ESE 186`, showing that the ESER contains the binary value 10111010.

Related Commands *CLS, *ESR?, *SRE, *STB?

*ESR? (Query Only)

Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). For a more detailed discussion of the use of these registers, see page 5–2.

Syntax `*ESR?`

Arguments None

Examples `*ESR?`
might return the value `213`, showing that the SESR contains binary 11010101.

Related Commands *CLS, *ESE, *SRE, *STB?

*IDN? (Query Only)

Returns the analyzer identification code. This command is equivalent to the `[:Util8]:Id?` command.

Syntax `*IDN?`

Returns The instrument ID in the following format:

`SONY/Tektronix,WCAXXX,FV:<firmware_version_number>,`
`SV:<software_version_number>` where `WCAXXX` is `WCA330` or `WCA380`.

Examples `*IDN?`
might return `SONY/Tektronix,WCA380,FV:1.4,SV:2.0`

Related Commands `[:Util8]:Id?`

*LRN? (Query Only)

Returns a string listing the analyzer settings, except for configuration information for the calibration values. You can use this string to return the analyzer to the state it was in when you made the :LRN? query. This command is equivalent to the [:Util8]:Set? command.

***NOTE**. The :LRN? query always returns a string including command headers, regardless of the setting of the* [:Util8]:Header *command. This is because the returned string is intended to be sent back to the analyzer as a command string.*

Syntax

`*LRN?`

Returns

`string`

Examples

`:*LRN?`

a partial response might look like this:

```
:Config:Util1 SelfGainCal;:Config:Util2 SaveLoad;:Config:Util3
Average;:Config:Util4 None;:Config:Util5 None;:Config:Util6
None;:Config:Util7 None;:Config:Util8 Remote;
```

Related Commands

`[:Util8]:Set?`

*OPC (?)

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending operations finish. The *OPC? query places the ASCII character "1" into the output queue when all pending operations are finished. The *OPC? response is not available to read until all pending operations finish. For a complete discussion of the use of these registers and the output queue, refer to page 5–2.

Syntax

`*OPC`

`*OPC?`

Arguments

None

The *OPC command allows you to synchronize the operation of the analyzer with your application program. Synchronization methods are described on page 5–9.

*RST (No Query Form)

Returns the instrument settings to the factory defaults (see *Appendix B: Factory Initialization Settings*), but does not alter those set by the Remote commands.

Syntax

```
*RST
```

Arguments

None

*SRE (?)

(Service Request Enable) sets and queries the bits in the Service Request Enable Register (SRER). For a complete discussion of the use of these registers, refer to page 5–2.

Syntax

```
*SRE <NR1>
```

```
*SRE?
```

Arguments

`<NR1>` is a value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error.

Examples

`*SRE 48`
sets the bits in the SRER to 00110000 binary.

`*SRE?`
might return a value of 32, showing that the bits in the SRER have the binary value 00100000.

Related Commands

*CLS, *ESE, *ESR?, *STB?

*STB? (Query Only)

(Read Status Byte) query returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For a complete discussion of the use of these registers, see page 5–2.

Syntax `*STB?`

Returns `<NR1>`

Examples `*STB?`
might return the value `96`, showing that the SBR contains the binary value 01100000.

Related Commands *CLS, *ESE, *ESR?, *SRE

[:Util8]:Clipboard? (Query Only)

Queries the contents of the Windows clipboard. Use this command to read the clipboard after executing the command `:View<x>:CopyTo Clipboard`.

Syntax `:Clipboard?`

Arguments None

Returns `<NR1>`

The line feed character (LF) will be replaced by “, ” (comma and space).

Examples `:Clipboard?`
a partial response might look like this:

```
   0: 00111001 00001001 01101001 11001000,   16: 10001011
01001100 11000100 01010101,   32: 01100110 00000000 10101011
11111100,   48: 00000111 11110111 11101111 11001111
```

Related Commands `:View<x>:CopyTo Clipboard`

[:Util8]:Data:<type>? (Query Only)

Gets data of the specified type from the frame specified with the `[:Util8]:Source` and `[:Util8]:Z` commands.

Syntax

`[:Util8]:Data:<type>?`

where
`<type>::={ FreqAmpl | TimeIQ }`

Returns

The amplitude data with the following format when the type is `FreqAmpl`.

```
<Amplitude 1>,<Amplitude 2>,<Amplitude 3>, ...
```

The I and Q data with the following format when the type is `TimeIQ`.

```
<I data 1><tab><Q data 1>,<I data 2><tab><Q data 2>, ...
```

Examples

`:Data:FreqAmpl?`
might return `-100, -100, -100, ...`

`:Data:TimeIQ?`
might return `-0.5 0.4, 0.3 -0.1, ...`

Related Commands

`[:Util8]:Source`, `[:Util8]:Z`

[:Util8]:Dev<x>:<command> (No Query Form)

Sends the specified command to the GPIB device. The analyzer as the controller receives this `[:Util8]:Dev<x>:<command>` command from the PC via TCP/IP, and sends the `<command>` to the other device through GPIB interface. You can not use this command through the GPIB interface.

Syntax `[:Util8]:Dev<x>:<command> [<value>]`

where
`<x>::=<NR1>` is the GPIB address of the device.

`<command>` is the GPIB command to be sent.

Arguments `<value>` is the argument of the specified command.

Examples `:Util8:Dev1:*IDN?`
sends the command *IDN? to the GPIB device with the address 1.

[:Util8]:Error? (Query Only)

Returns an error code and message.

Syntax `[:UTIL8]:Error?`

Returns `<NR1>,<message>`

Refer to page 5–11 for the error codes and messages.

Examples `:UTIL8:Error?`
might return `0,"No error"`.

[:Util8]:Event:<message> (No Query Form)

Specifies that the analyzer generates the event with the specified message. For example, when you save a file using the Save/Load commands (refer to page 4–453) along with this command, you can determine when the process completes.

Syntax

```
[:Util8]:Event:<message>
```

where `<message>` specifies the event message.

Arguments None

Examples

```
:Util2:Data:Save Active,199,0,"SAMPLE1.IQ"
:Util8:Event:Saved
```

sends the message “Saved” to the Event port when the file save operation completes.

[:Util8]:FactoryReset (No Query Form)

Returns the instrument settings to the factory defaults (refer to *Appendix B: Factory Initialization Settings*). This command does not alter the settings with the Util8 command group.

***NOTE**. You can also use *RST of the IEEE 488.2 command instead of [:Util8]:FactoryReset.*

Syntax

```
[:UTIL8]:FactoryReset
```

Arguments None

Examples

```
:UTIL8:FactoryReset
```

returns the instrument settings to the factory defaults.

[:Util8]:Format (?)

Selects or queries the waveform display format. This setting is necessary to execute the [:Util8]:Source:<item>? command.

Syntax

`[:Util8]:Format { FreqAmpl | FreqPhase | FreqI | FreqQ | TimeAmpl | TimePhase | TimeI | TimeQ }`

`[:Util8]:Format?`

Arguments

Defines the parameters associated with the horizontal and vertical axes:

Argument	Horizontal axis	Vertical axis
FreqAmpl	Frequency (span)	Amplitude
FreqPhase	Frequency (span)	Phase
FreqI	Frequency (span)	I (In–Phase)
FreqQ	Frequency (span)	Q (Quadrature-Phase)
TimeAmpl	Time	Amplitude
TimePhase	Time	Phase
TimeI	Time	I (In–Phase)
TimeQ	Time	Q (Quadrature-phase)

Examples

`:View1:Format FreqAmpl`
shows the waveform with frequency along the horizontal axis and amplitude along the vertical axis.

Related Commands

`[:Util8]:Source:<item>?`

[:Util8]:GPIB:Interface (?)

Selects or queries the GPIB configuration. You can not use this command through the GPIB interface.

Syntax

`[:Util8]:GPIB:Interface { Off | Talker/Listener | Controller }`

`[:Util8]:GPIB:Interface?`

Arguments `Off` disables all communication with the controller.

`Talker/Listener` sets the analyzer to Talker and Listener.

`Controller` sets the analyzer to Controller.

Examples `:Util8:GPIB:Interface Talker/Listener`
sets the analyzer to Talker and Listener.

[:Util8]:GPIB:PrimaryAddress (?)

Sets or queries the GPIB primary address of the analyzer. You can not use this command through the GPIB interface.

Syntax `[:Util8]:GPIB:PrimaryAddress <value>`

`[:Util8]:GPIB:PrimaryAddress?`

Arguments `<value>::=<NR1>` ranges 1 to 30.

Examples `:Util8:GPIB:PrimaryAddress 1`
sets the GPIB primary address of the analyzer to 1.

[:Util8]:GPIB:RWTimeout (?)

Sets or queries the time-out for transferring data via GPIB.

Syntax `[:Util8]:GPIB:RWTimeout <value>`

`[:Util8]:GPIB:RWTimeout?`

Arguments `<value>::={ 1000 | 300 | 100 | 30 | 10 | 3 | 1 | 300m | 100m | 30m | 10m | 3m | 1m | 300u | 100u | 30u | 10u | 0 }`

Examples `:Util8:GPIB:RWTimeout 10`
sets the time-out to 10 s.

[:Util8]:Header (?)

Determines whether to include or omit headers on query responses.

Syntax

`[:Util8]:Header { On | Off }`

`[:Util8]:Header?`

Arguments

`On` specifies that the analyzer includes headers on applicable query responses. You can then use the query response as a command.

`Off` specifies that the analyzer omit headers on query responses, so that only the argument is returned.

Examples

`:Util8:Header On`
adds header to the response.

[:Util8]:Id? (Query Only)

Returns the analyzer identification code. This command is equivalent to the IEEE 488.2 command *IDN?.

Syntax

`[:Util8]:Id?`

Returns

The instrument ID in the following format:

`SONY/Tektronix,WCAXXX,FV:<firmware_version_number>,SV:` `<software_version_number>` where `WCAXXX` is `WCA330` or `WCA380`.

Examples

`:Util8:Id?`
might return `SONY/Tektronix,WCA380,FV:1.3,SV:2.0`

Related Commands

`*IDN?`

[:Util8]:Key (No Query Form)

This command is equivalent to pressing the specified front-panel button.

Syntax `[:Util8]:Key <button>`

Arguments `<button>` specifies the front-panel button. The following table lists the arguments and their corresponding buttons.

