LeCroy WaveStation
LW400/LW400A Series AWG
Remote Programmer's Manual

August 1996
Rev. C

LeCroy

Corporate Headquarters
700 Chestnut Ridge Road
Chestnut Ridge, NY 10977–6499
Tel: (914) 578-6020, FAX: 578-5985

European Headquarters
Mannheimerstrasse 175
D-69123 Heidelberg, Germany
Tel: (49) 6221 840989
FAX: (49) 6221 833827

European Manufacturing
2, rue du Pré-de-la-Fontaine
P.O. Box 341
1217 Meyrin 1/Geneva, Switzerland
Tel: (41) 22 719 21 11, FAX: 22 782 39 15

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

THIS PAGE LEFT INTENTIONALLY BLANK

1 GENERAL INFORMATION

INITIAL INSPECTION

It is recommended that the shipment be thoroughly inspected immediately upon delivery to the purchaser. All material in the container should be checked against the enclosed Packing List. LeCroy cannot accept responsibility for shortages in comparison with the Packing List unless notified promptly. If the shipment is damaged in any way, please contact the Customer Service Department.

WARRANTY

LeCroy warrants its products to operate within specifications under normal use for a period of one year from the date of shipment. Spares, replacement parts and repairs are warranted for 90 days. The instrument's firmware is thoroughly tested and thought to be functional, but is supplied "as is" with no warranty of any kind covering detailed performance. Products not manufactured by LeCroy are covered solely by the warranty of the original equipment manufacturer.

In exercising this warranty, LeCroy will repair or, at its option, replace any product returned to the Customer Service Department or an authorized service facility within the warranty period, provided that the warrantor's examination discloses that the product is defective due to workmanship or materials and that the defect has not been caused by misuse, neglect, accident or abnormal conditions or operation.

The purchaser is responsible for transportation and insurance charges for the return of products to the servicing facility. LeCroy will return all in–warranty products with transportation prepaid. This warranty is in lieu of all other warranties, expressed or implied, including but not limited to any implied warranty of merchantability, fitness, or adequacy for any particular purpose or use. LeCroy shall not be liable for any special, incidental, or consequential damages, whether in contract or otherwise.

PRODUCT ASSISTANCE

Answers to questions concerning installation, calibration, and use of LeCroy equipment are available from the Customer Service Dept., 700 Chestnut Ridge Road, Chestnut Ridge, New York 10977–6499, U.S.A., tel. (914)578–6020.

MAINTENANCE AGREEMENTS

LeCroy offers a selection of customer support services. Maintenance agreements provide extended warranty and allow the customer to budget maintenance costs after the initial one year warranty has expired. Other services such as installation, training, enhancements and on-site repair are available through specific Supplemental Support Agreements.

UPDATED MANUALS

LeCroy is committed to providing state-of-the-art instrumentation and is continually refining and improving the performance of its products. While physical modifications can be implemented quite rapidly, the corrected documentation frequently requires more time to produce. Consequently, this manual may not agree in every detail with the accompanying product. There may be small discrepancies in the values of components for the purposes of pulse shape, timing, offset, etc., and occasionally, minor logic changes. Where any such inconsistencies exist, please be assured that the unit is correct and incorporates the most up-to-date circuitry. In a similar way the firmware may undergo revision when the instrument is serviced. Should this be the case, manual updates will be made available as necessary.

SERVICE PROCEDURE

Products requiring maintenance should be returned to the Customer Service Department or authorized service facility. LeCroy will repair or replace any product under warranty at no charge. The customer is responsible for transportation charges to the factory. All in–warranty products will be returned to the customer with transportation prepaid.

For all LeCroy products in need of repair after the warranty period, the customer must provide a Purchase Order Number before repairs can be initiated. The customer will be billed for parts and labor for the repair, as well as for shipping.

RETURN PROCEDURE

To determine your nearest authorized service facility, contact the Customer Service Department or your field office. All products returned for repair should be identified by the model and serial numbers and include a description of the defect or failure, name and phone number of the user, and, in the case of products returned to the factory, a Return Authorization Number (RAN). The RAN may be obtained by contacting the Customer Service Department in New York, tel. (914)578–6020. Return shipments should be made prepaid. LeCroy will not accept C.O.D. or Collect Return Shipments. Wherever possible, the original shipping carton should be used. If a substitute carton is used, it should be rigid and be packed such that the product is surrounded with a minimum of four inches of excelsior or similar shock-absorbing material. In addressing the shipment, it is important that the Return Authorization Number be displayed on the outside of the container to ensure its prompt routing to the proper department within LeCroy.

HOW TO USE THIS MANUAL

This manual explains the programming protocol for controlling the LW400/LW400A Series Arbitrary Waveform Generators, including the LW420,LW420A, LW410 and LW410A, from a host computer. These models may also be reffered to as the WaveStation.

Pupose of this manual:

- Gain an overview of the instrument remote programming interface.
- Familiarize yourself with the SCPI programming language as it applies to the LW400/LW400A.
- Provide detailed information on all of the WaveStation remote commands.

The following sections are contained in this manual:

Section 1 — **Introduction**
Gives a brief history of remote control interfaces and protocols and explains the advantages of the SCPI command language and how it is used in the WaveStation.

Section 2 — **About Remote Control**
Explains how to operate the WaveStation remotely across the GPIB bus.

Section 3 — **Instrument Model and Subsystem Hierarchy**
Presents the function representation of the instrument as viewed from the remote control interface, often referred to as the instrument Model. Describes the command hierarchy and introduces basic SCPI syntax and subsystems. Provides an overview of the command hierarchy and how it relates to the arbitrary waveform generator functional sections.

Section 4 — **Status and Error Reporting**
Describes in detail the Status and Error reporting system.

Section 5 — **Waveform Transfers via GPIB**
Explains the format for transferring waveforms between an external device and the WAVESTATION via GPIB.

Section 6 — **Remote Commands**
Provides a detailed command reference, including command syntax and purpose.

Section 7 — **Remote Programming Example**

Introduction

The remote control interface consists of hardware, the GPIB port, as well as a software protocol. The hardware interfaces are described in your user manual for the instrument. The software protocol is described in this manual and builds upon the rapidly emerging industry standard SCPI (Standard Commands For Programmable Instruments).

What is SCPI

SCPI is a remote command language for test and measurement instruments. It was developed by a consortium of test and measurement instrument manufacturers and is intended to provide a consistent programming language for instrument control and data transfer.

IEEE-488 (GPIB) was adopted as a standard remote control interface in 1975. The standard specified system interconnections and communication protocols which provided a universal hardware interface for integrating multiple instruments into a test system. The original standard put instruments on a common bus, but each instrument manufacturer used a proprietary command set. Every time a user added a new instrument to the bus, he had to learn another set of, often enigmatic, commands. Updates to the standard in 1987, led to IEEE-488.1 and 488.2 which further refined the standard but still fell short of ensuring a common command syntax beyond a few mandated "common commands". In 1990, the Standard Commands for Programmable Instruments (SCPI) consortium developed a system of common remote commands. Although SCPI was originally defined for GPIB, it has now spread well beyond that interface and is being used to support a wide range of hardware interfaces. For example SCPI has became a major element in the implementation of VXI based systems.

The SCPI command language standardizes command syntax and structure used in remote control of test and measurement instrumentation and is being rapidly adopted by leaders in test & measurement instrumentation. This allows the user to learn a single set of remote commands for instruments which are supplied by different manufacturers. Because the functionality of instruments can vary widely, and because new instruments and measurement techniques are constantly being developed, the SCPI standard makes provision for new commands to be added

as needed. Because LW400 has many unique features (for example, waveform formats), LeCroy has enhanced the SCPI language to provide access to these advanced capabilities.

SCPI benefits the user by providing a single command set for integrating multiple instruments into a test system. The greatest benefit occurs on the second or subsequent system integration programs, where the user does not learn yet another command language.

This manual will provide you with all the information you require to control your LW400 using the SCPI programming language. Because SCPI is an industry standard and not specific to LeCroy, details on the generic standard are available in industry standard SCPI manuals.

Interface Configuration and Special Commands

The WaveStation can be operated remotely from an instrument controller or computer across the GPIB bus and commands sent over GPIB can set or read any WaveStation front panel instruction.

GPIB Remote Control

The GPIB bus can interconnect many instruments to allow communication with one another over shared cables. The GPIB bus uses a bit-parallel, byte-serial format. A device connected to the GPIB is either a talker, listener, or controller. Although some devices can change roles, a device can perform just one role at a time.

Talker Places messages or data on the GPIB bus for transmission to other devices. Only one device on the network can be the talker.

Listener Receives data or commands over the bus. Several listeners may be active at one time.

Controller

Governs the operation of the bus. A controller, usually a computer, normally sends program messages to devices and receives responses from them. One controller task is to decide which device is the talker and which is a listener(s). The controller may assign itself to be the talker at one time, and a listener at other times. If devices on the bus never change their roles, a controller is not required.

GPIB Signals and Lines

The GPIB bus has 16 signal lines and eight ground lines. Eight of the 16 signal lines form a bi-directional data bus which transfers data and commands. The remaining eight signal lines control the bus operation. Three lines are for handshaking signals which synchronize data transmission. The remaining five lines are management lines which control the flow of information across the bus and take special action.

Setting the GPIB Address

The GPIB address is set in the System Sub-menu, accessed through the Project and Preference menu. From the front panel press the Project key. Press the soft keys adjacent to the Preferences and then system entries on the menus to enter the system menu. Press the soft key adjacent to the GPIB entry on the

menu to enter the GPIB setup menu. Turn the rotary to select the GPIB address.

The factory default setting for the GPIB address is 1.

GPIB Remote Control and Hardcopy Operation

The WaveStation can communicate across the GPIB bus as a talker or as a listener with a remote host controller (computer). For this talker/listener remote control operation, the WaveStation conforms to the guidelines specified by IEEE 488. The hardcopy output can also communicate across GPIB in one of two ways. First, if the hardcopy port is the same as the remote control port, then a remote hardcopy command sends the output to the remote host as a query response. Second, if the hardcopy port is different from the remote control port or the local hardcopy key is pressed (Hardcopy Execute), then the WaveStation enters talk only mode and does not expect any controller present on the bus.

Remote Control Operation over GPIB

Talk/Listen

The WaveStation enters this mode whenever a command is received via the GPIB bus. In this mode, the Wavestation can both receive commands and setups from the remote host computer (controller) and send data and measurement results.

End or Identify (EOI) Operation

Except where specifically noted, all commands to and from the WaveStation are terminated by asserting the EOI signal line simultaneously with the last byte transmitted. No other command terminators are required.

Hardcopy Operation over GPIB

Talk Only

The WaveStation enters this mode whenever the hardcopy destination is set to GPIB and the Hardcopy Execute soft key is pressed. Talk only is a special GPIB mode where there is no controller allowed on the bus; the WaveStation is the only talker and all connected devices must be listeners (i.e., printers/plotters must be in Listen Only mode).

Talk/Listen

If hardcopy destination is GPIB and then sending the HCOPy command over the GPIB bus will cause the WaveStation to send the hardcopy output to the host computer as a response message. In this mode, the WaveStation will wait to be addressed to talk before sending the hardcopy data. The host computer then has three options in generating the hardcopy:

1) The host computer may read the data into internal memory and then send the data to a printer/plotter.
2) The host computer may send the HCOPy remote command and then address the printer to listener and the WaveStation to talk and read the data from the WaveStation. As the data is read into the computer, it is also printed to the printer which is a listener.
3) The host computer may send the HCOPy remote command and then address the printer/plotter to listen, the WaveStation to talk, and the controller to go into stand-by mode waiting for EOI.

IEEE-488 Standard Messages

This section explains how the WaveStation reacts to the Standard 488.2 messages.

Serial Poll Function

The WaveStation implements a full Serial Poll Interface Function:

1. It can assert the SRQ (Service Request) control line.
2. It will respond with the current serial poll byte or STB when addressed to Talk and after the Serial Poll Enable interface message is received.
3. After transmitting its status message, the WaveStation stops asserting the SRQ line and clears its internal status byte.

Receiving the Trigger Message

The WaveStation responds to the Trigger message [*TRG command] by triggering the output waveform. It is executed after all previously received commands have been processed.

Interface Clear

The Interface Clear message (asserting IFC line) is an asynchronous control line that causes all bus activity to halt. When the WaveStation receives the IFC message, it becomes unaddressed, stops talking or listening, and will not participate in future bus transactions until readdressed to talk or listen.

Device Clear (Selective or Universal)

The WaveStation will respond to a Selective Device Clear or a Universal Device Clear interface message. The former requires that the WaveStation first be addressed to listen, followed by the Selective Device Clear message. The latter does not require that the instrument be previously addressed to listen. Device Clear causes the input buffer, the output queue, and the message available (MAV) status bit to be cleared.

Go to Local, Go to Remote, Go to Remote with Lockout Local

The WaveStation can operate in Local or Remote mode. In Local mode, all front panel controls are operational and commands from the host computer will also be processed. In Remote mode, the WaveStation operates under computer control and no front panel controls are operational except the Local soft key (if enabled). The WaveStation always powers on in Local mode).