Argument	Button	Argument	Button
Roll	**ROLL**	ViewSearch	VIEW:**SRCH**
Block	**BLOCK**	View1X1 to View2X2	View 1X1 to 2X2
PrintScreen	**PRINT**	0 to 9, ., –	Numeric keys
Mode	CONFIG:**MODE**	Enter	**ENTER/dBm**
Setup	CONFIG:**SETUP**	MHz	**MHz/s**
View	CONFIG:**VIEW**	kHz	**kHz/ms**
Util	CONFIG:**UTILITY**	Hz	**Hz/μs**
SetupMain	SETUP:**MAIN**	Clear	**CLEAR**
SetupFreq	SETUP:**FREQ**	BackSpace	**BS**
SetupSpan	SETUP:**SPAN**	Up/Down	Up/Down arrow button in **ENTRY**
SetupRef	SETUP:**REF**		
ViewA to ViewD	VIEW:**A** to **D**	StepUp/StepDown	Step up/down button beside the knob
ViewMain	VIEW:**MAIN**		
ViewScale	VIEW:**SCALE**	F1 to F8	Side buttons
ViewMarker	VIEW:**MKR**		

Examples `:Util8:Key Roll`
is equivalent to pressing the front-panel **ROLL** button.

[:Util8]:NumericOutput (?)

Sets or queries the numeric format of the returned value.

Syntax

```
[:Util8]:NumericOutput { EXP | SubUnit }
```

```
[:Util8]:NumericOutput?
```

Arguments

`EXP` specifies that returned-values are expressed by the scaled explicit point format, such as 1.0E–6 and 1.0E+6.

`SubUnit` specifies that returned-values are expressed by the SI units (which conform to the Systeme International d'Unites standard), such as 1m and 1M.

For more information about the unit, refer to page 4–5.

Examples

`:NumericOutput SubUnit`
specifies that returned-values are expressed by the SI units.

[:Util8]:ReadFile (No Query Form)

Transfers a file. After executing this command, you can receive the file by the National Instruments IBRDF subroutine for example. This command is available via GPIB only.

Syntax

```
[:Util8]:ReadFile <file_name>
```

Arguments

`<file_name>::=<string>` specifies the file.

Examples

`:Util8:ReadFile "C:\TMP.IQ"`
transfers the file C:\TMP.IQ.

[:Util8]:Register:Data (No Query Form)

Writes a value to the specified location in the data register.

Syntax `[:Util8]:Register:Data <register>,<location>,<value>`

Arguments `<register>::={ D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }`
selects the data register D1 to D8, respectively.

`<location>::=<NR1>` is the location in the data register to write the value on. It ranges 0 to the number of data points on the horizontal axis – 1.

`<value>::=<NRf>` is the value to be written.

Examples `:Util8:Register:Data D1,5,0.9`
sets the sixth point in the data register D1 to 0.9.

Related Commands `[:Util8]:Register:Header`

[:Util8]:Register:Data:<register> (?)

Writes values to the specified data register.

Syntax `[:Util8]:Register:Data:<register> <data_array>`

`[:Util8]:Register:Data:<register>?`

where
`<register>::={ D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }`

Arguments `<data_array>::=<data1>,<data2>,<data3>, ...`
where `<data#>::=<NRf>`

Examples `:Util8:Register:Data:D1 1,2,3, ...`
writes the values to the data register D1.

[:Util8]:Register:Header (No Query Form)

Specifies axis scaling and labeling for the data register.

Syntax

```
[:Util8]:Register:Header <register>,<item>,<value>
```

Arguments

`<register>::={ D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 }`
selects the data register D1 to D8, respectively.

The following table lists the `<item>` and `<value>`.

<item>	<value>	Meaning
XNum	<NRf>	The number of data points on the horizontal axis. It must be less than 8142.
XStart	<NRf>	The minimum (left) edge of the horizontal axis
XScale	<NRf>	Full-scale of the horizontal axis
XUnit	<string>	The unit for the horizontal axis, e.g. Hz, s
XLeftLabel	<string>	The label displayed on the left side under the horizontal axis. “Start” displays the start value, “Center” displays the center value, otherwise displays the specified string.
XRightLabel	<string>	The label displayed on the right side under the horizontal axis. “Stop” displays the stop value, “Span” displays the span, otherwise displays the specified string.
YStart	<NRf>	The minimum (bottom) edge of the vertical axis
YScale	<NRf>	Full-scale of the vertical axis
YUnit	<string>	The unit for the vertical axis, e.g. dB, V
YMiddleUnit	<string>	The unit displayed at middle for the vertical axis, e.g. dBm
ZNum	<NRf>	The number of frames of the register: 0 or 1. 0 indicates the frame is unavailable.

Examples

`:Util8:Register:Header D1,XNum,1000`
sets the number of data points on the horizontal axis for the D1 register to 1000.

Related Commands

`[:Util8]:Register:Data`

[:Util8]:RemoteCommand:<command> (?)

This command is used to send a command from your analyzer to another analyzer on the TCP/IP network specified with the `[:Util8]:TCPIP:Port:RemoteCommand` command.

Syntax

`[:Util8]:RemoteCommand:<command> [<value>]`

`[:Util8]:RemoteCommand:<command>?`

Arguments

`<value>` specifies the argument for `<command>`.

Examples

`:Util8:RemoteCommand:*IDN`
queries the identification code of the analyzer.

Related Commands

`[:Util8]:TCPIP:Port:RemoteCommand`

[:Util8]:Set? (Query Only)

Returns a string listing the analyzer settings, except for configuration information for the calibration values. You can use this string to return the analyzer to the state it was in when you made the `:Set?` query. This command is equivalent to the IEEE 488.2 command `*LRN?`.

Syntax

`:Set?`

Arguments

None

***NOTE**. The :Set? query always returns a string including command headers, regardless of the setting of the* [:Util8]:Header *command. This is because the returned string is intended to be sent back to the analyzer as a command string.*

Examples

`:Set?`
a partial response might look like this:

```
:Config:Util1 SelfGainCal;:Config:Util2 SaveLoad;:Config:Util3
Average;:Config:Util4 None;:Config:Util5 None;:Config:Util6
None;:Config:Util7 None;:Config:Util8 Remote;
```

[:Util8]:Source (?)

Selects or queries the data source for the `[:Util8]:Source:<item>?` command.

Syntax

```
[:Util8]:Source { None | Active | Average | Zoom | D1D2 |
D3D4 | D5D6 | D7D8 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 |
<file_name> }
```

```
[:Util8]:Source?
```

Arguments

`None` specifies no source. The display area in the view will be emptied.

`Active` specifies the currently acquired data as the source.

`Average` specifies the averaged data as the source. When you select this item, you have to set the average type and the number of averages using the `:View<x>:Average:Type` and the `:View<x>:Average:Times` commands, respectively.

`Zoom` specifies the zoomed data as the source.

`D1D2` to `D7D8` specify the register pair D1D2 to D7D8 as the source, respectively.

`D1` to `D8` specify the data register D1 to D8 as the source, respectively.

`<file_name>::=<string>` specifies the data file as the source. The file name must be "*.IQ" (the IQ format) or "*.AP" (the AP format).

Examples

`:Util8:Source Active`
specifies the currently acquired data as the source.

Related Commands

`:Config:Mode`, `[:Util8]:Source:<item>?`

[:Util8]:Source:<item>? (Query Only)

Queries the settings for the data source specified with the `[:Util8]:Source` command. You need to set the display format with the `[:Util8]:Format` command before executing this command.

Syntax

```
[:Util8]:Source:<item>?
```

Returns

The following table lists the <item> and parameter type.

<item>	Returns	Meaning
Bins	<NR1>	The number of bins
BlockSize	<NRf>	Block size
CenterFrequency	<NRf>	Center frequency
DateTime	<NRf>	Date and time
FFTPoints	<NR1>	The number of FFT points
FFTWindow	<string>	FFT window type
FrameLength	<NR1>	Frame length
FramePeriod	<NRf>	Frame period
FrameReverse	<string>	Frame sequence: On (reverse) or Off (normal)
Frames	<NR1>	The number of frames
FrequencyOffset	<NRf>	Frequency offset
FrequencyDomain	On/Off	Indicates whether the frequency domain data exists or not
GainOffset	<NRf>	Gain offset
LevelOffset	<NRf>	Level offset
MaxInputLeve	<NRf>	Maximum input level
MultiAddr	<NR1>	The last frame address in the multi–frame mode
MultiFrames	<NR1>	The number of frames in the multi–frame mode
ReferenceLevel	<NRf>	Reference level
Span	<NRf>	Span
Ticks	<NR1>	The count for time-stamping
TimeDomain	On/Off	Indicates whether the time domain data exists or not
UnitPeriod	<NRf>	Unit period for time-stamping
ValidFrames	<NRf>	The number of all frames in the file
XNum	<NR1>	The number of data points on the horizontal axis
XScale	<NRf>	Full-scale of the horizontal axis
XStart	<NRf>	The start point of the horizontal axis
XUnit	<string>	The unit of data for the horizontal axis
XLeftLabel	<string>	The label displayed on the left under the horizontal axis
XRightLabel	<string>	The label displayed on the right under the horizontal axis
YScale	<NRf>	Full-scale of the vertical axis
YStart	<NRf>	The start point of the vertical axis
YUnit	<string>	The unit of data for the vertical axis
YMiddleUnit	<string>	The unit displayed in the middle of the vertical axis
ZNum	<NR1>	The number of data points on the Z (frame) axis

Examples

`:Util8:Source:ReferenceLevel?`
might return `-30`, indicating the reference level is 30 dBm.

`:Util8:Source:TimeDomain?`
might return `Off`.

`:Util8:Source:FFTWindow?`
might return `Blackman`.

Related Commands

`[:Util8]:Format`, `[:Util8]:Source`

[:Util8]:Status? (Query Only)

Queries the status of the analyzer.

Syntax

`[:Util8]:Status?`

Returns

The following table lists the responses and their meanings:

Response	Meaning
Active:Start	Data acquisition has started
Active:Stop	Data acquisition has stopped
Active:Restart	Data acquisition has restarted
Active:Block	A block data has been acquired
Active:ChangeBlock	A block data has been acquired after you changed some settings during data acquisition
Active:EndRoll	A block data has been acquired in the Roll mode
Active:ZoomStart	Zoom has started
Active:ZoomBlock	Data has been acquired for zoom
Active:ZoomStop	Zoom has stopped
Active:Quick	Data acquisition has completed in the Quick trigger mode
Active:Prepare	Data acquisition is ready
PowerOn	The analyzer is on and ready for operation

Examples

`Util8:Status?`
might return `Active:Stop`.

[:Util8]:TCPIP:NewLine (?)

Selects or queries the new-line character of query responses through TCP/IP.

Syntax

`[:Util8]:TCPIP:NewLine { CR | LF | CRLF }`

`[:Util8]:TCPIP:NewLine?`

Arguments

`CR` sets the new-line character to Carriage Return.