Note: The WaveStation processes all messages regardless of being in Remote or Local modes.

The WaveStation switches to Remote mode (with Local soft key enabled) when the WaveStation receives a command with the REN line asserted. All instrument settings remain unchanged during local-to-remote transitions. The WaveStation screen indicates that Remote mode is enabled by the appearance of the Local soft key. No other front panel controls operate.

If the WaveStation is under remote control and the Local soft key is pressed, the instrument interrupts program control and returns to local control. Data and/or settings cannot be changed locally.

***Caution:**. In Local Lockout state, all front panel keys and knobs are disabled. Once Remote with Local Lockout is set using the "RWLS" or "LLO" commands it can only be cleared when the* WaveStation *is put into Local mode by sending the "LOC" command or readdressing the* WaveStation *with REN deasserted.*

Checking GPIB Communications Using National Instruments IBIC Program

This quick checkout requires a computer with a National Instrument GPIB card and the National Instruments IBIC program supplied by National Instruments with the purchase of a GPIB card. This quick checkout also assumes that the GPIB card is already installed in the computer and has passed all test successfully. For help installing or configuring the National Instruments GPIB card please contact National Instruments at (800) IEEE-488 or (512) 794-0100.

These example instructions are for an IBM-PC or compatible computer. The method for other computers is very similar.

Change to the National Instruments GPIB-PC subdirectory with the command:

```
CD \GPIB-PC
```

Start the IBIC program by with the command:

```
IBIC
```

Tell the IBIC program the address of the WaveStation (we assume address 1) with the command:

```
IBFIND DEV1
```

Send the identify command to the WaveStation with the command:

```
IBWRT "*IDN?"
```

Read the id of the WaveStation with the command:

```
IBRD 100
```

The WaveStation response should have included the model number, serial number and other information. The full IBIC sequence should look as follows:

```
National Instruments Interface Bus
Interactive Control Program (IBIC)
Copyright 1984, 1989 National Instruments Corporation.
All rights reserved.

Type 'help' for help.

Use IBFIND to initially open a board or device.
Use SET to select an already opened board or device.

: IBFIND DEV1

dev1:  IBWRT "*IDN?"
[0100]        ( cmpl )
count: 55

dev1:  IBRD 100
[2100]     ( end cmpl )
count: 31
4C 65 43 72 6F 79 2C 4C          L e C r o y , L
57 34 30 30 2c 4c 57 34          W 4 0 0 , L W 4
32 30 2f 55 31 30 30 30          2 0 / U 1 0 0 0
2C 31 2e 34 2e 32 0a             , 1 . 4 . 2 .
```

If IBIC returned an error on any of the commands, double check to make sure you typed the command exactly as given above, then consult the National Instruments GPIB-PC manual for help interpreting the error codes. A brief list of some of the common errors and possible solutions follows:

Error Code	Check
EDVR	Check that config.sys contains the line: device = c:\dir\GPIB.COM where dir is the directory that contains GPIB.COM.
ENOL	No listener. Check IBFIND DEVx matches the GPIB address of the WaveStation. Where the WaveStation GPIB address is x.
EARG	Invalid argument. Check that the command was typed correctly.
ESAC	GPIB board is not system controller. Check to make sure the GPIB board is configured as controller using IBCONF.
EABO	Check that the WaveStation is powered on and cables are connected securely.
ENEB	Can't find GPIB board. Check GPIB installation and configuration.

In Case of GPIB Communications Problems Check the Following:

1. WaveStation is turned on, and finished booting up.
2. WaveStation passes power up self tests.
3. GPIB board is installed and passes all tests. (See National Instruments IBTEST).
4. GPIB cable is connected securely.
5. GPIB address is set correctly.
6. No other instrument on the GPIB bus is set to the same address.
7. GPIB name (DEV1) set in IBFIND command corresponds to the name given in the IBCONF device map for address 1.

This page left intentionally blank

Remote Command System Model

It is important to understand the remote control subsystem hierarchy in order to rapidly locate the desired command and associated message you require. Figure 1 shows the functional block diagram of the arbitrary waveform generator as viewed by the remote programming interface. The structure of the instrument subsystems is closely related to this block diagram.

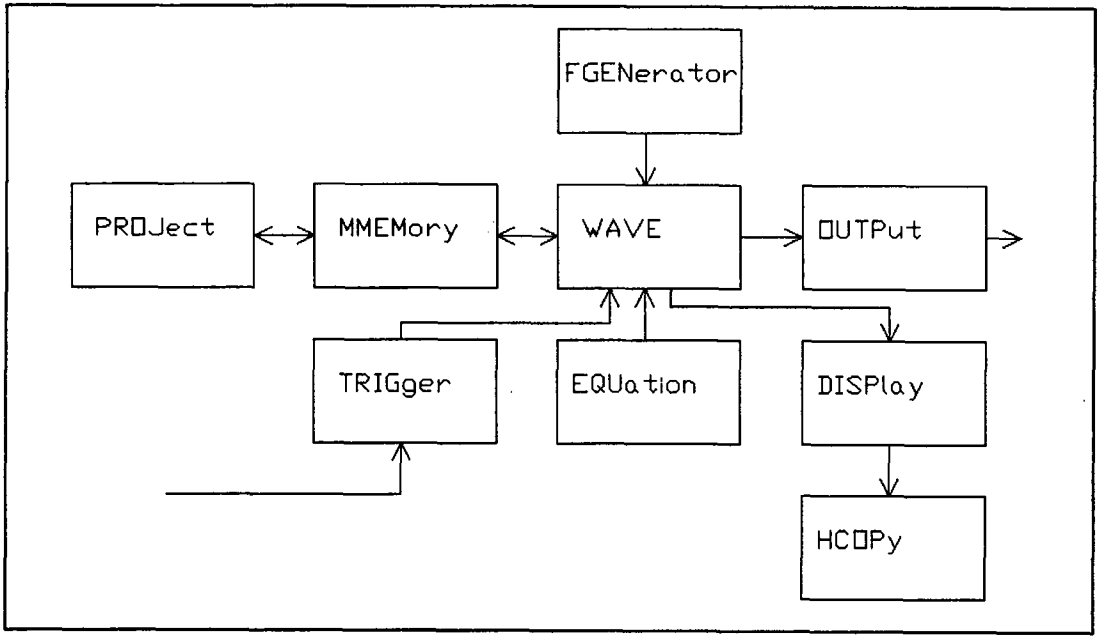


Figure 1

Introduction to SCPI Command Syntax

SCPI commands are English language based ASCII text strings. The SCPI command set is based on a hierarchical model of a generic instrument. The instrument is broken down into major system elements like OUTPUT, DISPLAY, etc. The command follows a path from major functional elements down through

subsystems, to specific functions within the subsystem. For example to turn on Channel 1's 1 MHz output bandwidth limit filter the command would be:

OUTPut1:FILTer: FREQuency 1E6

The command is shown in its long (or verbose) form. As with all commands described in this manual, the uppercase letters indicate the characters required to represent the short form of the command. Note that SCPI instruments are not case sensitive, the use of capitalization in this manual is only intended to show the difference between the long and short forms of the command.

Note also that the short form and long form are the only acceptable forms of a command. So, for "frequency" we can send "freq" or "frequency" but not "frequ", for example. The short form is the first four letters, unless the fourth is a vowel, in which case the short form is the first three letters.

Keywords are separated by colons, while arguments use a space as a delimiter. Multiple commands can be included in a single multi-element command by using a semi-colon to separate each element. Multiple elements within the same command may be abbreviated if each element is within the same subsystem. The second element in a multi-element command must be preceded with a colon if it is not within the same subsystem. Commands enclosed in square brackets indicate default subsystems. For example, OUTPut1:STAte ON is equivalent to OUTPut1 ON.

These are four valid WaveStation commands under two different subsystems. The WAVE and OUTPut subsystems.

WAVE:SELECT ch1 - Enable channel 1 editor

WAVE:OPEN "new_wave" - Select waveform new_wave

OUTPut1:FILTer:FREQuency 1E6 - Enables the Channel 1 MHz Bandwidth filter

OUTPut1 on - Enables channel 1 output

The above commands may be sent to the WaveStation one command at a time or they may be combined into a single multi-element command. Following are valid forms for a multi-element command. Each element in the command is separated by semi-colon.

```
WAVE:SELECT ch1;OPEN "new_wave"
OUTPut1:FILTer:FREQuency 1E6;:OUTPut1 on
```

Note that when commands are combined using the semicolon they must be at the same level in the command hierarchy. So the second line, in the example above, cannot contain just the argument "on", it requires that the keyword :OUTPut1 be included. An alternative form of the combined command places the commands in hierarchical order and doesn't require a re-statement of the keyword:

```
OUTPut1 on; FILTer:FREQuency 1E6
```

A complete discussion of SCPI command structure is contained in "SCPI 1993, Volume 1:Syntax and Style" available from the SCPI Consortium.

The English nature of SCPI commands often means that a command can directly be mapped to a corresponding menu control. Where standard commands are not available in the 1993 SCPI standard, LeCroy has extended the language to facilitate control of the instrument. Extensions to the language use command names and arguments that adhere to the terminology used in the menu system wherever possible.

Command Subsystems

This section provides a comprehensive overview of the SCPI command subsystems. All command keywords are shown. This section is intended to assist the user in rapidly locating the command form required to carry out AWG actions or query settings and values. Commands with only a query form are shown with a '?' as a suffix. Command arguments are not described in detail in this section. Refer to Section 6 of this manual for details of command arguments and for additional information on the commands.

OUTPut Subsystem

The OUTPut subsystem provides control of the output channel(s), additive noise, and low pass filter bandwidth selections.

Because the instrument may have two channels, the OUTPut subsystem is controlled using OUTPut1 or OUTPut2 in order to uniquely control each of the arbitrary waveform generator's outputs. In this manual, the numeric suffix to the OUTPut subsystem is shown in general form using a # character i.e., OUTPut#:NOISe controls the noise output of either channel.

Overview of OUTPut Commands

OUTPut#	
[STATe]	Enables or disables the output for the specified channel.
FILTer	
[LPASs]	
FREQuency	Sets the bandwidth for the specified channel.
NOISe	
[STATe]	Enables or disables the addition of uncorrelated, pseudo random noise into the specified output channel.
LEVel	Sets the level of noise that is inserted into the waveform for the specified channel.
PATH	INTERNAL or EXTERNAL. EXTERNAL = routed through BNC's on rear. *Note: OUTP1: NOISE:PATH is functionally coupled to OUTP2:NOISE:PATH. Both are either internal or external.*
OUTPut2:RESample	Issues command to resample channel 2 waveform. This command only applies to channel 2.

WAVE Subsystem

The WAVE subsystem controls the selection, creation, editing, and mathematical manipulation of waveforms in the selected waveform editor, channel 1, channel 2, or scratch pad. The operation of the WAVE subsystem is augmented by the FGENerator and EQUation subsystems which handle the specialized operations associated with waveform creation.

Overview of WAVE commands

WAVE	
AMPLitude	
AMPLitude	Sets the peak-to-peak amplitude of the region between the left and right time cursors.
MEDian	Sets the median voltage level of the region between the left and right time cursor.
VMAX	Sets the maximum voltage of the region between the left and right time cursors.
VMIN	Sets the minimum voltage of the region between the left and right time cursors.

WAVE	
CLOCk	
DECade	Selects the clock decade in which the internal clock runs.
FIXed	Selects whether the clock is fixed or variable.
FREQuency	Sets the frequency of the clock.
PREServe	POINTS or TIME. Affects the operation of CLOCK:DECADE. *Preserve points keep data unchanged; preserve time* resamples to keep output timing the same, if possible.
ACSet	Selects auto clock set mode or manual .
LIMit	Selects/deselects option to limit clock to internal filters.
MAX	With LIMit set to Yes, MAX selects the clock decade in which the internal clock runs.
WAVE	
CUT	
COPY	*Copies the region between the right and left time cursors* to the cut buffer.
DELete	Deletes the data between the left and right time cursors, stores it to the cut buffer.
EXTRact	Copies the value of the waveform minus the value of the baseline to the cut buffer.
WAVE	
DATA	Transfer waveform in Data Interchange Format (DIF) to or from host computer.
PREamble	Transfer waveform DIF preamble to or from host computer.
INSert	
MODE	Selects insert or overwrite insertion mode.
PASTe	
[IMMediate]	Inserts the contents of the cut buffer into the waveform.
COUNt	Sets the insert repetition count, i.e. number of times the contents of the cut buffer is inserted into the waveform.
CURSor	Selects if waves are inserted before or after the cursor.
WRAP	Selects if waveform is to be continuous with the last point wrapped to first or if waveform is single shot.

WAVE	
INSert	
SCOPe	
[IMMediate]	Downloads the data from the specified digital oscilloscope (DSO).
ADDRess	Sets the GPIB address of the source DSO.
BWLimit	Select option to check for and correct waveform discontinuities or to not check or correct discontinuities.
CONTrol	Selects the GPIB request control mode for DSO transfers.
PREServe	Sets how the data from the digital oscilloscope is preserved. The data can be preserved in time or by points.
SOURce	Selects waveform source from available DSO traces.
TYPE	Selects DSO type (model).
SHAPe	
DC	
DURation	Set the time duration (length) of the inserted DC function.
LEVel	Set the voltage level of the inserted DC function.
PULSe	
AMPLitude	Sets the base to top amplitude of the standard wave pulse.
BASE	Sets the base voltage level of the pulse.
CYCLes	Sets the number of pulse cycles inserted into the waveform.
ETIMe	The 10%-90% transition time of the rising and falling edges of the standard wave pulse.
PERiod	Sets the period (1/frequency) of the standard wave pulse.
TDELay	Sets time delay from the beginning of the waveform and the beginning of the first edge of the pulse.
WIDTh	Sets the half amplitude width of the standard wave pulse.
RAMP	
AMPLitude	Sets the peak-to-peak amplitude of the standard wave ramp.
CYCLes	Sets the number of cycles of the standard wave ramp inserted into the waveform.
FREQuency	Sets the frequency of the standard wave ramp.
INVert	Controls the polarity of the ramp's slope, i.e. rising or falling.
OFFSet	Sets the voltage of the zero degree phase of the ramp."
SPOSition	Sets the start position of the ramp in percentage of the ramp amplitude.

WAVE	
INSert	
SHAPe	
SELect	Selects which standard wave shape will be inserted into the waveform.
SINE	
AMPLitude	Sets the peak-to-peak amplitude of the standard wave sine.
CYCLes	Sets the number of cycles of the standard wave sine to be inserted into the waveform.
FREQuency	Sets the frequency of the standard wave sine.
OFFSet	Set the voltage of the zero degree phase of the standard wave sine.
PHASe	Sets the start phase of the standard wave sine.
SQUare	
AMPLitude	Sets the base to top amplitude of the square wave.
BASE	Sets the voltage of the base level of the square wave.
CYCLes	Sets the number of cycles of the square wave that will be inserted into the waveform.
ETIMe	Sets the 10%-90% transition time of the rising and falling edges of the square wave.
FREQuency	Sets the frequency of the square wave.
TDELay	Sets the delay time between the start of the waveform and the first edge of the square wave.
TRIangle	
AMPLitude	Sets the peak-to-peak amplitude of the standard triangle wave.
CYCLes	Sets the number of cycles of the triangle wave that will be inserted into the waveform.
FREQuency	Sets the frequency of the triangle wave.
OFFSet	Set the voltage of the base of the triangle.
PHASe	Phase of the triangle wave.
[IMMediate]	Inserts the specified shape at the left time cursor.
WAVE	Insert the named waveform into the current waveform at the TIME LEFT cursor.

WAVE	
MARKer	
CLOCk	
FIRSt	Sets the time at which the first edge of the clock marker begins. WAVE:MARKer:TYPE must be set to CLOCk.
FREQuency	Sets the frequency of the marker clock. WAVE:MARKer:TYPE must be set to CLOCk."
EDGE	
DEFault	Sets default edge marker.
NDEFined	Query only. Number of edges defined.
TIME	Sets the time at which STATE will act.
[STATE]	Low or High.
LEVel	Sets the voltage level of the marker to TTL or ECL levels.
TYPE	Selects either a clock marker or an edge marker.
MATH	
COUPling	AC or DC, used only for INTEGRATION. If DC, integration of a constant non-zero voltage becomes a ramp.
IMMediate	Performs the math function specified by WAVE:MATH[:OPERation] on the current waveform and WAVE:SOURce2 (if applicable) on the region between the left and right time cursors. The result is placed into the current waveform.
SOURce2	Name of the "other" waveform for two waveform operations such as ADD, SUBTRACT, MULTIPLY DIVIDE.
[OPERation]	Specifies which math operation will be performed by WAVE:MATH :IMMedate. Operation can be SMOOTH, ADD, SUBTract, MULTiply, DIVide, INTegrate DIFFerentiate CONVolve.
NEW	Creates a new waveform with the name specified by the argument.
OPEN	Opens a waveform from the current project.
REGion	
LEFT	Set the position of the left time cursor.
RIGHt	Set the position of the right time cursor. This command requires time cursors not to be in the track mode.

WAVE	
SAVE	Saves the current waveform with the name supplied by the argument.
SELect	Selects the active waveform editor CH1, CH2, or SCR.
TIME	
DELay	Delays the waveform from the left cursor to the end of the waveform for the given amount of time.
DURation	
MODE	Selects the mode, insert or overwrite, for changing the duration of a feature.
[TIME]	Changes the duration of the region between the left and right time cursors using the duration change mode defined by the duration modes.
MOVE	Moves the feature between the left and right time cursors.
SEQuence	
ADVance	Advance to the next sequence in a group sequence list.
AON	Specifies which channel advance and jump operate on.
COMPile	Cause the desired sequence to play.
Data	Tansfers a sequence file identified by a filename to or from the WaveStation via GPIB in #0 blobk format.
GDATa	Transfers a group sequence file to or from the AWG via GPIB in #0 block format.
GLINk	Add a new sequence to the end of the sequence list in the currently selected group sequence.
GNEW	Creates a new group sequence.
IRECall	Recall a saved image file.
ISAVe	Save a binary image of the hardware to a file.
JUMP	Jump to the nth sequence in the list.
LINK	Add on entry to the end of the sequence list in memory.
NEW	Empty the sequence list, associate a new name with sequence list.
OPEN	Open and compile a sequence file from the project.
SAVE	Save the sequence list from memory to the current project.

FGENerator Subsystem

The WaveStation's standard function generator mode is controlled by the FGENerator subsystem. Any of the seven standard waveforms, sine, triangle, square, ramp, pulse, multitone, and DC can be specified. Key parameters such as frequency, amplitude, offset, and start phase can be controlled directly. Additionally, the frequency of the sine, triangle, square, ramp and pulse waveforms can be swept linearly or logarithmically.

Overview of FGEN Commands

FGENerator#	
DC	
LEVel	Set the DC voltage level for the specified channel's function generator (either 1 or 2).
MULTitone	
AMPLitude	Sets the peak-to-peak amplitude of the multitone function in the specified channel's function generator (either 1 or 2).
NTONes	Sets the number of tones to be calculated for the multitone function.
OFFSet	Set the voltage of the zero degree phase of the multitone waveform.
TONE#	
RAMPlitude	Sets the relative amplitude of the current tone in the multitone waveform.
[FREQuency]	Set the frequency of the current tone in the multitone waveform.
PULSe	
AMPLitude	Sets the base to top amplitude of the pulse in the specified channel's function generator (either 1 or 2).
BASE	Sets the voltage of the base level of the pulse waveform in the specified channel's function generator (either 1 or 2).
ETIMe	Sets the 10%-90% edge time of both the rising and falling edges of the pulse waveform.
PERiod	Sets the period (1/frequency) of the pulse in the specified channel's function generator (either 1 or 2).
SWEep	
SPACing	Selects the type of sweep (either linear or log) in the specified channel's function generator (either 1 or 2).
STARt	Sets the start frequency of the sweep.
STOP	Sets the stop frequency of the sweep .
TIME	Sets the sweep duration.
[STATe]	Turns the sweep on or off.

FGENerator#	
PULSe	
TDELay	Sets the amount of time between the beginning of the waveform and the beginning of the first edge of the pulse.
WIDTh	Sets the width of the pulse from 50% up the rising edge to 50% down the falling edge.
RAMP	
AMPLitude	Sets the peak-to-peak amplitude of the ramp in the specified channel's function generator (either 1 or 2).
FREQuency	Sets the frequency of the ramp.
INVert	Controls whether the ramp is rising or falling.
OFFSet	Set the median voltage of the ramp waveform.
SPOSition	Sets the start position of the ramp in percentage of the ramp's peak-to-peak amplitude.
SWEep	
STARt	Sets the start frequency of the sweep.
STOP	Sets the stop frequency of the sweep.
TIME	Sets the sweep duration.
[STATe]	Turns the sweep on or off .
SELect	Selects which function the specified channel's function generator outputs. The available functions are: SINE, TRIangle, SQUare, RAMP, PULSe, MULTitone, and DC.
SINE	
AMPLitude	Sets the peak-to-peak amplitude of the sine wave in the specified channel's function generator (either 1 or 2).
FREQuency	Sets the frequency of the sine wave.
OFFSet	Sets the voltage of the zero degree phase of the sine waveform.
PHASe	Sets the start phase of the sine wave.
SWEep	
SPACing	Selects the sweep type (either linear or log).
STARt	Sets the start frequency of the sweep.
STOP	Sets the stop frequency of the sweep.
TIME	Sets the sweep duration.
[STATe]	Turns the sweep on or off.

FGENerator#	
SQUare	
AMPLitude	Sets the peak-to-peak amplitude of the square wave in the specified channel's function generator (either 1 or 2).
BASE	Sets the voltage of the base level of the square wave.
ETIMe	Sets the 10%-90% edge time of both the rising and falling edges of the square wave.
FREQuency	Sets the frequency of the square wave.
SWEep	
SPACing	Selects the sweep type (either linear or log).
STARt	Sets the start frequency of the sweep.
STOP	Sets the stop frequency of the sweep.
TIME	Sets the sweep duration.
[STATe]	Turns the sweep on or off .
TDELay	Sets the amount of time between the start of the waveform and the first edge of the square wave. Useful in single trigger mode; in continuous this time lowers the frequency.
TRIangle	
AMPLitude	Sets the peak-to-peak amplitude of the triangle wave in the specified channel's function generator (either 1 or 2).
FREQuency	Sets the frequency of the triangle wave.
OFFSet	Sets the median voltage of the triangle waveform.
PHASe	Sets the start phase of the triangle wave.
SPACing	Selects the sweep type (either linear or log).
SWEep	
STARt	Sets the start frequency of the sweep.
STOP	Sets the stop frequency of the sweep.
TIME	Sets the sweep duration.
[STATe]	Turns the sweep on or off.
[STATe]	Turns the function generator on or off in the specified channel (either 1 or 2).

EQUation Subsystem — The equation subsystem is used to enter, select, save, and recall equations which describe waveforms mathematically. It is also used to calculate the waveform sample values based on the equation.

Overview of EQUation Commands

EQUation	
CALCulate	Calculates the currently specified equation line for the preset duration and inserts it into the current waveform at the left cursor position using the current insert mode.
DATA	Transfers all the lines of the equation sheet as a "#0" block. #0 is an indefinite length block of data terminated with EOI. Defined in IEEE 488.2.
DEFine	Defines an equation for the current equation line. The equation line may be up to 50 characters in length and must be surrounded by quotes. Valid functions are: SIN, COS, SQRT,PULSE, STEP, LN, LOG, ABS, EXP and TAN. Valid operators are: +, -, *, /, (,), "","", = and ^. Valid variable names are X1 through X16. Valid arguments are T, PI, NOISE, and GNOISE.
DURation	Sets the time span over which the equation will be calculated.
LINE	Selects an equation line from the current equation sheet.
NEW	Creates a new equation sheet.
OPEN	Opens an existing equation sheet.
SAVE	Saves the current equation sheet.

DISPlay Subsystem The DISPlay subsystem controls the selection and presentation of text, graphics and waveform information. In addition, the cursor system is controlled by this subsystem.

Overview of DISPlay Commands

DISPlay	
ANNotation	
DATE[:STATe]	Allows the time/date annotation field to be switched on or off.
LOGO[:STATe]	Allows the Company Logo to be switched on or off.
PARameter[:STATe]	Turns the parameters readouts on or off.
[ALL]	For SCPI compatibility. Same as "Logo".
SSAVe	Allows the automatic screen saver to be enabled or disabled.
[WINDow]	
TRACe	
ALL	Displays the whole waveform on the screen.
COLor	Sets the trace intensity. Setting the intensity for one trace will set the same intensity for all traces.
CURSors	
TIME	
DELTa	Change the delta time between the time cursors. This command only has effect if the cursors are in the track mode.
LEFT	Set the position of the left time cursor.
RIGHt	Set the position of the right time cursor. This command only has effect if the cursor track mode is off.
SALL	Select All selects the entire waveform by placing the left cursor at time zero and the right cursor at the end of the waveform.
TEND	Places both cursors at the end of the waveform.
TGRid	Moves both time cursors so they are on the display.
TRACk	Enables or disables time cursor tracking.
[STATe]	Turns the time cursors on or off.
VOLTage	
BOTTom	Set the position of the bottom voltage cursor.
DELTa	Change the delta voltage between the voltage cursors. This command only has effect if the voltage cursors are in the track mode.
TGRid	Moves both voltage cursors so they are on the display.

```
DISPlay
    [WINDow]
        TRACe
            VOLTage
                TOP          Sets the position of the top voltage cursor. This command
                             only has effect if track is off.
                TRACk        Enables or disables voltage cursor tracking.
                [STATe]      Turns the voltage cursors on or off.

            GRATicule
                COLor        Set the display intensity for the grid.
                GRID
                    [STATe]  Select or query the grid style. The grid may be a full grid
                             (ON), no grid (OFF), or set to a cross hair (CHAir).
                TYPE         Selects the type of grid to display. Single, dual, SXY, XY.

            TRACe
                X[:SCALe]
                    CENTer   Sets the time at the horizontal center of the grid.
                    PDIVision Sets the horizontal time per division of the grid.
                    TCURsors Displays the portion of the waveform between the time
                             cursors with the left cursor one division from the left edge of
                             the grid and the right cursor one division from the right edge
                             of the grid.
                Y[:SCALe]
                    PDIVision Sets the vertical volts per division of the grid.
                    RLEVel   Sets the voltage at the vertical center of the grid.

                ZPRevious    Restores the zoom settings to the previous time and voltage
                             zoom settings.
```

HCOPy Subsystem The HCOpy subsystem provides control over printing and output of screen graphics form the WaveStation.