`LF` sets the new-line character to Line Feed.

`CRLF` sets the new-line character to Carriage Return and Line Feed.

Examples

`:Util8:TCPIP:NewLine CR`
sets the new-line character to Carriage Return.

[:Util8]:TCPIP:Port:Command (?)

Sets or queries the TCP/IP command port number. You can not use this command via TCP/IP. Refer to page 3–6 for information about the TCP/IP ports.

Syntax

`[:Util8]:TCPIP:Port:Command <value>`

`[:Util8]:TCPIP:Port:Command?`

Arguments

`<value>::=<NR1>` ranges 1024 to 32767. It must not be the same as the event port number.

Examples

`:Util8:TCPIP:Port:Command 3000`
sets the command port number to 3000.

Related Commands

`[:Util8]:TCPIP:Port:Command:Reset`, `[:Util8]:TCPIP:Port:Event`

[:Util8]:TCPIP:Port:Command:Reset (No Query Form)

Disconnects the TCP/IP command port from the network and closes command input and response output. You can not use this command via TCP/IP. Refer to page 3–6 for information about the TCP/IP ports.

Syntax `[:Util8]:TCPIP:Port:Command:Reset`

Arguments None

Examples `:Util8:TCPIP:Port:Command:Reset`
closes the TCP/IP command port.

Related Commands `[:Util8]:TCPIP:Port:Command`

[:Util8]:TCPIP:Port:Event (?)

Sets or queries the TCP/IP event port number. You can not use this command via TCP/IP. Refer to page 3–6 for information about the TCP/IP ports.

Syntax `[:Util8]:TCPIP:Port:Event <value>`

`[:Util8]:TCPIP:Port:Event?`

Arguments `<value>::=<NR1>` ranges 1024 to 32767. It must not be the same as the command port number.

Examples `:Util8:TCPIP:Port:Event 3001`
sets the TCP/IP event port number to 3001.

Related Commands `[:Util8]:TCPIP:Port:Command`

[:Util8]:TCPIP:Port:Event:Reset (No Query Form)

Disconnects the TCP/IP event port from the network, and closes event output. You can not use this command via TCP/IP. Refer to page 3–6 for information about the TCP/IP ports.

Syntax `[:Util8]:TCPIP:Port:Event:Reset`

Arguments None

Examples `:Util8:TCPIP:Port:Event:Reset`
closes the TCP/IP event port.

Related Commands `[:Util8]:TCPIP:Port:Event`

[:Util8]:TCPIP:Port:RemoteCommand (No Query Form)

Connects the specified analyzer on the TCP/IP network with the IP address and port number.

Syntax `[:Util8]:TCPIP:Port:RemoteCommand <address>,<port>`

Arguments `<address>::=<string>` specifies the TCP/IP address.
You can also use the host name from the name server, if available.

`<port>::=<NR1>` specifies the command port number, ranging from 0 to 32,767.

Examples `:Util8:TCPIP:Port:RemoteCommand "134.62.36.161",3066`
sets the IP address to 134.62.36.161 and the port number to 3066.

Related Commands `[:Util8]:TCPIP:Port:RemoteCommand:Reset`

[:Util8]:TCPIP:Port:RemoteCommand:Reset (No Query Form)

Disconnects the TCP/IP command port from the network. Refer to page 3–6 for information about the TCP/IP ports. To re-connect the TCP/IP command port, use the `[:Util8]:TCPIP:Port:RemoteCommand` command.

Syntax `[:Util8]:TCPIP:Port:RemoteCommand:Reset`

Arguments None

Examples `:Util8:TCPIP:Port:RemoteCommand:Reset`
disconnects the TCP/IP command port from the network.

Related Commands `[:Util8]:TCPIP:Port:RemoteCommand`

[:Util8]:TCPIP:RwTimeout (?)

Sets or queries the time-out for transferring data via TCP/IP.

Syntax `[:Util8]:TCPIP:RwTimeout <value>`

`[:Util8]:TCPIP:RwTimeout?`

Arguments `<value>::=<NR3>` ranges from 0 to 1000 s.

Examples `:Util8:TCPIP:RwTimeout 1000`
sets the time-out for transferring data via TCP/IP to 1000 s.

[:Util8]:Version? (Query Only)

Queries the version of the Remote program.

Syntax `[:Util8]:Version?`

Returns `<NR2>`

Examples `:Util8:Version?`
might return `1.1`.

[:Util8]:ViewName? (Query Only)

Queries the name of the Remote program.

Syntax `[:Util8]:ViewName?`

Returns `<string>::="Util<X>"`

Examples `:Util8:ViewName?`
might return `Util8`.

[:Util8]:Z (?)

Specifies or queries the frame number for the `[:Util8]:Z:<item>?` command.

Syntax `[:Util8]:Z <value>`

`[:Util8]:Z?`

Arguments `<value>::=<NR1>` ranges from 0 to the number of frames – 1.

Examples `:Util8:Z 199`
sets the frame number to 199.

Related Commands `[:Util8]:Z:<item>?`

[:Util8]:Z:<item>? (Query Only)

Queries the time stamp or the status of the frame specified with the [:Util8]:Z command.

Syntax `[:Util8]:Z:<item>?`

Returns The following table lists the <item>, the returned values, and their meanings:

<item>	Returned value	Meaning
TimeStamp	<NRf>	Time stamp for the frame
Status	NORMAL	The frame is normal.
	INVALID	The frame has no data.
	MISSFRAME	The frame was not acquired within the specified period.
	OVERLOAD	The analyzer's A/D converter overloads.
	TRIGGERED	The analyzer has been triggered.
	LASTFRAME	The frame is the last one.
ExternalSync	<NRf>	Trigger position in seconds before the last frame acquisition in the external trigger mode.

The time stamp is zero when the last frame is acquired.

Examples `:Util8:Z:TimeStamp?`
might return `-1.6e-4`, indicating that the frame was acquired 160 μs before the last frame.

`:Util8:Z:Status?`
might return `OVERLOAD,TRIGGERED`.

`:Util8:Z:ExternalSync?`
might return `-0.0046268791875`, indicating that the external trigger occurred at about 4 ms before the last frame was acquired in the analyzer memory.

Related Commands `[:Util8]:Z`

Retrieving Response Message

When a query command is sent from the external controller, the analyzer puts the response message on the output buffer. This response message cannot be retrieved unless you perform a retrieval operation through the external controller. For example, call IBRD subroutine with the National Instruments drivers.

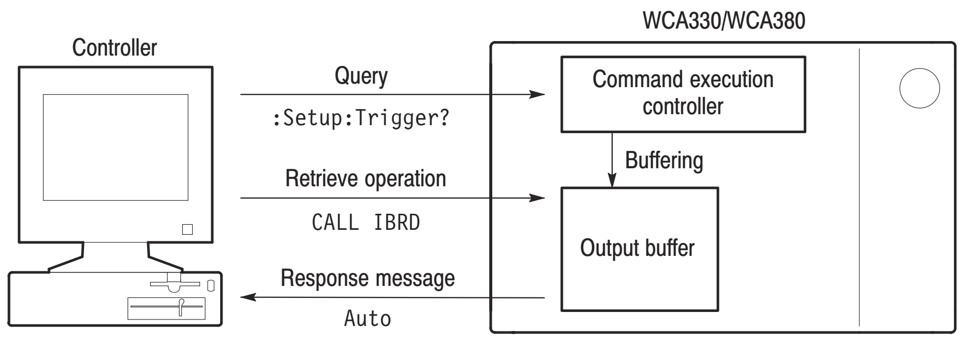


Figure 4–14: Retrieving response message

Current response message overwrites the previous response message, if any. When you send chained queries to the analyzer, such as `:Setup:Span?;ReferenceLevel?;Trigger?`, all the response messages will be written in the output buffer. These messages remain in the buffer until the next query responses overwrite them.

Status and Events

Status and Events

The analyzer provides a status and event reporting system for the GPIB and TCP/IP Ethernet interfaces. This system informs you of certain significant events that occur within the analyzer.

Obtaining Event and Error Messages

Event and error messages can be obtained by using the following queries:

- `Status?` query returns the latest event.
- `Error?` query returns the error code and message in the following format:

  `<error code>,"<error message>"`

Also, you can use the status port in TCP/IP environment. Refer to the next topic below.

Table 5–3 to 5–7 on page 5–10 to 5–12 show all event and error messages.

TCP/IP Event Port

In TCP/IP environment, the analyzer always sends the event message to the Event port. Access this port from your program to get the latest event. Refer to page 3–7 for setting the port.

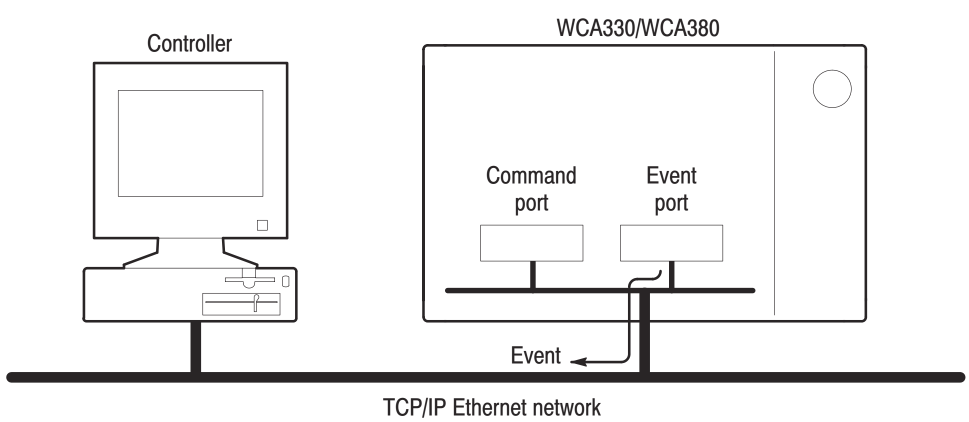


Figure 5–1: Obtaining event on the TCP/IP Ethernet

Status Reporting Structure

The analyzer status reporting function conforms to the IEEE-488.2 standard. The status reporting function is used to check for instrument errors and to identify the types of events that have occurred on the instrument.

Figure 5–2 shows an outline of the instrument's status reporting function. The status reporting function is implemented by the Standard/Event Status block. The operations processed in this block are summarized in status bytes, which provide the error and event data.

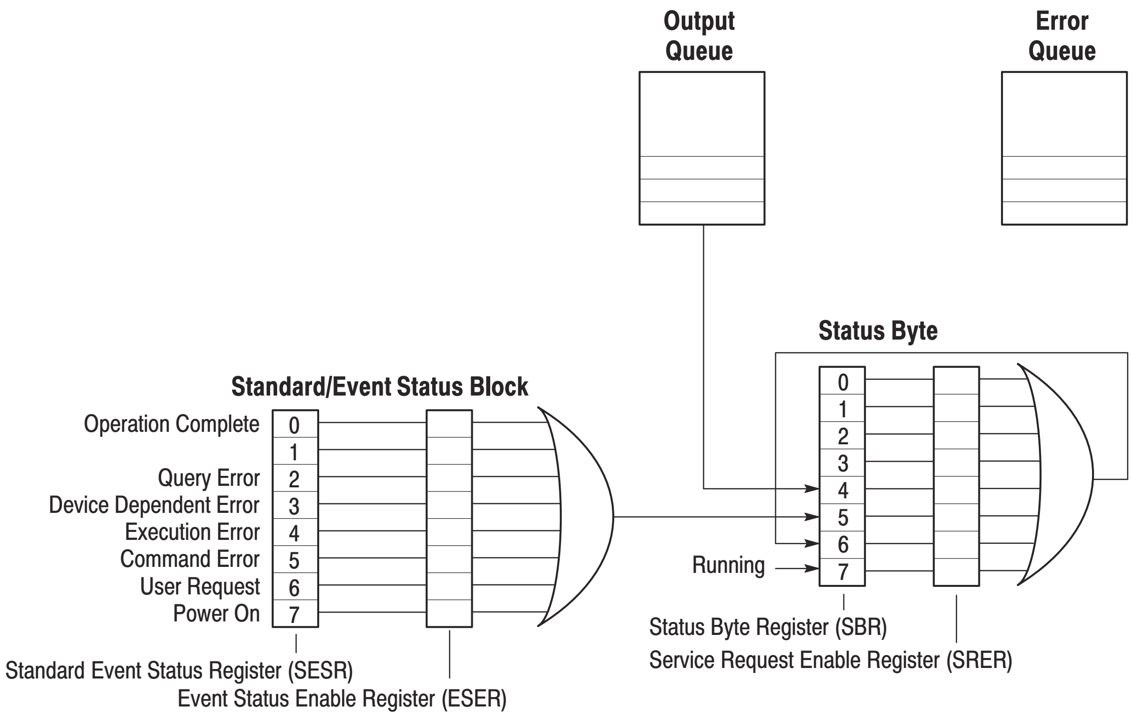


Figure 5–2: Error and Event handling process overview

Registers

There are two Status registers and two Enable registers:

- *Status Registers* store data relating to instrument status. These registers are set by the analyzer. There are two types of status registers:

  - Status Byte Register (SBR)
  - Standard Event Status Register (SESR)

  Read the contents of these registers to determine errors and conditions.

- *Enable Registers* determine whether to set events that occur in the instrument to the appropriate bits in the status registers. This type of register can be set by the user. There are two types of enable registers:

  - Event Status Enable Register (ESER)
  - Service Request Enable Register (SRER)

  Each bit in these enable registers corresponds to a bit in the controlling status register. By setting and resetting the bits in the enable register, you can determine whether or not events that occur will be registered to the status register.

Status Byte Register (SBR)

The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1987 (see Figure 5–3 and Table 5–1). These bits are used to monitor the output queue, SESR and service requests, respectively. The contents of this register are returned when the *STB? query is used.

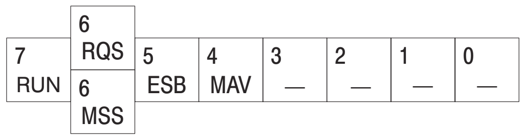


Figure 5–3: The Status Byte Register (SBR)

Table 5–1: SBR bit functions

Bit	Function
7	RUN (Running): This bit indicates that the analyzer is acquiring or zooming data. It is more effective to use the URQ bit of the Standard Event Status Register (SESR) for waiting completion of the acquisition or zoom.
6	RQS (Request Service)/MSS (Master Summary Status). When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (in other words, that the GPIB bus SRQ line is LOW). The RQS bit is cleared when serial poll ends. When the instrument is accessed using the *STB? query, this bit is called the Master Summary Status (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query.
5	ESB (Event Status Bit). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed.
4	MAV (Message Available Bit). This bit indicates that a message has been placed in the output queue and can be retrieved.
3–0	Not used

Standard Event Status Register (SESR)

The SESR is made up of 8 bits. Each bit records the occurrence of a different type of event, as shown in Figure 5–4 and Table 5–2. The contents of this register are returned when the *ESR? query is used.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	—	OPC

Figure 5–4: The Standard Event Status Register (SESR)

Table 5–2: SESR bit functions

Bit	Function
7	PON (Power On). Indicates that the power to the instrument is on.
6	URQ (User Request). This bit is set when data acquisition or zoom is completed.
5	CME (Command Error). Indicates that a command error has occurred while parsing by the command parser was in progress.
4	EXE (Execution Error). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons: ■ A value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument ■ The command could not be executed properly because the conditions for execution differed from those essentially required
3	DDE (Device-Dependent Error). An instrument error has been detected.
2	QYE (Query Error). Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons: ■ An attempt was made to retrieve messages from the output queue, despite the fact that the output queue is empty or in pending status. ■ The output queue messages have been cleared despite the fact that they have not been retrieved.
1	Not used.
0	OPC (Operation Complete). This bit is set with the results of the execution of the *OPC command. It indicates that all pending operations have been completed.

Event Status Enable Register (ESER)

The ESER is made up of bits defined exactly the same as bits 0 through 7 in the SESR (see Figure 5–5). You can use this register to designate whether the SBR ESB bit should be set when an event has occurred and to determine whether the corresponding SESR bit has been set.

To set the SBR ESB bit (when the SESR bit has been set), set the ESER bit corresponding to that event. To prevent the ESB bit from being set, reset the ESER bit corresponding to that event.

Use the *ESE command to set the bits of the ESER. Use the *ESE? query to read the contents of the ESER.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	—	OPC

Figure 5-5: The Event Status Enable Register (ESER)

Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR (see Figure 5–6). You can use this register to determine which events will generate service requests.

The SRER bit 6 cannot be set. Also, the RQS is not maskable.

The generation of a service request with the GPIB interface involves changing the SRQ line to LOW and making a service request to the controller. The result is that a status byte for which an RQS has been set is returned in response to serial polling by the controller.

Use the *SRE command to set the bits of the SRER. Use the *SRE? query to read the contents of the SRER. Bit 6 must normally be set to 0.

7	6	5	4	3	2	1	0
RUN	—	ESB	MAV	—	—	—	—

Figure 5-6: The Service Request Enable Register (SRER)

Queues

There are two types of queues in the status reporting system used in the analyzer: output queues and error queues.

Output Queue

The output queue is a FIFO (first-in, first-out) queue and holds response messages to queries, where they await retrieval. When there are messages in the queue, the SBR MAV bit is set.

The output queue is emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error occurs and the output queue is emptied; however, the operation proceeds even if an error occurs.

Error Queue

The error queue is a FIFO queue and stores events as they occur in the instrument. The oldest error code and text are retrieved with the `[:Util8]:Error?` query.

Status and Event Processing Sequence

Figure 5–7 shows an outline of the sequence for status and event processing.

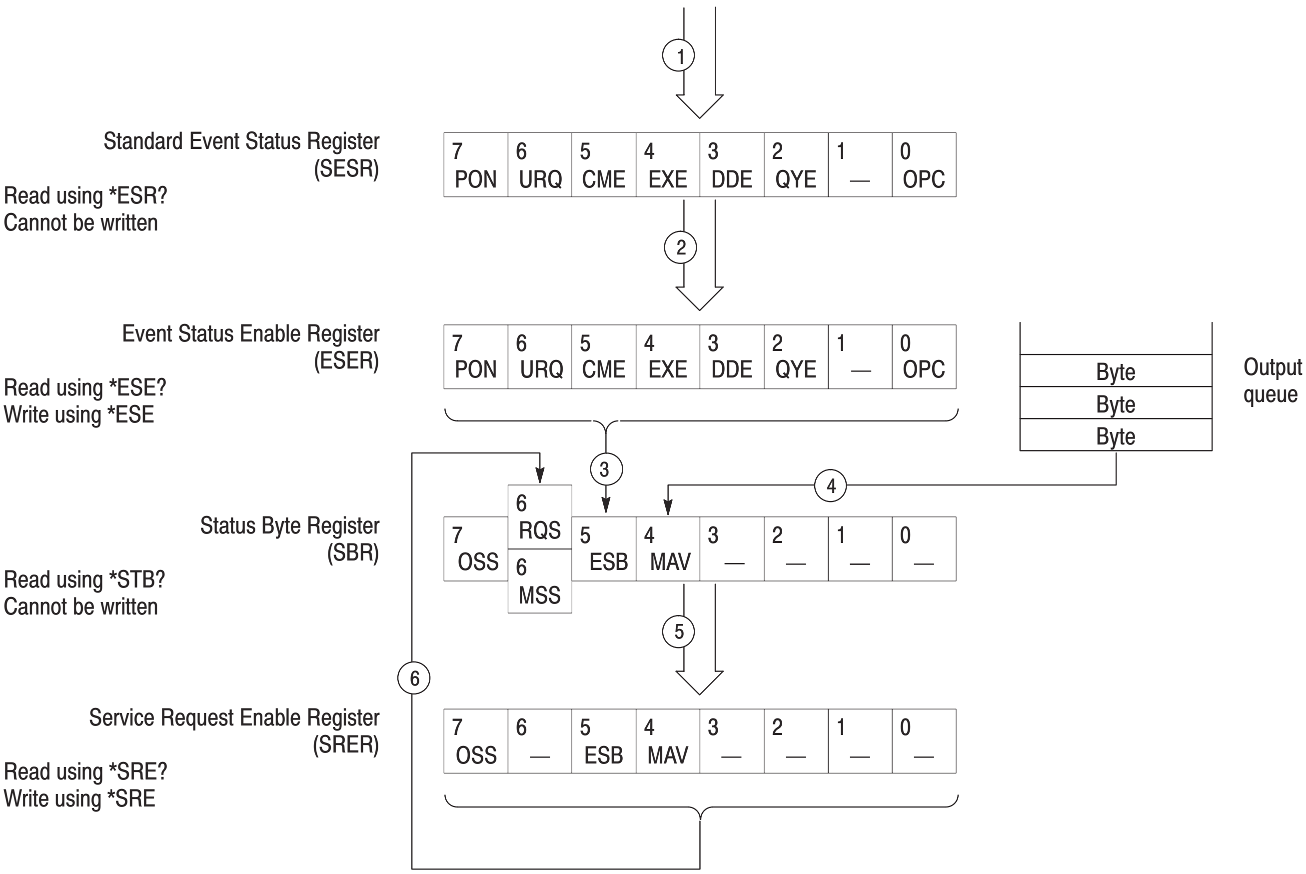


Figure 5–7: Status and Event processing sequence — Standard/Event status block

1. If an event has occurred, the SESR bit corresponding to that event is set.
2. If a bit corresponding to that event in the ESER has is set, then
3. The SBR ESB bit is set to reflect the status of the ESER.
4. When a message is sent to the output queue, the SBR MAV bit is set.
5. If a SRER bit corresponding to the SBR bit is set, then
6. The SBR MSS bit is set and a service request is generated when using the GPIB interface.