Overview of HCOPy Commands

HCOPy	
AUToincr	Enables automatic increment of the filename index when a hardcopy is stored to a file.
FILename	Set or query the current hardcopy file name.
INDex	Set the hardcopy filename index number. The index may range from 0 to 999.
TARGet	
GRAPhics	
DESTination	Set the destination for the hardcopy graphics file.
FORMat	Set the hardcopy graphics file format.
PRINter	
DESTination	Set the destination of the hardcopy printer data. The destination may be the GPIB or Centronics port, or it may be the floppy disk drive where a file in printer format will be stored.
FFEed	Set whether a form feed is automatically generated following a hardcopy.
MODel	Set the specified printer model.
QUALity	Set the print quality, draft or proof. This setting is not available for all supported printers.
SIZE	Set the size of the hardcopy, notebook or presentation.
TYPE	Sets the hardcopy format. Hardcopies may be formatted to provide data suitable for printers or graphics files.
[IMMediate]	Begin a hardcopy.

TRIGger Subsystem

The trigger subsystem is used to control the Trigger section of the AWG. This includes controls for triggering such as level, mode, source and slope.

Overview of TRIGger Commands

INITiate [:IMMediate]	Triggers the system, equivalent to the IEEE 488.2 command *TRG.
TRIGger[:SEQuence]	
BCOunt	Sets the burst count or number of repetitions of the waveform that will be output after a trigger is received in burst mode.
DELay	Sets the delay from trigger to start of output of the waveform.
LEVel	Sets the trigger level in volts.
MODE	Sets the trigger mode. The trigger mode may set to CONTinuous, SINGle, BURSt, or GATE.
SLOPe	Sets the trigger slope.
SOURce	Sets the trigger source. The trigger source may internal or external.

MMEMory Subsystem

The MMEMory (mass memory) subsystem provides support for the extensive hard disk storage capability of the WaveStation.

Overview of MMEMory Commands

MMEMory	
CATalog	
EQUation	Returns a list of all equations in the current project.
IMAGe?	Returns a listing of image files located in the current project
SEQuence	Returns a list of all sequences in the current project.
WAVeform	Returns a list of all waveforms in the current project.
[ALL]	Returns a list of all objects in the current project.
.	
DATA	Upload or download the waveform named in the associated argument. Waveforms are stored in DIF format.
PREamble	Upload or download the header of the waveform named in the associated argument.
DELete	
EQUation	Deletes the named equation.
IMAGe	Deletes the named image.
PROJect	Deletes the named project.
SEQuence	Deletes the named sequence.
[WAVeform]	Deletes the named waveform.

PROJect Subsystem The project subsystem is used to create, open, and save individual user work areas called projects.

Overview of PROJect Commands

PROJect	
NEW	Creates a new project with the specified name. The current project is closed and the new project is created.
OPEN	Opens the specified project if it exists (no action is taken if it doesn't exist) and closes current project.
SAVE	Saves the current project.

SYSTem Subsystem Provides controls not specific to the vertical, horizontal, trigger, or measurement subsystems.

Overview of SYSTem Commands

SYSTem	
CLOCk	
EREFerence	Sets whether the system uses the internal clock reference or an external 10 MHz clock reference.
COMMunicate	
GPIB[:SELF]	
ADDRess	Sets the GPIB address of the AWG.
ERRor?	Query the last three system errors. The result of the query is the error number followed by the error text for each of the last three system errors.
HELP	
SYNTax?	Finds out the arguments for and full form of a header. Example, SYST:HELP:SYNTAX? "WAVE:OPEN".
VERSion?	Returns SCPI version number for which instrument complies.

CALibration Subsystem

CALibration[:ALL]?	Performs an Internal calibration and returns a status code indicating if the calibration was successful: 0 = Calibration successful 1 = Calibration failed

STATus Subsystem

The status Subsystem is used to control the status reporting registers. This includes the 488.2 specified condition, event and enable registers as well as the SCPI defined QUEStionable and OPERation registers. There are two event status registers, the Status Byte Register (STB) and the Standard Event Status Register (ESR) within the WaveStation. There are also two dual purpose (event and condition) registers: the OPERation Status Register and the QUEStionable Status Register. Finally there is an Error/Event queue that records the last error. For full information on the Status Registers, please refer to Section 4 of this manual.

Overview of STATus Commands

STATus	
OPERation	
CONDition?	Query the Operation Status Condition Register.
ENABle	Enable bits in the Operation Status Event Register that will be summarized in the Status Byte Register.
[EVENt]?	Query the contents of the Operation Status Event register.
PRESet	Clear all status registers and clear all enable registers. Sets enable registers to the same as power on conditions.
QUEStionable	
CONDition?	Query the Questionable Status Condition Register.
ENABle	Enable bits in the Questionable Status Event Register that will be summarized in the Status Byte Register.
[EVENt]?	Query the Questionable Status Event Register.

488.2 Common Commands

In addition to the SCPI subsystems, 488.2 mandatory are supported by the WaveStation. Following is a brief listing of the standard 488.2 commands. The 488.2 commands work in combination with the SCPI commands to provide full control of the WaveStation.

*CAL?	Performs a system calibration and returns a status code indicating if the calibration was successful: 0 = Calibration successful 1 = Calibration failed
*CLS	Clears all status registers.
*ESE	Enable bits in the Event Status Register.
*ESR?	Reads and clears the contents of the Event Status Register.
*IDN?	Identifies the instrument. The response indicates the manufacturer, the model, the serial number and the software revision level.
*LRN?	Read the current instrument setup.
*OPC?	When overlapped operations are complete place a `1' into the output queue.
*OPC	When overlapped operations are complete assert the OPC bit in the EVENT STATUS register.
*PCB	Identifies the address to pass control back to when the WaveStation is about to be given control of the GPIB bus.
*RST	Sets all settings (I/O and Scope setup) to their default values.
*SRE	Enable bits in the Service Request Enable mask.
*STB?	Read the contents of the main status byte.
*TRG	Same as the manual button on the Trigger menu.
*TST	Performs selftest and returns a status code indicating if selftest was successful: 0 = success.
*WAI	WAIT for completion of overlapped operations before parsing more commands. The operations under WAVE:TIME, and SEQ:COMP.LC are overlapped operations.

This page left intentionally blank

4 STATUS & ERROR REPORTING

Status Register

A set of status registers allows the user to quickly determine the AWG's internal processing status at any time. The status registers as well as the status and event reporting system adhere to the SCPI recommendations.

Status Byte Operation

The WaveStation continually updates its status to report the latest events, conditions, and settings.

Changes are summarized by designated bits in the Status Byte register (STB). The seventh bit, RQS, is asserted whenever any other bits in the STB are reported as set and their corresponding enable bits are set. Also, whenever the RQS bit is set, the GPIB bus SRQ line is automatically asserted.

Status Data Structures

In general, an asserted bit in the main status byte (STB) reflects, or summarizes, a change in a corresponding status register or queue (i.e. Standard Event Status Register, Questionable Status Register, Operation Status Register, or Error/Event Queue).

Two types of status structures, the Register (individual bits) and the Queue (encoded number), are used in the WaveStation.

Register Model

In the Register Model individual bits identify a specific WaveStation condition or event.

Alternatively, each bit could act as a summary bit for an associated status register. Using bits in one status register to indicate changes in other registers allows for a layered status description. This layering of detail enables the controller to limit the amount of information it receives. The Status Byte Register, Standard Event Status Register, Questionable Status Register, and Operation Status Register all use the register model status structure.

Queue Model

The Queue Model is a single register which contains an encoded number. For example, this number may be an error code which corresponds to an error condition.

The WaveStation's Error/Event Queue is the only register in the WaveStation employing the queue model. The Error/Event Queue can hold one error code. When read, the queue reports the most recent error code, and clears itself.

When the queue is cleared (empty), the corresponding bit in the Status Byte Register will be cleared. Conversely, when the queue contains an error code, the corresponding bit in the Status Byte Register will be set.

Event Recording

IEEE-488.2 allows two ways to record an event and the WaveStation registers are implemented as both condition and event registers to provide full functionality. The names of the condition and event registers are the same. Only the commands to query the event and condition registers differ.

Condition Registers

Condition Registers are updated continually and are not cleared when read. If a condition was true but is no longer true the corresponding bit in the condition register will be false. The WaveStation has only two condition registers, the Questionable Status Register and the Operational Status Register. These two registers also function as event registers. Whether the condition or event register is queried depends on the form of the query used.

Event Registers

Event Registers capture changes in conditions. They are not cleared until they are read, even if the condition which caused the event no longer exists. All registers in the WaveStation function as event registers. The Questionable Status Register and Operational Status Register function as both event and condition registers depending on how they are queried. Each bit in an Event Register either summarizes an event register, or reports a condition or event in the WaveStation. A bit is set to true (1) when the summary, condition, or event changes from false (0) to true (1) and will remain set until cleared using the *CLS command or by reading the register.

Querying the Operational and Questionable Status Register

Since the Operational Status Register and the Questionable Status Register can be both condition and event registers depending on the query form the query form is very important. To read the Operational and Questionable Event Registers use the following commands:

STATus:OPERation? - Read Operation Status Event Register.

STATus:QUEStionable? - Read Questionable Status Event Register.

To read the Operation and Questionable Condition Registers use the following commands:

STATus:OPERation:CONDition? - Read Operation Status Condition Register.

STATus:QUEStionable:CONDition? - Read Questionable Status Condition Register.

The following example illustrates how the condition and event registers can return different values.

The waiting for trigger status is shown in bit 5 of the Operation Status register. (The bit meaning of each bit in each register is documented later in this section.)

While the WaveStation is waiting for a trigger, the commands STATus:OPERation? and STATus:OPERation:CONDition? return the same value for bit 5. Both commands return true (32) because the WaveStation is waiting for a trigger.

If both commands are issued again, while the WaveStation is still waiting for a trigger, the results will be different. The command STATus:OPERation? will return false (0) because it was cleared when the event register was read with the command above. The command STATus:OPERation:CONDition? will return true (1)

because it was not cleared when read and the WaveStation is still waiting for a trigger.

When the waveform is being generated, the command STATus:OPERation? will return false (0) because the event register was read and cleared the first time the command was sent. The command STATus:OPERation:CONDition? will return false (0) because the WaveStation is not waiting for a trigger..

If the WaveStation was waiting for a trigger, receives a trigger and we send the query STATus:OPERation? While the waveform is being generated, then this query will return true (32) because the event of waiting for trigger has occurred since the event register was last cleared. The query STATus:OPERation:COND? Will return false (0) because the WaveStation is not currently waiting for a trigger.

Event Enable Registers

The WaveStation registers are arranged in a tree like structure. The Status Byte Register is the root of the structure and branches out to summarize the Standard Event Status Register, the Operation Status Registers, the Questionable Status Register, and the Error/Event Queue. Coupled with each event register is an Enable Register. The Enable Registers determine which if any bits of the associated Event Register will be summarized in the Status Byte Register.

Each bit in an event enable register is "AND'ed" with its corresponding bit in its associated status event register. If the result of the AND operation is a one (true) the summary bit will be set in the Status Byte Register.

All event registers are edge sensitive, meaning they are set when the status changes state. The SCPI standard allows for choosing the edge of interest (positive going or negative going), but this capability is not implemented in the WaveStation. The WaveStation will set the bit in the status register to true (1) whenever the status changes from false (0) to true (1). Event register bits are set on a positive going transition.

The status registers and enable registers are associated as follows:

Status Byte Register	Service Request Enable Register
Standard Event Status Register	Event Status Enable Register
Operation Status Register	Operation Status Enable Register
Questionable Status Register	Questionable Status Enable Register

The following commands are used to set the value of the enable registers:

*SRE	Service Request Enable Register
*ESE	Event Status Enable Register
STATus:OPERation:ENABle	Operation Status Enable Register
STATus:QUEStionable:ENABle	Questionable Status Enable Register

The enable registers for the Operation Status Register and the Questionable Status Register are 15 bits wide with each bit selecting a different condition or event. The enable registers for the Service Request Register and the Event Status Register are 7 bits wide with each bit selecting a different condition or event. The bit positions for the enable register match the bit positions for the status registers and have the same names. While the Operation Status Register and the Questionable Status Register can function as both event and condition registers, only the results of the event register are AND'ed with the enable register to set the summary bit in the Status Byte Register.