Synchronizing Execution

Almost all commands are executed in the order in which they are sent from the controller, and the execution of each command is completed in a short period of time. However, the following commands are executed in another thread, so another command can be executed simultaneously:

- `:View<x>:Measure` and `:View<x>:MeasureData` commands in the CDMATime view
- `:View<x>:Analyze` command in the 3gppPolar, CodePolar, and CodeWPolar views

These commands are designed so that the next command to be sent is executed without waiting for the previous command to be completed. In some cases, a process executed by another command must first be completed before these commands can be executed; in other cases, these commands must be completed before the next command is executed.

To synchronize execution, use the following commands:

```
*OPC
*OPC?
```

Using the *OPC Command

The `*OPC` command sets the SESR OPC bit when all pending operations have been completed. It is possible to synchronize execution by using this command together with the serial poll or service request functions when using a GPIB interface.

For example, the following command string will set the OPC bit to 1 on completion of the measurement.

```
:View1:Analyze;*OPC
```

Using the *OPC? Query

The `*OPC?` query writes an ASCII code “1” to the output queue when all pending operations have been completed. Synchronization can be performed using the following procedure:

```
:View1:Analyze;*OPC?
```

`*OPC?` waits for a “1” to be written to the output queue. In the event that the system is waiting for data to be retrieved from the output queue, a time-out may occur before the data is written to the output queue.

Messages

Tables 5–3 to 5–7 list all the programming interface messages the analyzer generates in response to commands and queries.

Event Messages

These are certain types of events that may occur while the analyzer is in use. You can get these messages with the `Status?` command. You can also access to the status port via TCP/IP to obtain events.

Table 5–3: Event message

Returned value	Meaning
PowerOn	The analyzer is on and ready for operation
Active:Prepare	The analyzer is preparing to acquire data
Active:Start	The analyzer is ready to acquire data
Active:Release	Data acquisition has started
Active:Restart	Data acquisition has restarted
Active:Block	A block of data has been acquired
Active:ChangeBlock	A block of data has been acquired after you changed some settings during the data acquisition in the Block mode
Active:EndRoll	A block of data has been acquired in the Roll mode
Active:ChangeEndRoll	A block of data has been acquired after you changed some settings during the data acquisition in the Roll mode
Active:Quick	Data acquisition has completed in the Quick trigger mode
Active:Stop	Data acquisition has stopped
Active:ZoomStart	Zoom has started
Active:ZoomBlock	Data has been acquired for zoom
Active:ZoomStop	Zoom has stopped

Error Messages

You can get these messages with the `Error?` command.

No Error. Table 5–7 shows the messages when the system has no error.

Table 5–4: No error

Code	Message
0	No error

Command Error. Command error is returned when there is a syntax error in the command.

Table 5–5: Command error

Code	Message
–100	Command error

Execution Error. These error codes and messages are returned when an error is detected while a command is being executed.

Table 5–6: Execution error

Code	Message
–200	Execution error
–220	Parameter error
–250	Mass storage error
–254	Media full
–256	File name not found
–258	Media protected

GPIB Error. When the analyzer is a controller, these error codes are returned when a GPIB error is detected.

Table 5-7: GPIB error

Code	Message
2000	System error
	Function requires GPIB board to be CIC
	Write function detected no Listeners
	Interface board not addressed correctly
	Invalid argument to function call
	Function requires GPIB board to be SAC
	I/O operation aborted
	Non-existent interface board
	Error performing DMA
	I/O operation started before previous
	(Operation completed) No capability for intended operation
	(Operation completed) File system operation error
	(Operation completed) Serial poll status byte lost
	(Operation completed) SRQ remains asserted
	(Operation completed) The return buffer is full
	(Operation completed) Address or board is locked

Programming Examples

Programming Examples

This section lists two example programs that illustrate methods you can use to control the analyzer from your application over the TCP/IP Ethernet network. These programs are written in the PERL (a freeware).

- Example 1 acquires a block of data and averages them.
- Example 2 writes peak-hold data of 30 frames to a file every 3 seconds.

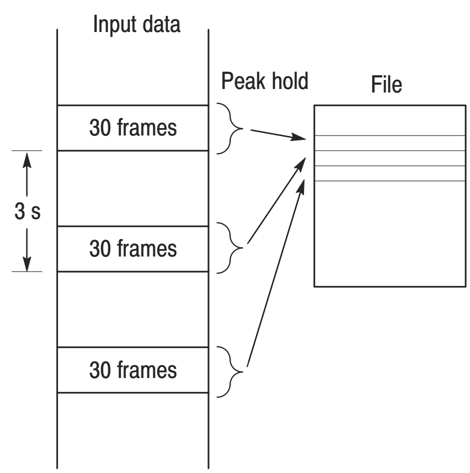
Example 1

```
#
# Acquire a block of data and average them.
#

# Set up TCP/IP. Refer to "TCP/IP Setup" on page 6-4.
use lib '(The name of the directory in which the file TCPIP.pm exists)';
use TCPIP;

# End process routine
sub signal_handler
{
   shutdown(SS, 2);
   print S "Config:Pause Off\n";
   shutdown(S, 2);
   die @_;
}
$SIG{'INT'} = 'signal_handler';
$SIG{'BREAK'} = 'signal_handler';

#
# Main
#
tcp_client(S, '', '3000');
tcp_client(SS, '', '3001');

print S "NewLine LF\n";
print S "Config:Pause On\n";
print S "Config:View1 Waveform\n";
print S "View1:Trace2:Source D1\n";
print S "Setup:Span?\n"; chomp($tmp = <S>);
   if ($tmp > 50e6) {print S "Setup:Span 50e6\n";}
print S "Setup:BlockSize 20\n";
print S "Setup:TriggerCount Off\n";

print "Press BLOCK button \n";
```

```
while (<SS>)
{
   print;
   chomp;

   if (/^Active:/)
   {
      if (/Block/) {print S "Util3:RMS Active,19,0,D1\n";}
      print S "Config:Continue\n";
   }
}
signal_handler;
```

Example 2

```
#
# Write peak-hold data of 30 frames to a file every 3 seconds.
# See the figure below.
#

# Set up TCP/IP. Refer to "TCP/IP Setup" on page 6-4.
use lib '(The name of the directory in which the file TCPIP.pm exists)';
use TCPIP;

# End process routine
sub signal_handler
{
   shutdown(SS, 2);
   print S "Config:Pause Off\n";
   shutdown(S, 2);
   die @_;
}
$SIG{'INT'} = 'signal_handler';
$SIG{'BREAK'} = 'signal_handler';
```




```
#
# Main
#
tcp_client(S, '', '3000');
tcp_client(SS, '', '3001');

print S "NewLine LF\n";
print S "Config:Pause On\n";
print S "Config:View1 Waveform\n";
print S "View1:Source Active\n";
print S "Setup:Span?\n"; chomp($tmp = <S>);
   if ($tmp > 50e6) {print S "Setup:Span 50e6\n";}
print S "Setup:BlockSize 30\n";
```

```
print S "Setup:Trigger Interval\n";
print S "Setup:TriggerCount On\n";
print S "Setup:TriggerTimes 10\n";
print S "Setup:TriggerInterval 3\n";

print "Press BLOCK button \n";

$file = 'IntBlock.ap';
$c = 0; # The number of acquisitions

while (<SS>)
{
   print;
   chomp;

   if (/^Active:/)
   {
      if (/Block/)
      {
         print S "Util3:PeakHold Active,29,0,D1\n";
# If you clear the UnitPeriod value in the header, it is
# automatically set to 100 ms by the Util2:Data:SaveHeader
# command.
         #print S "Register:Header D1,UnitPeriod,\"\"\n";
         if ($c == 0)
         {
            print S "Util2:Data:SaveHeader D1,0,-1,$file\n";
         }
         print S "Util2:Data:SaveFrame D1,0,0\n";
         $c++;
      }
      if (/Stop/)
      {
         if ($c > 0)
         {
            print S "Util2:Data:SaveFlatness D1\n";
            print S "Util2:Data:SaveDateTime D1,0\n";
            print S "Util2:Data:Saved\n";
            $c = 0;
         }
      }
      print S "Config:Continue\n";
   }
}
signal_handler;
```

TCP/IP Setup

The examples 1 and 2 above read the *.pm* file listed below (*TCPIP.pm* in these examples) to set up TCP/IP. Create this file and put it in a directory (for instance, c:\Example\Setup) and then specify the directory in the `use lib` command line at the beginning of the program as shown in Examples 1 and 2. For instance, describe `use lib 'c:\Example\Setup'`.