The value of the Enable registers may also be changed to a preset value with the STATus:PREset command. STATus:PREset clears the Operation and Questionable Enable registers. Refer to command details for STATus:PREset for the further information. During power-on the enable registers are set to their STATus:PREset states. The *RST and *CLS commands have no effect on the enable registers.

Status Byte Register Definition

The main Status Byte register (STB) reflects instrument status at the time it is read. The register is read when the system controller (remote computer) polls the WaveStation with the *STB? command or with a serial poll. Bits in the STB summarize all the other status registers.

*The STB is read with the command *STB? or by serial polling the WaveStation. The Status Byte Register's enable register is set with *SRE n. The Status Byte Enable Register is read with *SRE?. (Note: n is the sum of the decimal bit weights of all bits that are true.)*

The *STB? query does not alter any bits in the status byte. Only the *CLS command can clear the status byte, except for the MAV Message Available but which depends on the state of the output queue.

Status Byte Register Definition

Bit#	Associated Status Register	Significance
7 (MSB)	Operation Status Register	Summarizes Operation Status Register
6	none	RQS (service request) Bit
5	Standard Event Status Register	Summarizes Standard Event
4	MAV	Message Available
3	Questionable Status Register	Summarizes Questionable Status Register
2	Error/Event Queue	Error/Event Bit
1	none	Not Used
0 (LSB)	none	Not Used

Bit 0: Not Used This bit is not used by the WaveStation and has no significance.

Bit 1: Not Used This bit is not used by the WaveStation and has no significance

Bit 2: Error/Event Queue Bit The Error/Event Queue can hold three error codes. When the queue contains an error code, bit 2 is true (1). When the queue is cleared (empty), the corresponding bit 2 is false (0). This bit will sense that an error has occurred. To read the error code from the Error/Event Queue the queue must be read using the SYSTem:ERRor? command.

Bit 3: Questionable Status Summary Bit

If this bit is true (1) it indicates that an event has caused one of the enabled bits in the Questionable Status register to become true. To determine the reason that caused the questionable status query the Questionable Status Register using the STATus:QUEStionable? command. Further documentation is available in the section on the Questionable Status Register.

Bit 4: MAV - Message Available Bit

MAV is set if data is in the output queue. It is reset once the output queue is empty. This condition bit is not set or reset when the system controller reads STB. Also, the *CLS command does not affect this bit.

Bit 5: Standard Event Status Summary Bit

The ESB is set if one of the bits in the ESR which is enabled in the ESE becomes set. This bit summarizes the Event Status Register (*ESR).
The *ESR identifies the type of event. Since the *ESR is an Event Register, any bits stay set until the register is read. After it is read, all the bits are cleared. Once cleared, its summary bit (bit 5) in the STB is also cleared.

*ESR's event enable register, or mask is *ESE. To set the *ESE use *ESE n, and to read it use *ESE?. The command to read the *ESR is *ESR?.

Further documentation is available in the section on the Standard Event Status Register.

Bit 6: RQS - Request Service Bit

The RQS bit is the summary bit for the other bits in the STB byte. For GPIB, an SRQ interrupt is generated when the RQS bit is set. The RQS bit is set when a bit in the STB is set and the corresponding bit in the Status Byte Enable Register (SRE) is set.

Bit 7: Operation Status Summary Bit

If this bit is true (1) it indicates that an event has caused one of the bits in the Operation Status register to become true. In the WaveStation this indicates that the WaveStation is waiting for a trigger. To determine what caused the Operation Status bit to be set, query the Operation Status Register using the STATus:OPERation? command. Further documentation is available in the section on the Question Status Register.

Standard Event Status

Register Definition The Standard Event Status Register reports error conditions common to most automatic test equipment. The WaveStation uses these bits for error reporting and synchronization. The Standard Event Status Register is read and cleared using the *ESR? command. The register may also be cleared without being read using the *CLS command. Each of the bits in the Event Status Register will be summarized in bit 5 of the Status Byte Register provide the bits are set in the Event Status Enable register. For example to have only the operation complete bit of the Event Status Register summarized in the Status Byte register using the following command to enable only the operation complete bit (bit 0):

*ESE 1 - where 1 is the decimal value when bit 0 is set (true) and all other bits are not set (false).

Event Status Register Bit Assignments

BIT #	Associated Status Byte	Significance
7	none	Power On
6	none	User Request
5	none	Command Error
4	none	Execution Error
3	none	Device Specific Error
2	none	Query Error
1	none	Request Control
0	none	Operation Complete

Bit 0: Operation Complete This bit is set upon completion of any operation.

Bit 1: Request Control This bit is set by the WaveStation as part of the 488.2 REQUSTCLTL protocol. The WaveStation becomes the controller in order to get data from a digital oscilloscope. If WAVE:INSert:SCOPe:CONTrol is set to ON, the WaveStation will request control, and pass control back when it is done. The controller must be capable of supporting IEEE Std. 488.2-1992 pass control protocol.

Bit 2: Query Error

This bit indicates that an error occurred in the last query. Typical errors include: input and output buffers full, unterminated query (controller reads before sending a complete query message), interrupted query (controller sends new command before reading last query)

Bit 3: Device Specific

This bit indicates an error which is not related to the execution of commands.

Bit 4: Execution Error

If the Execution Error Bit is set, a command was sent with an invalid parameter.

Bit 5: Command Error

If the Command Error Bit is set, a command parsing error has occurred.

Bit 6: User Request

The User Request bit is set when the WaveStation is being remotely controlled using the GPIB bus and the hardcopy destination is GPIB and a hardcopy is requested via the front panel. In this case, if the Hardcopy were to start, the WaveStation would enter Talk-only mode and disrupt the remote control connection. To prevent this, the User Request bit is set allowing the remote host to detect the hardcopy request and initiate it remotely after first setting up all connected devices. Please refer to the section on Interface Configuration for more information.

Bit 7: Power On

This event bit indicates that an off-to-on transition has occurred in the WaveStation.

Operation Status

Register Definition The Operation Status Register reports conditions which are part of the instrument's normal operation. The Operation Status Event Register is read and cleared using the STATus:OPERation? command. The event register may also be cleared without being read using the *CLS command. The Operation Status Condition Register is read using the STATus:OPERation:CONDition? command. Each of the bits in the Operation Status Event Register will be summarized in bit 7 of the Status Byte Register provide the bits are set in the Event Status Enable register. For example, to have only the Waiting for Trigger bit of the Operation Status Register summarized in the Status Byte register, use the following command to enable only the operation complete bit (bit 5):

STATus:OPERation:ENABle 32- where 32 is the decimal value when bit 5 is set (true) and all other bits are not set (false).

Operation Status Register Bit Assignments

BIT #	Associated Status Byte	Significance
14	none	Not Used
13	none	Not Used
12	none	Resample Channel 2
11	none	Not Used
10	none	Sequence Compile Complete
9	none	Reserved for future use
8	none	Reserved for future use
7	none	Not Used
6	none	Not Used
5	none	Waiting for Trigger
4	none	Not Used
3	none	Not Used
2	none	Not Used
1	none	Not Used
0	none	Not Used

Bit 5: Waiting for Trigger This bit is set when the WaveStation is in a triggered mode and is waiting for a trigger.

Bit 10: Sequence Compile Complete Set when a sequence has finished compilation.

Bit 12: Resample Channel 2 This bit is set when an operation is performed that requires resampling of channel 2.

Questionable Status Register Definition

The Questionable Status Register contains bits which give an indication of the quality of various aspects of a signal or measurement. Since the WaveStation does not acquire data and make measurements, these bits are not used by the WaveStation. The Questionable Event Status Register is read and cleared using the STATus:QUEStionable? command. The event register may also be cleared without being read using the *CLS command. The Questionable Condition Status Register is read using the STATus:QUEStionable:CONDition? command. Each of the bits in the Questionable Event Status Register will be summarized in bit 3 of the Status Byte Register provided the bits are set in the Questionable Status Enable register.

For example, to have only the command warning bit of the Questionable Event Status Register summarized in the Status Byte register, use the following command to enable only the measurement bit (bit 14):

STATus:QUEStionable:ENABle 16384 - where 16384 is the decimal value when bit 14 is set (true) and all other bits are not set (false).

Questionable Status Register Bit Assignments

BIT #	Associated Status Byte	Significance
14	none	Command Warning
13	none	Not Used
12	none	Not Used
11	none	Not Used
10	none	Not Used
9	none	Not Used
8	none	Not Used
7	none	Not Used
6	none	Not Used
5	none	Not Used
4	none	Not Used
3	none	Not Used
2	none	Not Used
1	none	Not Used
0	none	Not Used

Bit 14: Command Warning At this time the WaveStation does not set this bit.

Checking Status and Requesting Service

There are two basic methods for checking the status of the WaveStation. The first is by polling the status registers in the WaveStation to check status. The second is by having the WaveStation assert the SRQ line on the GPIB bus to indicated that a status condition has been met. The second method is know as requesting service and is only available using the GPIB bus.

Polling to Check Status

Polling is the process of repeatedly querying the status register until a bit changes reflecting a change in state. The simplest method of polling is to poll the single register of interest. For example, to poll to see if the WaveStation is waiting for a trigger the following command would be sent to the WaveStation until a value with bit 5 set (containing a 32) is returned.

STATus:OPERation?

Or

STATus:OPERation:CONDition?

Note: If using the STATus:OPERation? command it is important to clear the register before using it since once it is set it will remain set until cleared.

Another method of polling is to poll the Status Byte Register with the *STB? command and enable Operation Status register to be reflected in the Status Byte Register bit 7. To use this method the Operation Status Enable bits must be set and then the Status Byte Register is polled. The steps are as follows:

STATus:PREset	clear all status registers set all enable registers to 0 (everything disabled)
STATus:OPERation:Enable 32	- Enable Waiting for Trigger Bit
*CLS	- Clear all Registers
*STB?	- Poll to check for bit 7 (decimal 64).

Note: All registers should be cleared before starting the next operation, but there is no need to re-enable the Operation Register. The Operation Register bit 5 (decimal 32) will remain enabled until altered with the :ENABle or :PREset command.

The *STB? command may also show that other bits are set as well as bit 7. For example, bit 6 will also be set because it summarizes all the other bits in the register. It is possible to check only for bit 6 and then if bit 6 is set check for other bits of interest. To check for a single bit in the register AND the *STB? results with the decimal value of the bit and test to see if the result is greater than 0.

Hint: In the C programming language this can be done with the following test:

```
If (STB_result & 32)
  {
/* RQS bit is set */
/* take action here */
          }
        else
          {
/* RQS bit is not set */
/* take action here */
  }
```

In the QBASIC programming language the AND operation can be done with the following test:

```
IF (STB_result AND 32) THEN
; RQS bit is set
; take action here
ELSE
; RQS bit is not set
; take action here
END IF
```

GPIB Service Request

When the WaveStation reports a change in its condition, it can asynchronously request service from the GPIB controller (for example when a measurement is questionable). The WaveStation requests service asynchronously by asserting the GPIB Service Request (SRQ) bus line.

To identify the source of the SRQ, the controller serial polls the devices attached to the GPIB and reads the main Status Byte register (STB) of each device polled. To read the STB, the controller sends the device a Serial Poll bus command. In return the device sends its STB. The device whose STB has an asserted RQS bit (seventh bit) generated the SRQ.

Serial polling the device will clear the SRQ line but the serial poll must be followed by sending the *CLS message to the device to fully clear the status that caused the SRQ to be generated. The *CLS command does not have to be sent immediately following the serial poll but **MUST** be sent before waiting for the next SRQ.

The commands to generate an SRQ when the WaveStation is waiting for trigger are as follows:

```
STATus:PREset                    - Set QUEStionable and
                                 - OPERation enable
                                 - registers to 0
                                 - (everything disabled)
STATus:OPERation:ENABle 32       - Enable bit 5, waiting for trigger
*SRE 128                         - Enable SRQ, bit 6 (RQS)
                                 - Enable operation summary,
                                 - bit 7.
```

When the WaveStation is waiting for a trigger, the SRQ line on the GPIB bus will be asserted. When the SRQ is asserted it must be serviced with a serial poll.

```
*CLS                             - Clear all status registers following
                                 - serial poll
```

The commands to fully setup and service the SRQ using the National Instruments IBIC program are as follows. The IBIC program is provided with all National Instruments GPIB boards, but does require a GPIB board. Please refer to the Interface Configuration Section of this manual or to the National Instruments GPIB manual for additional information on the IBIC program.

CD \GPIB-PC	Change to the National Instrument GPIB-PC subdirectory.
IBIC	Start the IBIC program
IBFIND dev1	Set the GPIB address to 1, the WaveStation address.
IBWRT "*IDN?"	Ask for WaveStation Identification to check communications.
IBRD 100	Read back id. If ID does not return please refer to Interface Configuration Section of this manual for possible problems. DO NOT CONTINUE if identification is not returned.
IBWRT "STATus:PREset	Set QUEStionable and OPERation enable registers to 0 (everything disabled)
IBWRT "STATus:OPER:ENA 32"	Enable bit 5, measurements
IBWRT "*SRE 128"	Enable SRQ
IBWAIT RQS	Wait for SRQ.

Note: The computer will wait infinitely here until the WaveStation asserts SRQ. If it never does the computer will wait forever. To have the computer wait for a SRQ or a time-out send the following command: IBWAIT (TIMO RQS)

IBRSP - Serial poll the bus

Note: The serial poll will return one byte on data. This is the status byte. The status byte can be checked to see which bits were set. This is particularly useful if several conditions could have caused the SRQ.

IBWRT "*CLS" - clear all registers

This page left intentionally blank

Introduction

Waveforms can be transferred between the host computer and the WaveStation via GPIB. The WaveStation stores waveform internally using the standard Data Interchange Format or DIF. This format is fully documented in Volume 3 of the Standard Commands For Programmable Instruments (SCPI) manual, 1993. Waveforms transferred from a host computer to the WaveStation must be in this format. Waveforms exported from the WaveStation to floppy disk, in WaveStation format, are stored in a compressed form and cannot be transferred directly back to the WaveStation via GPIB.

Transferring Waveforms Via GPIB

Waveforms can be read from the WaveStation using the GPIB command query:

WAVE:DATA?

The response will be a data block containing the currently selected waveform in the Data Interchange Format (DIF).

A DIF file can be sent to the WaveStation using the command:

WAVE:DATA <block>

where the data block is the DIF filename.

The Data Interchange Format (DIF)

An ASCII printout of a typical DIF file is shown below. Please note that the actual file would be output as one continuous record, without line feeds. New lines have been inserted for readability. The preamble, which is ASCII readable describes the waveform and all the necessary AWG setup parameters. The waveform data is included in the data array as a series of IEEE 32 bit, single precision, floating point numbers. The waveform data is not in an ASCII compatible format and is not printed in this example.

WaveStation waveform files contain two DIF expressions, as shown and explained below.

```
(DIF (VERSion 1993.0)
IDENtify( NAME "NEW_WAVE"
PROJect "USER" )
ENCode( FORMat IFP32 HRANGE 0.500000 LRANGE
-0.500000 )
DIMension = Volts ( TYPE EXPLicit
SIZE 64
UNITs "V"
ANALog 0)
DIMension = Time ( TYPE IMPLicit
SCALe 2.5e-009
OFFSet 0
UNITs "s" )
TRACe = Cursors_include (LABEL Time
STARt 0
STOP 1.5999e-006)
DATA = data_array ( CURVe ( VALues #3256
.....
.....
.....) ) )
(DIF (VERSion 1993.0)
DIMension = Time ( TYPE EXPLicit
SIZE 2 )
DIMension = Polarity (TYPE EXPLicit
UNITs "TTL")
ORDer (BY TUPLe
DATA = markers ( CURVe ( VALues 0.000000e+000,1,
8.00000e-008,  0) ) )
```

DIF Preamble The DIF preamble consists of the following major blocks:

DIF - Identifies the file as a DIF file and contains the version of the DIF standard, 1993 in this case.

IDENtify - Names the waveform and the source/destination project.

ENCode - Lists the data encoding format and the maximum and minimum waveform amplitude value in Volts. The waveform data for the WaveStation is encoded as IEEE, 32 bit single precision floating point numbers.

DIMension - Specifies the structure and format of the data in the data block. The "=Volts" statement identifies the first dimension block as defining the waveform amplitude. Waveform data consists of explicit amplitude values, i.e. each amplitude value is listed individually. The size field lists the number of data values included in the data block, 64 in this example. The UNIT's field lists the amplitude units, V stands for Volts. The ANALog field indicates the type of waveform 0 for analog, 1 for digital. If this field does not exist it is assumed to be an analog wave.

The second dimension block, with the "=Time" statement, defines the waveform horizontal scale as an implicit function of time. The time information is determined implicitly by knowing the amplitude sample number and the spacing between samples. The SCAle field supplies the horizontal or sampling interval and the OFFSet lists the horizontal offset displacement. The UNIT's field lists the horizontal units, s stands for seconds.

ANALog - This field indicates the waveform is "analog" (0) or "digital" (1). If the field is not present the waveform is "analog".

TRACe - The trace block is used to report the time cursor positions as indicated by "= cursor_include". The ABel field defines the time interval between the time left cursor (STARt) and time right cursor (Stop).

DATA - The data block contains the actual values of the waveform amplitude data. This is a fixed length block of 256 bytes defined by the block length field, in this example the #3 indicates that the byte count contains 3 digits which are 256. The data, which is not printable follows.

A second DIF expression, which contains information on the waveform marker is appended to the file describing the waveform. This is done because the marker data is described differently from the waveform data. Two marker types are available, edge or clock markers. If the edge marker type is selected then the marker is described as a series of paired data values or "tuples". The first value in the pair is the marker time position. The second is its binary state, i.e. 1 or 0. The following blocks are specific to the waveform edge marker description.

DIMension - The "= time" statement defines the first value in the marker data pair. In this example the marker consists of two edges at 0 and 80ns. Up to 125 marker edges can be defined. The second dimension block describes the marker amplitudes, at each time value, in terms of the logical value. The UNIT's field defines the selected marker logic level which can be TTL or ECL.

ORDer - This block specifies that the data will be paired into tuples consisting of a time value and a binary state (1 or 0).

DATA - The marker data block, identified by the "=markers" statement, contains ordered pairs of data values representing the edge marker time position and logical state. All values in the data field will be separated by commas.

If the clock marker has been selected then the data block will be different. A typical data block for the clock marker follows:

DATA = markers (WAVeform (PERiod 8.000000e-008 TMAX 5.000000e-8))

The marker is described as waveform type data which summarizes the key clock marker parameters, the clock period and time to the first rising or positive going edge, (TMAX).

Viewing Waveform Data In The DIF file

The waveform data, within the DIF file, is encoded as IEEE 32 bit, single precision, floating point numbers. Viewing this data requires a program which converts binary data into printable hexadecimal (hex) values. Programs such as DOS's debug provide this capability. A DIF file for the waveform, NEW_WAVE, is shown below in an HEX/ASCII format. The waveform data is indicated by bold text.

```
000000 28 44 49 46 20 28 56 45 52 53 69 6f  6e 20 31 39   (DIF (VERSion 19
000010 39 33 2e 30 20 53 43 4f  50 65 20 46 55 4c 4c 29   93.0 SCOPe FULL)
000020 20 49 44 45 4e 74 69 66 79 28 20 4e 41 4d 45 20    IDENtify( NAME
000030 22 4e 45 57 5f  57 41 56 45 22 20 50 52 4f  4a 20   "NEW_WAVE" PROJe
000040 63 74 20 22 44 45 4d 4f  2e 50 52 4a 22 20 29 20   ct "DEMO.PRJ" )
000050 45 4e 43 6f  64 65 28 20 46 4f  52 4d 61 74 20 49   ENCode( FORMat I
000060 46 50 33 32 20 48 52 41 4e 47 45 20 30 2e 35 30    FP32 HRANGE 0.50
000070 30 30 30 30 20 4c 52 41 4e 47 45 20 2d 30 2e 35    0000 LRANGE -0.5
000080 30 30 30 30 30 20 29 20 44 49 4d 65 6e 73 69 6f    00000 ) DIMensio
000090 6e 20 3d 20 56 6f  6c 74 73 20 28 20 54 59 50 45   n = Volts ( TYPE
0000a0 20 45 58 50 4c 69 63 69 74 53 49 5a 45 20 36 34    EXPLicitSIZE 64
0000b0 20 55 4e 49 54 73 20 22 56 22 20 29 20 44 49 4d    UNITs "V" ) DIM
0000c0 65 6e 73 69 6f  6e 20 3d 20 54 69 6d 65 20 28 20   ension = Time (
0000d0 54 59 50 45 20 49 4d 50 4c 69 63 69 74 20 53 43    TYPE IMPLicit SC
0000e0 41 4c 65 20 32 2e 35 65 2d 30 30 39 20 4f  46 46   ALe 2.5e-009 OFF
0000f0  53 65 74 20 30 20 55 4e 49 54 73 20 22 73 22 20   Set 0 UNITs "s"
000100 29 20 54 52 41 43 65 20 3d 20 43 75 72 73 6f  72   ) TRACe = Cursor
000110 73 5f  69 6e 63 6c 75 64 65 20 28 4c 41 42 45 4c   s_include (LABEL
000120 20 54 69 6d 65 20 53 54 41 52 74 20 30 20 53 54    Time STARt 0 ST
000130 4f  50 20 31 2e 35 39 39 65 2d 30 30 37 29 20 44   OP 1.599e-007) D
000140 41 54 41 20 3d 20 64 61 74 61 5f  61 72 72 61 79   ATA = data_array
000150 20 28 20 43 55 52 56 65 20 28 20 56 41 4c 75 65    ( CURVe ( VALue
000160 73 20 23 33 32 35 36 00 00 00 00 00 00 f0 be 00    s #3256.......?.
000170 30 8d 24 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    0.$.....0.....?.
000180 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
000190 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001a0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001b0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001c0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001d0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001e0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
0001f0 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
000200 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
000210 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
000220 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
000230 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00    .S%.....0.....?.
```

```
000240 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00   .S%.....0.....?.
000250 c8 53 25 00 00 00 bf 00 30 0d a5 00 00 00 3f 00   .S%.....0.....?.
000260 30 8d 24 00 00 00 bf 20 29 20 29 20 29 28 44 49   0.$.... ) ) )(DI
000270 46 20 28 56 45 52 53 69 6f 6e 20 31 39 39 33 2e   F (VERSion 1993.
000280 30 29 20 44 49 4d 65 6e 73 69 6f 6e 20 3d 20 54   0) DIMension = T
000290 69 6d 65 20 28 20 54 59 50 45 20 45 58 50 4c 69   ime ( TYPE EXPLi
0002a0 63 69 74 20 53 49 5a 45 20 32 29 20 44 49 4d 65   cit SIZE 2) DIMe
0002b0 6e 73 69 6f 6e 20 3d 20 50 6f 6c 61 72 69 74 79   nsion = Polarity
0002c0 20 28 20 54 59 50 45 20 45 58 50 4c 69 63 69 74    ( TYPE EXPLicit
0002d0 20 55 4e 49 54 73 20 22 54 54 4c 22 29 20 4f 52    UNITs "TTL") OR
0002e0 44 65 72 28 42 59 20 54 55 50 4c 65 29 20 44 41   Der(BY TUPLe) DA
0002f0 54 41 20 3d 20 6d 61 72 6b 65 72 73 20 28 20 43   TA = markers ( C
000300 55 52 56 65 20 28 20 56 41 4c 75 65 73 20 32 2e   URVe ( VALues 2.
000310 35 30 30 30 30 30 65 2d 30 30 39 2c 31 2c 20 38   500000e-009,1, 8
000320 2e 30 30 30 30 30 30 65 2d 30 30 38 2c 30 29 20   .000000e-008,0)
000330 29 20 29 0a  ) ).
```

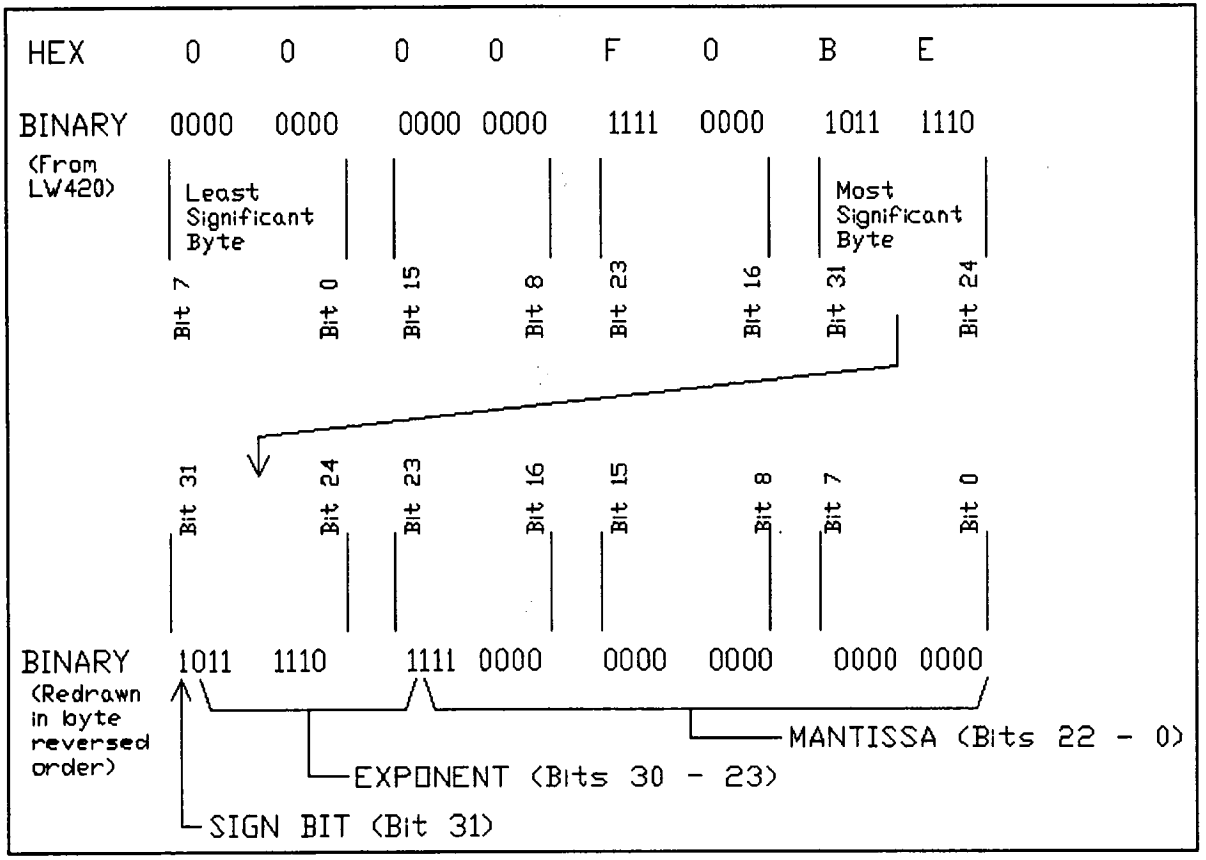
Interpreting Waveform Data Values

32 Floating point data can be converted back to fixed point decimal data using the following equation:

$$\text{DATA Value (Volts)} = (-1)^{S} \cdot 2^{E-127} \cdot (1.F)$$

where:
- S - sign of the number (1 bit)
- E - exponent (8 bits)
- F - mantissa or fractional part (23 bits)

The sign, exponent, and mantissa elements must be extracted from the 32 bit binary value output from the WaveStation. The following example, which uses the second 32 bit data value in the file above (0000F0BE), shows how this is accomplished:



Note that interpretation of the floating point values is simplified by reversing the byte order of the data as shown. The sign bit, bit 31, is now the most significant bit. The exponent is represented by bits 30 through 23. The mantissa, or fractional part of the floating point number, is contained in bits 22 through 0.