Example TCPIP.pm file

```
#
# TCP/IP setup program
#
package TCPIP;
use strict "subs";
use Socket;
use Exporter;
@ISA = qw(Exporter);
@EXPORT = qw(tcp_server tcp_client);

#
# Usage: tcp_server(handle[, port=3066])
#
sub tcp_server ($$)
{
   my ($H, $port) = @_;
   my $p = caller;        # Specify the caller
   $H = $p . "::" . $H;   # Declare the caller
   $port = $port || 3066;
   $proto = getprotobyname('tcp');
   $port = getservbyname($port, 'tcp') unless $port =~ /^\d+/;

   socket($H, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";
   $ent = sockaddr_in($port, INADDR_ANY);
   bind($H, $ent) || die "bind: $!";
   listen($H, 5) || die "listen: $!";
}

#
# Usage: tcp_client handle[, host[, port=3066]]
#
sub tcp_client ($;$$)
{
   my ($H, $host, $port) = @_;
   my $p = caller;        # Specify the caller
   $H = $p . "::" . $H;   # Declare the caller
   $host = $host || "localhost";
   $port = $port || 3066;
```

```
    my $proto = getprotobyname('tcp');
    socket($H, PF_INET, SOCK_STREAM, $proto);
    $port = getservbyname($port, 'tcp') unless $port =~ /^\d+/;
    $ent = sockaddr_in($port, inet_aton($host));
    connect($H, $ent) || die "connect : $!";
    select($H); $| = 1; select(STDOUT);       # Inhibit buffuring
}
1;
```

Appendices

Appendix A: Character Charts

Table A–1: The WCA330 and WCA380 character set

	0	1	2	3	4	5	6	7
0	NUL 0	16	space 32	0 48	@ 64	P 80	‘ 96	p 112
1	1	17	! 33	1 49	A 65	Q 81	a 97	q 113
2	2	18	” 34	2 50	B 66	R 82	b 98	r 114
3	3	19	# 35	3 51	C 67	S 83	c 99	s 115
4	4	20	$ 36	4 52	D 68	T 84	d 100	t 116
5	5	21	% 37	5 53	E 69	U 85	e 101	u 117
6	6	22	& 38	6 54	F 70	V 86	f 102	v 118
7	7	23	’ 39	7 55	G 71	W 87	g 103	w 119
8	8	24	(40	8 56	H 72	X 88	h 104	x 120
9	HT 9	25	) 41	9 57	I 73	Y 89	i 105	y 121
A	LF 10	26	* 42	: 58	J 74	Z 90	j 106	z 122
B	11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
C	12	28	, 44	< 60	L 76	\ 92	l 108	| 124
D	CR 13	29	— 45	= 61	M 77	] 93	m 109	} 125
E	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	15	31	/ 47	? 63	O 79	_ 95	o 111	rubout 127

Table A-2: ASCII & GPIB code chart

B7 B6 B5 BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 **NUL** 0 0	20 **DLE** 10 16	40 LA0 **SP** 20 32	60 LA16 **0** 30 48	100 TA0 **@** 40 64	120 TA16 **P** 50 80	140 SA0 **`** 60 96	160 SA16 **p** 70 112
0 0 0 1	1 GTL **SOH** 1 1	21 LL0 **DC1** 11 17	41 LA1 **!** 21 33	61 LA17 **1** 31 49	101 TA1 **A** 41 65	121 TA17 **Q** 51 81	141 SA1 **a** 61 97	161 SA17 **q** 71 113
0 0 1 0	2 **STX** 2 2	22 **DC2** 12 18	42 LA2 **"** 22 34	62 LA18 **2** 32 50	102 TA2 **B** 42 66	122 TA18 **R** 52 82	142 SA2 **b** 62 98	162 SA18 **r** 72 114
0 0 1 1	3 **ETX** 3 3	23 **DC3** 13 19	43 LA3 **#** 23 35	63 LA19 **3** 33 51	103 TA3 **C** 43 67	123 TA19 **S** 53 83	143 SA3 **c** 63 99	163 SA19 **s** 73 115
0 1 0 0	4 SDC **EOT** 4 4	24 DCL **DC4** 14 20	44 LA4 **$** 24 36	64 LA20 **4** 34 52	104 TA4 **D** 44 68	124 TA20 **T** 54 84	144 SA4 **d** 64 100	164 SA20 **t** 74 116
0 1 0 1	5 PPC **ENQ** 5 5	25 PPU **NAK** 15 21	45 LA5 **%** 25 37	65 LA21 **5** 35 53	105 TA5 **E** 45 69	125 TA21 **U** 55 85	145 SA5 **e** 65 101	165 SA21 **u** 75 117
0 1 1 0	6 **ACK** 6 6	26 **SYN** 16 22	46 LA6 **&** 26 38	66 LA22 **6** 36 54	106 TA6 **F** 46 70	126 TA22 **V** 56 86	146 SA6 **f** 66 102	166 SA22 **v** 76 118
0 1 1 1	7 **BEL** 7 7	27 **ETB** 17 23	47 LA7 **'** 27 39	67 LA23 **7** 37 55	107 TA7 **G** 47 71	127 TA23 **W** 57 87	147 SA7 **g** 67 103	167 SA23 **w** 77 119
1 0 0 0	10 GET **BS** 8 8	30 SPE **CAN** 18 24	50 LA8 **(** 28 40	70 LA24 **8** 38 56	110 TA8 **H** 48 72	130 TA24 **X** 58 88	150 SA8 **h** 68 104	170 SA24 **x** 78 120
1 0 0 1	11 TCT **HT** 9 9	31 SPD **EM** 19 25	51 LA9 **)** 29 41	71 LA25 **9** 39 57	111 TA9 **I** 49 73	131 TA25 **Y** 59 89	151 SA9 **i** 69 105	171 SA25 **y** 79 121
1 0 1 0	12 **LF** A 10	32 **SUB** 1A 26	52 LA10 ***** 2A 42	72 LA26 **:** 3A 58	112 TA10 **J** 4A 74	132 TA26 **Z** 5A 90	152 SA10 **j** 6A 106	172 SA26 **z** 7A 122
1 0 1 1	13 **VT** B 11	33 **ESC** 1B 27	53 LA11 **+** 2B 43	73 LA27 **;** 3B 59	113 TA11 **K** 4B 75	133 TA27 **[** 5B 91	153 SA11 **k** 6B 107	173 SA27 **{** 7B 123
1 1 0 0	14 **FF** C 12	34 **FS** 1C 28	54 LA12 **,** 2C 44	74 LA28 **<** 3C 60	114 TA12 **L** 4C 76	134 TA28 **** 5C 92	154 SA12 **l** 6C 108	174 SA28 **¦** 7C 124
1 1 0 1	15 **CR** D 13	35 **GS** 1D 29	55 LA13 **–** 2D 45	75 LA29 **=** 3D 61	115 TA13 **M** 4D 77	135 TA29 **]** 5D 93	155 SA13 **m** 6D 109	175 SA29 **}** 7D 125
1 1 1 0	16 **SO** E 14	36 **RS** 1E 30	56 LA14 **.** 2E 46	76 LA30 **>** 3E 62	116 TA14 **N** 4E 78	136 TA30 **^** 5E 94	156 SA14 **n** 6E 110	176 SA30 **~** 7E 126
1 1 1 1	17 **SI** F 15	37 **US** 1F 31	57 LA15 **/** 2F 47	77 UNL **?** 3F 63	117 TA15 **O** 4F 79	137 UNT **–** 5F 95	157 SA15 **o** 6F 111	177 **RUBOUT (DEL)** 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

octal →	5 PPC	← **GPIB code (with ATN asserted)**
	ENQ	← ASCII character
hex →	5 5	← decimal

Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: Factory Initialization Settings

The factory initialization settings provide you a known state for the analyzer. Factory initialization sets the values as shown in Table B–1 to B–31 for each command group. Table B–32 and B–35 present those values for the `:Config:Mode` command by its arguments.

Configuration Commands

Table B–1: Factory initialization settings — Configuration commands

Header	Default settings	
Config:Setup	Standard	
Config:BackgroundColor	Black	
Config:MarkerLink	On	
Config:Util<x>	Util1	Self Gain-Calibration
	Util2	Save/Load
	Util3	Average
	Util8	Remote
Config:View:Style	1 × 1	
Config:View<x>	View1	Waveform

Setup Commands

Table B–2: Factory initialization settings — Setup commands

Header	Default settings – Standard	Default settings – CDMA	Default settings – 3GPP
:Setup:ACP:BlockSize	–	–	1
:Setup:ACP:CarrierWidth	–	–	5 MHz
:Setup:ACP:CenterFrequency	–	–	[1]
:Setup:ACP:Span	–	–	15 MHz
:Setup:Band	[1]	RF	[1]
:Setup:BlockSize	Max. 200 (depends on span)	200	–
:Setup:CDMA:Channel	–	1	–
:Setup:CDMA:Standard	–	IS95	–
:Setup:CDMA:TriggerLevel	–	–30 dBm	–
:Setup:CenterFrequency	1.5 GHz	–	–
:Setup:FFTPoints	1024	–	–
:Setup:FFTType	HW	–	SW
:Setup:FFTWindow	Blackman	–	Blackman
:Setup:FramePeriod	80 ms	–	–
:Setup:FrequencyOffset	0	–	0
:Setup:IFMode	[1]	Normal	HiRes
:Setup:InputCoupling	AC	–	–
:Setup:LevelOffset	0	–	0
:Setup:Manual	Mixer	–	Mixer
:Setup:MemoryMode	Frequency	–	Dual
:Setup:MixerLevel	–25 dBm		–25 dBm
:Setup:ReferenceLevel	0 dBm	0 dBm	0 dBm
:Setup:ReferenceOsc	Internal	Internal	Internal
:Setup:RFAtt	20 dB	–	20 dB
:Setup:Span	[1]	5 MHz	–

Table B–2: Factory initialization settings — Setup commands (Cont.)

Header	Default settings – Standard	Default settings – CDMA	Default settings – 3GPP
:Setup:Trigger	Auto	Auto	–
:Setup:TriggerCount	On	On	–
:Setup:TriggerDelayed	0	0	–
:Setup:TriggerDomain	Frequency	Frequency	–
:Setup:TriggerInterval	60	60	–
:Setup:TriggerPosition	50 %	50 %	–
:Setup:TriggerSlope	Rise	Rise	–
:Setup:TriggerSource	Internal	Internal	–
:Setup:TriggerTimeout	0	0	–
:Setup:TriggerTimes	1	1	–
:Setup:Zoom:FFTWindow	Blackman	–	–
:Setup:Zoom:Frequency	0	–	–
:Setup:Zoom:Mag	2	–	–

1 **The previous setting is used.**

View Commands

Table B–3: Factory initialization settings — Waveform View commands

Header	Default settings
:View<x>:Average:Times	10
:View<x>:Average:Type	RMSExpo
:View<x>:Compression	Max
:View<x>:Edit	Off
:View<x>:Format	FreqAmpl
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:Band	Off
:View<x>:Marker:Band:Left	0
:View<x>:Marker:Band:Right	0
:View<x>:Marker:Band:Width	0
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:Trace	Trace1
:View<x>:Marker:X	0
:View<x>:MaskVisible	Off
:View<x>:Measurement	Off
:View<x>:Measurement:ACP:BW	0
:View<x>:Measurement:ACP:Marker	Center
:View<x>:Measurement:ACP:SP	0
:View<x>:Measurement:OBW	99 %
:View<x>:Position	100 %
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active
:View<x>:Trace2:Compression	Max

Table B–3: Factory initialization settings — Waveform View commands (Cont.)

Header	Default settings
:View<x>:Trace2:Format	FreqAmpl
:View<x>:Trace2:Source	None
:View<x>:Trace2:Z	0
:View<x>:Z	0

Table B–4: Factory initialization settings — Analog View commands

Header	Default settings
:View<x>:Format	AM
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	200
:View<x>:Scale:YStart	–200
:View<x>:Source	Active
:View<x>:Z	0

Table B–5: Factory initialization settings — FSK View commands

Header	Default settings
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off

Table B-5: Factory initialization settings — FSK View commands (Cont.)

Header	Default settings
:View<x>:Marker:DeltaX	0
:View<x>:Marker:X	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active
:View<x>:Z	0

Table B-6: Factory initialization settings — Spectrogram View commands

Header	Default settings
:View<x>:Compression	Max
:View<x>:Format	FreqAmpl
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:DeltaZ	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Marker:Z	0
:View<x>:Monochrome	Off
:View<x>:NumberColors	100
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	20
:View<x>:Scale:YStart	0
:View<x>:Scale:ZScale	0
:View<x>:Scale:ZStart	0
:View<x>:Source	Active
:View<x>:ZGap	1 pixel

Table B–7: Factory initialization settings — Waterfall View commands

Header	Default settings
:View<x>:Compression	Max
:View<x>:Format	FreqAmpl
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:DeltaZ	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Marker:Z	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	20
:View<x>:Scale:YStart	0
:View<x>:Scale:ZScale	0
:View<x>:Scale:ZStart	0
:View<x>:Source	Active
:View<x>:YHeight	20
:View<x>:ZGap	5

Table B–8: Factory initialization settings — Polar View commands

Header	Default settings
:View<x>:AlphaBT	0.5
:View<x>:AutoCarrier	On
:View<x>:Burst:BlockSize	4
:View<x>:Burst:NumberFrames	1
:View<x>:Burst:Offset	0
:View<x>:Burst:PeakThreshold	–40 dB
:View<x>:Burst:Search	Off
:View<x>:Burst:Threshold	–20 dB
:View<x>:Carrier	0
:View<x>:Display	Measurement
:View<x>:Format	Vector

Table B–8: Factory initialization settings — Polar View commands (Cont.)

Header	Default settings
:View<x>:Marker:T	0
:View<x>:Mask:Frequency	Off
:View<x>:Mask:MarkerLink	Off
:View<x>:MeasDestination	D5D6
:View<x>:MeasFilter	RootRaisedCosine
:View<x>:Modulation	1/4 π QPSK
:View<x>:Position	100 %
:View<x>:RefDestination	D7D8
:View<x>:RefFilter	RaisedCosine
:View<x>:Source	Active
:View<x>:SymbolRate	21 k
:View<x>:Z	0

Table B–9: Factory initialization settings — Eye diagram View commands

Header	Default settings
:View<x>:EyeLength	2
:View<x>:Format	I
:View<x>:Marker:T	0
:View<x>:Source	Measurement

Table B–10: Factory initialization settings — Symbol View commands

Header	Default settings
:View<x>:Marker:T	0
:View<x>:Radix	Bin
:View<x>:Rotate	0
:View<x>:Source	Measurement

Table B–11: Factory initialization settings — EVM View commands

Header	Default settings
:View<x>:Format	EVM
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:X	0
:View<x>:MaskArea	5 %
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	100
:View<x>:Scale:YStart	0

Table B–12: Factory initialization settings — CDMA Waveform View commands

Header	Default settings
:View<x>:Average:Times	10
:View<x>:Average:Type	RMSExpo
:View<x>:CDMA:Channel	1
:View<x>:CDMA:Standard	IS95
:View<x>:Compression	Max
:View<x>:Format	FreqAmpl
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Mask:RBW1M:Frequency	1.385 MHz

Table B–12: Factory initialization settings — CDMA Waveform View commands (Cont.)

Header	Default settings
:View<x>:Mask:RBW1M:Level	–60
:View<x>:Mask:RBW30k:Frequency1	900 k
:View<x>:Mask:RBW30k:Frequency2	1.98 MHz
:View<x>:Mask:RBW30k:Level1	–42
:View<x>:Mask:RBW30k:Level2	–54
:View<x>:Measurement	Off
:View<x>:Measurement:OBW	99
:View<x>:Measurement:Separation	2
:View<x>:Measurement:SortedBy	Frequency
:View<x>:Measurement:SpuriousSearch	Off
:View<x>:Measurement:Threshold	–100
:View<x>:Position	100 %
:View<x>:RBW	30 kHz
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active
:View<x>:Z	0

Table B–13: Factory initialization settings — CDMA Polar View commands

Header	Default settings
:View<x>:AlphaBT	0.2
:View<x>:AutoCarrier	On
:View<x>:Burst:BlockSize	5
:View<x>:Burst:NumberFrames	2
:View<x>:Burst:Offset	0
:View<x>:Burst:PeakThreshold	–40 dB
:View<x>:Burst:Search	Off
:View<x>:Burst:Threshold	–20 dB
:View<x>:Carrier	0

Table B-13: Factory initialization settings — CDMA Polar View commands (Cont.)

Header	Default settings
:View<x>:Display	Measurement
:View<x>:Format	Constellation
:View<x>:Marker:T	0
:View<x>:Mask:Frequency	Off
:View<x>:Mask:MarkerLink	Off
:View<x>:MeasDestination	D5D6
:View<x>:MeasFilter	RootRaisedCosine
:View<x>:Modulation	1/4 π QPSK
:View<x>:Position	100 %
:View<x>:RefDestination	D7D8
:View<x>:RefFilter	RaisedCosine
:View<x>:Source	Active
:View<x>:SymbolRate	1.2288 MHz
:View<x>:Z	0

Table B-14: Factory initialization settings — CDMA Time View commands

Header	Default settings
:View<x>:Average:Times	10
:View<x>:Block	0
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Mask:OffLeft	169 μs
:View<x>:Mask:OffLevel	–20
:View<x>:Mask:OffRight	1.431 ms
:View<x>:Mask:OnLeft	175 μs

Table B–14: Factory initialization settings — CDMA Time View commands (Cont.)

Header	Default settings
:View<x>:Mask:OnLevel	–3
:View<x>:Mask:OnRight	1.425 ms
:View<x>:Position	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active
:View<x>:Trace:TraceVisible	On
:View<x>:Trace2:TraceVisible	Off

Table B–15: Factory initialization settings — CodeSpectrogram View commands

Header	Default settings
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Marker:Z	0
:View<x>:Monochrome	Off
:View<x>:NumberColors	100
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	20
:View<x>:Scale:YStart	0
:View<x>:Scale:ZScale	0
:View<x>:Scale:ZStart	0
:View<x>:YAxis	Relative
:View<x>:ZGap	3 pixels

Table B–16: Factory initialization settings — CodePolar View commands

Header	Default settings
:View<x>:AlphaBT	0.2
:View<x>:Analysis:TimeSlot	0
:View<x>:AutoCarrier	On
:View<x>:Carrier	0
:View<x>:ChipRate	1.2288 MHz
:View<x>:Display	Measurement
:View<x>:Format	Vector
:View<x>:Marker:T	0
:View<x>:MeasFilter	RootRaisedCosine
:View<x>:Modulation	IS-95+EQ
:View<x>:RefFilter	RaisedCosine
:View<x>:Source	Active
:View<x>:Z	0

Table B–17: Factory initialization settings — CodePower View commands

Header	Default settings
:View<x>:Average	Off
:View<x>:Average:BeginZ	0
:View<x>:Average:EndZ	0
:View<x>:Average:Times	10
:View<x>:Average:Type	RMSExpo
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0

Table B-17: Factory initialization settings — CodePower View commands (Cont.)

Header	Default settings
:View<x>:Scale:YStart	0
:View<x>:XAxis	Code
:View<x>:YAxis	Relative
:View<x>:Z	0

Table B-18: Factory initialization settings — CodeWSpectrogram and 3gppSpectrogram View commands

Header	Default settings
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Marker:Z	0
:View<x>:Monochrome	Off
:View<x>:NumberColors	100
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	20
:View<x>:Scale:YStart	0
:View<x>:Scale:ZScale	0
:View<x>:Scale:ZStart	0
:View<x>:SymbolRate	Composite
:View<x>:YAxis	Relative
:View<x>:ZGap	3 pixels

Table B-19: Factory initialization settings — CodeWPolar View commands

Header	Default settings
:View<x>:AlphaBT	0.22
:View<x>:Analysis:TimeSlot	0
:View<x>:AutoCarrier	On
:View<x>:Carrier	0

Table B–19: Factory initialization settings — CodeWPolar View commands (Cont.)

Header	Default settings
:View<x>:ChipRate	4.096 M
:View<x>:Constellation:SymbolRate	Composite
:View<x>:Display	Measurement
:View<x>:Format	Vector
:View<x>:Marker:T	0
:View<x>:MeasFilter	RootRaisedCosine
:View<x>:Modulation	W-CDMA
:View<x>:RefFilter	RaisedCosine
:View<x>:ShortCode	0
:View<x>:Source	Active
:View<x>:SymbolConstellation	Off
:View<x>:TimeSlot	0

Table B–20: Factory initialization settings — CodeWPower and 3gppPower View commands

Header	Default settings
:View<x>:Average	Off
:View<x>:Average:BeginZ	0
:View<x>:Average:EndZ	0
:View<x>:Average:Times	10
:View<x>:Average:Type	RMSExpo
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	50
:View<x>:Scale:YStart	–50

Table B–20: Factory initialization settings — CodeWPower and 3gppPower View commands (Cont.)

Header	Default settings
:View<x>:ShortCode	0
:View<x>:SymbolRate	Composite
:View<x>:XAxis	ShortCode
:View<x>:YAxis	Relative
:View<x>:Z	0

Table B–21: Factory initialization settings — 3gppACPView commands

Header	Default settings
:View<x>:Average	Off
:View<x>:Average:BeginZ	0
:View<x>:Average:EndZ	0
:View<x>:Average:Times	10
:View<x>:Average:Type	RMSExpo
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:Band:Center	0
:View<x>:Marker:Band:Left	0
:View<x>:Marker:Band:Right	0
:View<x>:Marker:Band:Width	0
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:MaskVisible	Off
:View<x>:Measurement	ACP
:View<x>:Measurement:ACP:BW	4.096 M
:View<x>:Measurement:ACP:SP	5 M
:View<x>:Measurement:Filter	Off
:View<x>:Measurement:Filter:Alpha	0.22

Table B–21: Factory initialization settings — 3gppACPView commands (Cont.)