For the hex value 0000F0BE, the components of the floating point encoded amplitude value are:

S = 1

E = 125 (011 1110 1 in binary)

F = 0.875

Note that the fraction, F, is calculated as: 700000_h / 800000_h (7340032/ 8388608). This is the binary value of bits 22 - 0 divided by 2^{23}

Using the values obtained above in the equation for the data value:

$$\text{DATA Value (Volts)} = (-1)^1 \cdot 2^{125-127} \cdot (1.875) = -0.46875$$

Other Data Formats

The WaveStation can export and import files in multiple data formats including, spreadsheet, Mathcad, Matlab, Pspice, Easywave, and compressed DIF. Import and export file transfers are made directly to and from the internal floppy disk drive only.

*CAL?

Purpose: Performs a system calibration and returns a numeric response indicating if the calibration was successful.

Command: None

Query: *CAL?

Response: 0 = Calibration successful,
1 = Calibration failed

Arguments: None

*CLS

Purpose: Clears all event status registers. This includes the main Status Byte Register, Event Status Register, Operation Status Event Register, and Questionable Status Event Register. *CLS does not clear the Operation Status Condition Register or the Questionable Status Condition Register.

Command: *CLS

Query: None.

Response: None

Arguments: None

*ESE

Purpose: Sets the bits of the standard Event Status Enable register (ESE). Each bit in the Event Status Register must be enabled to be summarized in the main status byte. Any reported ESR bit, for which the matching ESE bit is set, sets the ESB summary message bit (bit #5) of the main status byte(STB). The bits in the ESE register have been defined by IEEE-488.2.

The Event Status Enable Register bit assignments are as follows:

Bit 7: Power On	(Decimal 128)
Bit 6: User Request	(Decimal 64)
Bit 5: Command Error	(Decimal 32)
Bit 4: Execution Error	(Decimal 16)
Bit 3: Calibration Error	(Decimal 8)
Bit 2: Query Error	(Decimal 4)
Bit 1: Request Control	(Decimal 2)
Bit 0: Operation Complete	(Decimal 1)

Command: *ESE <numeric_value>

Query: *ESE?

Response: <numeric_value>

Arguments: <numeric_value>

*ESR?

Purpose: Reads and clears the contents of the Standard Event Status Register (ESR).

IEEE-488.2 defines the ESR to report error conditions common to most automatic test equipment. These bits are used in synchronization and error reporting.

If the bits in the ESR have been enabled by the Standard Event Enable Register they will be summarized in bit 5 of the main Status Byte Register.

The bit assignments for the Standard Event Status Register are as follows:

Bit 7: Power On	(Decimal 128)
Bit 6: User Request	(Decimal 64)
Bit 5: Command Error	(Decimal 32)
Bit 4: Execution Error	(Decimal 16)
Bit 3: Device Dependent Error	(Decimal 8)
Bit 2: Query Error	(Decimal 4)
Bit 1: Request Control	(Decimal 2)
Bit 0: Operation Complete	(Decimal 1)

Command: None.

Query: *ESR?

Response: <value>

Arguments: None

*IDN?

Purpose: Identifies the instrument. The response indicates the manufacturer, the model, the serial number and the software revision level.

Command: None.

Query: *IDN?

Response: <manufacturer>, <model number>, <serial number>, <software revision>

Arguments: None

*LRN?

Purpose: Learn device setup

Command: *LRN <RESPONSE MESSAGE UNIT>

Query: *LRN?

Response: Sequence of <RESPONSE MESSAGE UNIT>

Arguments: None

Notes: *A sequence of <RESPONSE MESSAGE UNIT> elements may later be used as <PROGRAM MESSAGE UNIT> elements to return the device to this state.*

*OPC

Purpose: When pending operation complete, notify the controller

Command: *OPC - turns on the OPC bit in the ESR to notify the controller

Query: *OPC? - places a '1' into the output queue to notify the controller.

Response: 1

Arguments: None

Notes: *The operations under :WAVE:TIME, and SEQ:COMPile, are overlapped commands. Unlike WAI, *OPC does not wait - commands after *OPC continue to execute without delay.*

*PCB

Purpose: Identifies the address to Pass Control Back to when the LW400 is about to be given control of the GPIB bus.

Command: *PCB <numeric_value>

Query: None

Response: None

Arguments: <numeric_value> 0 to 30

Notes: *Secondary addresses are not supported by the LW400.*
This command is expected to be used when another controller is active, and the LW400 must get data from a DSO. See ":WAVE:INSert:SCOPe:CONTrol".

*RST

Purpose: Force service specific functions to a known state.

Command: *RST

Query: None

Response: None

Arguments: None

Notes: *The scope of *RST is the same as the scope of *LRN?*
**RST also cancels pending *OPC or *OPC? commands.*

*SRE

Purpose: Sets the 8-bit Status Byte Enable Register (SRE). The SRE mask determines which events in the main Status Byte (STB) register are able to generate a GPIB Service Request (SRQ). If an event is enabled and transitions from false (0) to true (1), an interrupt (SRQ) is sent to the GPIB controller. Clearing the SRE mask disables SRQ interrupts. The RQS (bit 6) is ignored in the SRE.

The bit assignments for the Main Status Byte Register are as follows:

Bit 7: Operation Status Summary	(Decimal 128)
Bit 6: RQS	(Decimal 64)
Bit 5: Standard Event Status Summary	(Decimal 32)
Bit 4: Message Available	(Decimal 16)
Bit 3: Questionable Status Summary	(Decimal 8)
Bit 2: Error/Event Queue	(Decimal 4)
Bit 1: Pass/Fail Status	(Decimal 2)
Bit 0: Not Used	(Decimal 1)

Command: *SRE <numeric_value>

Query: *SRE?

Response: <numeric_value>

Arguments: <numeric_value>

Notes: *A GPIB Service Request (SRQ) MUST be serviced by a serial poll and the registers must be cleared using the *CLS Command before another SRQ may be generated.*

*STB

Purpose: Reads and clears the contents of the Main Status Byte (STB). The main status byte summarizes the status for the entire system. If the status byte Enable register has enabled a cause of SRQ, a GPIB Service Request (SRQ) will be generated when an enabled bit changes from false (0) to true (1). Query of the Status Byte Register with *STB? (or *STB) will return a decimal number representing the bits that are set (true) in the status register. Reading the register will clear it.

The main Status Register may also be read by a GPIB serial poll. The bit assignments for the Main Status Byte Register are as follows:

Bit 7: Operation Status Summary	(Decimal 128)
Bit 6: RQS/MSS	(Decimal 64)
Bit 5: Standard Event Status Summary	(Decimal 32)
Bit 4: Message Available	(Decimal 16)
Bit 3: Questionable Status Summary	(Decimal 8)
Bit 2: Error/Event Queue	(Decimal 4)
Bit 1: Pass/Fail Status	(Decimal 2)
Bit 0: Not Used	(Decimal 1)

Command: None.

Query: *STB?

Response: <numeric_value>

Arguments: None

Notes: *When the status byte is read with *STB, the Master Summary Status appears in bit 6. Unlike RQS, which appears in bit 6 in response to serial poll, MSS does not go to 0 when the device is polled.*

*A GPIB Service Request (SRQ) MUST be serviced by a serial poll and the registers must be cleared using the *CLS Command before another SRQ may be generated*

*TRG

Purpose: Same as the MANUAL button on the TRIGGER menu, or GET (IEEE 488 Group Execute Trigger addressed command), or "INITIATE". Triggers the LW400.

Command: *TRG

Query: None

Response: None

Arguments: None

*TST?

Purpose: Perform an internal self-test, and return a numeric response indicating if self test was successful.

Command: *TST

Query: *TST?

Response: 0 = selftest successful
1 = selftest failed

Arguments: None

*WAI

Purpose: Wait until all overlapped (pending) operations have completed before executing any further commands or queries.

Command: *WAI

Query: None

Response: None

Arguments: None

CALibration[:ALL]?

Purpose: Performs a system calibration and returns a status code indicating if the calibration was successful.

0 = Calibration successful
1 = Calibration failed

Command: None.

Query: CALibration?

Response: <numeric_value>

Arguments: None

Notes: *This command is identical to *CAL?*

DISPlay:ANNotation:DATE[:STATe]

Purpose: Allows the date (top left-hand corner of screen) to be switched on or off.

Command: DISPlay:ANNotation:DATE <Boolean>

Query: DISPlay:ANNotation:DATE?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables the real time clock display
1 Enables the real time clock display
OFF Disables the real time clock display
ON Enables the real time clock display

DISPlay:ANNotation:LOGO[:STATe]

Purpose: Allows the Company Logo (top right-hand corner of screen) to be switched on or off.

Command: DISPlay:ANNotation:LOGO <Boolean>

Query: DISPlay:ANNotation:LOGO?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables the logo.
1 Enables the logo.
OFF Disables the logo.
ON Enables the logo.

DISPlay:ANNotation:PARameter[:STATe]

Purpose: Turns the parameters (bottom of the screen) on or off.

Command: DISPlay:ANNotation:PARameter <Boolean>

Query: DISPlay:ANNotation:PARameter?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables parameter display
1 Enables parameter display
OFF Disables parameter display
ON Enables parameter display

DISPlay:ANNotation[:ALL]

Purpose: Performs same function as DISP:ANN:LOGO. Present because this is a SCPI default node.

Command: DISPlay:ANNotation <Boolean>

Query: DISPlay:ANNotation?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

DISPlay:SSAVe

Purpose: Allows the automatic screen saver to be enabled or disabled.

Command: DISPlay:SSAVe <Boolean>

Query: DISPlay:SSAVe?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables screen saver
1 Enables screen saver
OFF Disables screen saver
ON Enables screen save

DISPlay[:WINDow]:TRACe:ALL

Purpose: Displays the whole waveform on the screen.

Command: DISPlay:TRACe:ALL

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:COLor

Purpose: Set the trace intensity. Although trace intensity may be set for each trace, these commands are coupled. Setting the intensity for one trace will set the same intensity for all traces.

Command: DISPlay:TRACe:COLor <numeric_value>

Query: DISPlay:TRACe:COLor?

Response: <numeric_value>

Arguments: <numeric_value> Intensity expressed as a percentage (0-100) Default is 75.

DISPlay[:WINDow]:TRACe:CURSors:TIME:DELTa

Purpose: Change the delta time between the time cursors. This command only has effect if DISPlay[:WINDow]:TRACe:CURSors:TIME:TRACk is on.

Command: DISPlay:TRACe:CURSors:TIME:DELTa <numeric_value>

Query: DISPlay:TRACe:CURSors:TIME:DELTa?

Response: <numeric_value>

Arguments: <numeric_value> Delta between the time cursors (0s - waveform length).

Notes: *If DISP:TRACE:CURSORS:TIME:TRACK is off, the value of Delta is not coupled to the cursors and the query does not necessarily indicate the separation of the cursors. See ...TIME:TRACK.*

DISPlay[:WINDow]:TRACe:CURSors:TIME:LEFT

Purpose: Set the position of the left time cursor.

Command: DISPlay:TRACe:CURSors:TIME:LEFT <numeric_value>

Query: DISPlay:TRACe:CURSors:TIME:LEFT?