Header	Default settings
:View<x>:Measurement:Filter:BW	4.096 M
:View<x>:Measurement:Filter:SP	5 M
:View<x>:Measurement:OBW	99 %
:View<x>:RBW	0
:View<x>:RBW:Alpha	0.5
:View<x>:RBW:Calculation	Off
:View<x>:Scale:HoldYScale	Off
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active

Table B–22: Factory initialization settings — 3gppPolar View commands

Header	Default settings
:View<x>:AlphaBT	0.22
:View<x>:Analysis:TimeSlot	0
:View<x>:AutoCarrier	On
:View<x>:Carrier	0
:View<x>:ChipRate	4.096 MHz
:View<x>:Constellation:SymbolRate	Composite
:View<x>:Display	Measurement
:View<x>:Downlink:ScramblingCode	0
:View<x>:Downlink:ScramblingCodeSearch	On
:View<x>:Format	Vector
:View<x>:HideSCHPart	On
:View<x>:Marker:T	0
:View<x>:MeasFilter	RootRaisedCosine
:View<x>:Modulation	W-CDMA
:View<x>:RefFilter	RaisedCosine
:View<x>:ShortCode	0
:View<x>:Source	Active
:View<x>:SymbolConstellation	Off
:View<x>:TimeSlot	0

Table B–23: Factory initialization settings — GSM View commands

Header	Default settings
:View<x>:Burst	10
:View<x>:Script	EVM

Table B–24: Factory initialization settings — GSMPolar View commands

Header	Default settings
:View<x>:Burst:AnalysisLength	148
:View<x>:Burst:SyncWordLength	26
:View<x>:Burst:SyncWordPattern	TSC0
:View<x>:Burst:SyncWordPosition	61
:View<x>:Marker:T	0

Table B–25: Factory initialization settings — GSMMask View commands

Header	Default settings
:View<x>:Line:X<n>	0
:View<x>:Line:X<n>:Visible	Off
:View<x>:Line:Y<n>	0
:View<x>:Line:Y<n>:Visible	Off
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:DeltaX	0
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Scale:XScale	160 μs
:View<x>:Scale:XStart	0
:View<x>:Scale:YScale	100
:View<x>:Scale:YStart	–100

Table B–26: Factory initialization settings — CCDF commands

Header	Default settings
:View<x>:BeginZ	0
:View<x>:Destination	D1
:View<x>:EndZ	0
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:OutputFormat	CCDF
:View<x>:Position	100 %
:View<x>:Resolution	0.1
:View<x>:Scale:HoldYScale	Off
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Scale:XStartZero	Off
:View<x>:Scale:YScale	0
:View<x>:Scale:YStart	0
:View<x>:Source	Active
:View<x>:Z	0

Table B–27: Factory initialization settings — CCDFView commands

Header	Default settings
:View<x>:Marker:DeltaMarker	Off
:View<x>:Marker:SearchSeparation	2 %
:View<x>:Marker:X	0
:View<x>:Scale:XScale	1
:View<x>:Scale:XStart	0
:View<x>:Source	Active

Utility Commands

Table B–28: Factory initialization settings — Self gain-calibration commands

Header	Default settings
:Util1:Gain:Auto	On

Table B–29: Factory initialization settings — Save/Load commands

Header	Default settings
:Util2:BeginZ	0
:Util2:EndZ	0
:Util2:Source	Active

Table B–30: Factory initialization settings — Average commands

Header	Default settings
:Util3:BeginZ	0
:Util3:Destination	D1
:Util3:EndZ	0
:Util3:Source	Active

Table B–31: Factory initialization settings — Remote commands

Header	Default settings
:Util8:Format	FreqAmpl
:Util8:GPIB:Interface	Off
:Util8:GPIB:PrimaryAddress	1
:Util8:Header	Off
:Util8:NewLine	CRLF
[:Util8]:NumericOutput	Exp
[:Util8]:Source	Active
[:Util8]:TCPIP:Port:Command	3066
[:Util8]:TCPIP:Port:Event	3067
[:Util8]:Z	0

:Config:Mode Command

The `:Config:Mode` command (refer to page 4–58) overwrites the above default values as shown in Table B–32 through B–35, depending on its arguments.

Table B–32: Factory initialization settings — :Config:Mode command – 1

	Freq1	Freq2	Dual	Zoom
View style	1 × 1	1 × 2	1 × 4	2 × 2
Setup	Standard	Standard	Standard	Standard
View1	Waveform	Waveform	Waveform	Waveform
View2	–	Spectrogram	Spectrogram	Spectrogram
View3	–	–	Waveform (Time vs I)	Waveform
View4			Waveform (Time vs Q)	Spectrogram
View5 – View8	None	None	None	None
Trigger mode	Auto	Auto	Auto	Auto
Memory mode	Frequency	Frequency	Dual	Zoom

Table B–33: Factory initialization settings — :Config:Mode command – 2

	opCDMA1	opCDMA2	opCDMA3	opCDMA4	opCDMA5	opCDMA6	opDemod1
View style	2 × 2	1 × 2	1 × 2	2 × 2	1 × 2	1 × 2	2 × 2
Setup	CDMA	CDMA	CDMA	CDMA	CDMA	CDMA	Standard
View1	CDMA Waveform	CDMA Waveform	CDMA Time	CDMA Waveform	CDMA Waveform	CDMA Time	Waveform
Hor. Start	0			0			0
Position	50 %	50 %		50 %	50 %		
Measurement	Power	Spurious		Power	Spurious		
View2	Spectrogram	CDMA Waveform	CDMA Time	Spectrogram	CDMA Waveform	CDMA Time	Spectrogram
Position		50 %			50 %		
Measurement		Spurious			Spurious		
View3	CDMA Polar	None	None	CDMA Polar	None	None	Polar
Position	50 %			50 %			
View4	EVM	None	None	EVM	None	None	Eye diagram
View5 – View8	None	None	None	None	None	None	None
Standard	IS-95	IS-95	IS-95	T-53	T-53	T-53	
Span	5 MHz	30 MHz	5 MHz	5 MHz	30 MHz	5 MHz	
Trigger mode	Auto		Normal	Auto		Normal	
Memory mode							Dual
Input mode	RF	RF	RF	RF	RF	RF	

Table B–34: Factory initialization settings — :Config:Mode command – 3

	opCode1	opCodeW1	op3gpp1	op3gpp2	opGSM1
View style	2 2	2 2	1 1	2 2	2 2
Setup	Standard	Standard	3gppACP	Standard	Standard
View1	Waveform	Waveform	3gppACPView	Waveform	GSMPolar
View2	Code-Spectrogram	CodeW-Spectrogram	None	3gpp-Spectrogram	SymbolTable
View3	CodePolar	CodeWPolar	None	3gppPolar	EVM
View4	CodePower	CodeWPower	None	3gppPower	GSM
View5 – View8	None	None	None	None	None
Span	5 MHz	10 MHz	15 MHz	10 MHz	1 MHz / 5 MHz [1]
Trigger mode	Auto	Auto	–	Auto	–
Memory mode	Zoom	Zoom	Zoom	Zoom	Zoom
Input mode	RF	Wide	RF	Wide	RF

[1] **5 MHz for the Spectrum (Peak) and Spectrum (RMS) measurement.**

Table B–35: Factory initialization settings — :Config:Mode command – 4

	opCCDF
View7	CCDF
View8	CCDFView

Glossary and Index

Glossary

Amplitude Modulation (AM)
The process, or result of a process, in which the amplitude of a sine wave (the carrier) is varied in accordance with the instantaneous voltage of a second electrical signal (the modulating signal).

Bin
A sample point in frequency domain. The frequency bandwidth of a bin is the span divided by the number of bins.

BNF (Backus-Naur Form)
A standard notation system for command syntax.

Block
A group of the specified number of frames.

Carrier Frequency
The frequency of the carrier signal.

Carrier-to-Noise Ratio (C/N)
The ratio of carrier signal power to average noise power in a given bandwidth surrounding the carrier; usually expressed in decibels.

Center Frequency
That frequency which corresponds to the center of a frequency span, expressed in hertz.

dBm
A unit of expressed power level in decibels referenced to 1 milliwatt.

Decibel (dB)
Ten times the logarithm of the ratio of one electrical power to another.

Display Reference Level
A designated vertical position representing a specified input level. The level may be expressed in dBm, volts, or any other units.

EVM
Acronym for the Error Vector Magnitude.

Frame
An area reserved in memory with the length represented by the number of FFT points. The analyzer acquires data in one frame by one scan in realtime mode.

Filter
A circuit which separates electrical signals or signal components based on their frequencies.

Frequency Band
The continuous range of frequencies extending between two limiting frequencies, expressed in hertz.

Frequency Domain Representation
The portrayal of a signal in the frequency domain; representing a signal by displaying its sine wave components; the signal spectrum.

Frequency Modulation (FM)
The process, or result of a process, in which the frequency of an electrical signal (the carrier) is varied in accordance with some characteristic of a second electrical signal (the modulating signal or modulation).

Frequency Range
That range of frequencies over which the performance of the instrument is specified.

Frequency Response
The unwanted variation in the displayed amplitude over a specified center frequency range, measured at the center frequency expressed in decibels.

Frequency Span (Dispersion)
The magnitude of the frequency band displayed; expressed in hertz or hertz per division.

Full Span (Maximum Span)
A mode of operation in which the spectrum analyzer scans an entire frequency band.

Markers
When the Marker function is enabled, it provides a movable cursor with readout of frequency and amplitude at the marker position. When the delta marker mode is enabled, a second marker allows operations and readout between the two marker positions.

Peak Hold
Digitally stored display mode which, at each frequency address, compares the incoming signal level to the stored level and retains the greater. In this mode, the display indicates the peak level at each frequency after several successive sweeps.

MAX/MIN
A display mode on the spectrum analyzer that shows the maximum and minimum signal levels at alternate frequency points; its advantage is its resemblance to an analog display.

MIN Hold
A spectrum analyzer feature which captures the minimum signal amplitude at all displayed frequencies over a series of sweeps.

Modulate
To regulate or vary a characteristic of a signal.

Modulating Signal
The signal which modulates a carrier. The signal which varies or regulates some characteristic of another signal.

Modulation
The process of varying some characteristic of a signal with a second signal.

Peak Detection
A detection scheme wherein the peak amplitude of a signal is measured and displayed. In spectrum analysis, 20 log (peak) is often displayed.

Reference Level
The signal level required to deflect the CRT display to the top graticule line.

Resolution Bandwidth (RBW)
The width of the narrowest filter in the IF stages of a spectrum analyzer. The RBW determines how well the analyzer can resolve or separate two or more closely spaced signal components.

Span Per Division, Span/Div
Frequency difference represented by each major horizontal division of the graticule.

Spurious Response
A response to a spectrum analyzer wherein the displayed frequency is not related to the input frequency.

Vertical Scale Factor, Vertical Display Factor
The number of dB, volts, etc., represented by one vertical division of a spectrum analyzer display screen.

View (Display)
Enables viewing of contents of the chosen memory section (e.g., “View A” displays the contents of memory A; “View B” displays the contents of memory B).

Index

Symbols

Numbers

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W