Response: <numeric_value>

Arguments: <numeric_value> Left time cursor position (0s - end of waveform)

DISPlay[:WINDow]:TRACe:CURSors:TIME:RIGHt

Purpose: Set the position of the right time cursor. This command only has effect if DISPlay[:WINDow]:TRACe:CURSors:TIME:TRACk is off.

Command: DISPlay:TRACe:CURSors:TIME:RIGHt <numeric_value>

Query: DISPlay:TRACe:CURSors:TIME:RIGHt?

Response: <numeric_value>

Arguments: <numeric_value> Right cursor position (0s - end of waveform).

Notes: *The query response is always correct, even if TRACK is on.*

DISPlay[:WINDow]:TRACe:CURSors:TIME:SALL

Purpose: Select All selects the entire waveform by placing the left cursor at time zero and the right cursor at the end of the waveform.

Command: DISPlay:TRACe:CURSors:TIME:SALL

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:CURSors:TIME:TEND

Purpose: To End places both cursors at the end of the waveform.

Command: DISPlay:TRACe:CURSors:TIME:TEND

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:CURSors:TIME:TGRid

Purpose: To Grid moves both time cursors so they are on the display. The left time cursor gets placed one division in from the left edge of the grid or at the beginning of the waveform if it is to the right of the first division. The right time cursor gets placed one division in from the right edge of the grid or at the end of the waveform if the end is to the left of that division.

Command: DISPlay:TRACe:CURSors:TIME:TGRid

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:CURSors:TIME:TRACk

Purpose: Enables or disables time cursor tracking. When enabled, the position of the right time cursor is LEFT plus DELTa. The TIME:RIGHt Command has no effect.

Command: DISPlay:TRACe:CURSors:TIME:TRACk <Boolean>

Query: DISPlay:TRACe:CURSors:TIME:TRACk?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables cursor tracking.
1 Enables cursor tracking.
OFF Disables cursor tracking.
ON Enables cursor tracking.

Notes: *Changing the state of TRACK does not move the cursors. The value of ...TIME:DELTA is set to reflect the current position of the cursors when TRACK transitions from off to on. ...TIME:RIGHT is always maintained, so changing track from ON to OFF does not move the cursors, either.*

DISPlay[:WINDow]:TRACe:CURSors:TIME[:STATe]

Purpose: Turns the time cursors on or off.

Command: DISPlay:TRACe:CURSors:TIME <Boolean>

Query: DISPlay:TRACe:CURSors:TIME?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turns the time cursors off.
1 Turns the time cursors on.
OFF Turns the time cursors off.
ON Turns the time cursors on.

DISPlay[:WINDow]:TRACe:CURSors:VOLTage:BOTTom

Purpose: Set the position of the bottom voltage cursor.

Command: DISPlay:TRACe:CURSors:VOLTage:BOTTom <numeric_value>

Query: DISPlay:TRACe:CURSors:VOLTage:BOTTom?

Response: <numeric_value>

Arguments: <numeric_value> Bottom voltage cursor position (± 5 volts).

DISPlay[:WINDow]:TRACe:CURSors:VOLTage:DELTa

Purpose: Change the delta voltage between the voltage cursors. This command only has effect if DISPlay[:WINDow]:TRACe:CURSors:VOLTage:TRACk is on.

Command: DISPlay:TRACe:CURSors:VOLTage:DELTa <numeric_value>

Query: DISPlay:TRACe:CURSors:VOLTage:DELTa?

Response: <numeric_value>

Arguments: <numeric_value> Delta between the voltage cursors in volts (+ 5 volts).

Notes: *The state of DELTa is not coupled to the cursors if ...VOLTAGE:TRACK is off. The command has no affect and the query response does not necessarily reflect the separation of the cursors.*

DISPlay[:WINDow]:TRACe:CURSors:VOLTage:TGRid

Purpose: To Grid moves both voltage cursors so they are on the display. The top voltage cursor gets placed one division below the top edge of the grid. The bottom voltage cursor gets placed one division above the bottom edge of the grid.

Command: DISPlay:TRACe:CURSors:VOLTage:TGRid

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:CURSors:VOLTage:TOP

Purpose: Set the position of the top voltage cursor. This command only has effect if DISPlay[:WINDow]:TRACe:CURSors:VOLTage:TRACk is off.

Command: DISPlay:TRACe:CURSors:VOLTage:TOP <numeric_value>

Query: DISPlay:TRACe:CURSors:VOLTage:TOP?

Response: <numeric_value>

Arguments: <numeric_value> Top voltage cursor position (±5 volts).

DISPlay[:WINDow]:TRACe:CURSors:VOLTage:TRACk

Purpose: Enables or disables time cursor tracking.

Command: DISPlay:TRACe:CURSors:VOLTage:TRACk <Boolean>

Query: DISPlay:TRACe:CURSors:VOLTage:TRACk?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables cursor tracking.
1 Enables cursor tracking.
OFF Disables cursor tracking.
ON Enables cursor tracking.

Notes: *Changing the state of TRACK does not move the cursors. The value of ...VOLTAGE:DELTA is set to reflect the current position of the cursors when TRACK transitions from OFF to ON.*

DISPlay[:WINDow]:TRACe:CURSors:VOLTage[:STATe]

Purpose: Turns the voltage cursors on or off.

Command: DISPlay:TRACe:CURSors:VOLTage <Boolean>

Query: DISPlay:TRACe:CURSors:VOLTage?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turns the voltage cursors off.
1 Turns the voltage cursors on.
OFF Turns the voltage cursors off.
ON Turns the voltage cursors on.

DISPlay[:WINDow]:TRACe:GRATicule:COLor

Purpose: Set the display intensity for the grid.

Command: DISPlay:TRACe:GRATicule:COLor <numeric_value>

Query: DISPlay:TRACe:GRATicule:COLor?

Response: <numeric_value>

Arguments: <numeric_value> Grid intensity in percentage (0 - 100)
Default is 40

DISPlay[:WINDow]:TRACe:GRATicule:GRID[:STATe]

Purpose: Select or query the grid style. The grid may be a full grid (enabled), no grid (disable), or set to a cross hair (CHAir).

Command: DISPlay:TRACe:GRATicule:GRID <character_data>

Query: DISPlay:TRACe:GRATicule:GRID?

Response: <character_data>

Arguments: one of: ON, OFF, CHAir

CHAir Select a Cross-Hair grid.
OFF Disable Grid.
ON Enable Grid.

Notes: *SCPI defines this command as taking a Boolean argument. Our implementation matches our menu controls but conflicts with SCPI in that 0 and 1 are not useable as arguments.*

DISPlay[:WINDow]:TRACe:GRATicule:TYPE

Purpose: Selects the type of grid to display. The query form returns the currently selected grid type.

Command: DISPlay:TRACe:GRATicule:TYPE <character_data>

Query: DISPlay:TRACe:GRATicule:TYPE?

Response: <character_data>

Arguments: one of: SINGle, DUAL, SXY, XY

DUAL	Select a dual grid display.
SINGle	Select a single grid display.
SXY	Select a single + XY display.
XY	Select a XY display.

DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer

Purpose: Sets the time at the horizontal center of the grid. Zoom functions zoom around the center of the grid.

Command: DISPlay:TRACe:X:CENTer <numeric_value>

Query: DISPlay:TRACe:X:CENTer?

Response: <numeric_value>

Arguments: <numeric_value> Sets the time at the center of the grid (0s - maximum waveform duration).

Notes: *Maximum waveform duration depends on the clock decade and the amount of installed high speed memory.*

DISPlay[:WINDow]:TRACe:X[:SCALe]:PDIVision

Purpose: Sets the horizontal time per division of the grid.

Command: DISPlay:TRACe:X:PDIVision <numeric_value>

Query: DISPlay:TRACe:X:PDIVision?

Response: <numeric_value>

Arguments: <numeric_value> Horizontal time per divison (3ns - maximum waveform duration /8).

DISPlay[:WINDow]:TRACe:X[:SCALe]:TCURsors

Purpose: To Cursors displays the portion of the waveform between the time cursors with the left cursor one division from the left edge of the grid and the right cursor one division from the right edge of the grid.

Command: DISPlay:TRACe:X:TCURsors

Query: None

Response: None

Arguments: None

DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision

Purpose: Sets the vertical volts per division of the grid.

Command: DISPlay:TRACe:Y:PDIVision <numeric_value>

Query: DISPlay:TRACe:Y:PDIVision?

Response: <numeric_value>

Arguments: <numeric_value> Vertical volts per divison (10 mV - 5 V).

DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel

Purpose: Sets the voltage at the vertical center of the grid. Zoom functions zoom around the center of the grid.

Command: DISPlay:TRACe:Y:RLEVel <numeric_value>

Query: DISPlay:TRACe:Y:RLEVel?

Response: <numeric_value>

Arguments: <numeric_value> The voltage at the vertical center of the grid (±5 volts).

DISPlay[:WINDow]:TRACe:ZPRevious

Purpose: Zoom Previous sets the zoom settings back to the previous time and voltage zoom settings.

Command: DISPlay:TRACe:ZPREvious

Query: None

Response: None

Arguments: None

EQUation:CALCulate

Purpose: Calculates the currently selected equation line (EQUation:LINE) for a duration of EQUation:DURation and inserts it into the current waveform at the left cursor position in the insert mode defined by WAVE:INSert:MODE.

Command: EQUation:CALCulate

Query: None

Response: None

Arguments: None

EQUation:DATA

Purpose: Transfers all the lines of the equation sheet as a "#0" block.

Command: EQUation:DATA <block>

Query: EQUation:DATA?

Response: <indefinite length block>

Arguments: <indefinite length block>

Notes: *An indefinite length block: "#0" followed by all 16 lines in the current equation sheet, each 50 characters followed by a "new line" character.*

EQUation:DEFine

Purpose: Defines an equation for the current equation line (EQUation:LINE). The equation line may be up to 50 characters in length and must be surrounded by quotes. Valid functions are: SIN, COS, SQRT,PULSE, STEP, LN, LOG, ABS, EXP and TAN. Valid operators are: +, -, *, /, (,), ",", = and ^. Valid variable names are X1 through X16. Valid arguments are T, PI, and NOISE.

Command: EQUation:DEFine <string>

Query: EQUation:DEFine?

Response: <string>

Arguments: <string>

EQUation:DURation

Purpose: Sets the time span over which the equation will be calculated. The equation will be calculated for DURation seconds with time zero starting at the left cursor.

Command: EQUation:DURation <numeric_value>

Query: EQUation:DURation?

Response: <numeric_value>

Arguments: <numeric_value>

Notes: *Limits of <numeric value> above depend on amount of installed memory and clock decade. With 1 M/channel:* *400 MHz: 2.62 ms, max*
40 kHz 26.2s, max

EQUation:LINE

Purpose: Selects an equation line from the current equation sheet. This is the line that other equation functions will operate on such as EQUation:DEFine, EQUation:DURation and EQUation:CALCulate.

Command: EQUation:LINE <numeric_value>

Query: EQUation:LINE?

Response: <numeric_value>

Arguments: <numeric_value> 1 to 16

EQUation:NEW

Purpose: Creates a new equation sheet in the equation editor.

Command: EQUation:NEW <string>

Query: EQUation:NEW?

Response: <string>

Arguments: <string>

EQUation:OPEN

Purpose: Opens an existing equation sheet.

Command: EQUation:OPEN <string>

Query: EQUation:OPEN?

Response: <string>

Arguments: <string>

Notes: *The <string> above is the name of an equation sheet which was previously SAVEd in this project. The equation sheet in memory is replaced.*

EQUation:SAVE

Purpose: Saves the current equation sheet. If a name other than the current name of the equation sheet is given then the current equation sheet is saved with the new name. The old equation sheet is left unchanged. If a name (other than the current equation sheet) is given that already exists, then an error status will be generated, an error code will be placed in the event queue and the equation sheet will not be saved.

Command: EQUation:SAVE <string>

Query: EQUation:SAVE?

Response: <string>

Arguments: <string>

FGENerator#:DC:LEVel

Purpose: Set the DC voltage level for the specified channel's function generator (either 1 or 2).

Command: FGENerator#:DC:LEVel <numeric_value>

Query: FGENerator#:DC:LEVel?

Response: <numeric_value>

Arguments: <numeric_value> (may be between + and - 5 V)

Notes: *See also FGEN#:STATe and FGEN#:SELect*

FGENerator#:MULTitone:AMPLitude

Purpose: Sets the peak to peak amplitude of the defined multitone function in the specified channel's function generator (either 1 or 2)..

Command: FGENerator#:MULTitone:AMPLitude <numeric_value>

Query: FGENerator#:MULTitone:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> may be from 0 to 10.0 V.

Notes: *See also FGEN#:STATE and FGEN#:SELect*
FGEN#:SINE, RAMP, Triangle, SQUare and MULTitone:AMPLitude are all value coupled.

FGENerator#:MULTitone:NTONes

Purpose: Sets the number of tones to be calculated for the multitone function in the specified channel's function generator).

Command: FGENerator#:MULTitone:NTONes <numeric_value>

Query: FGENerator#:MULTitone:NTONes?

Response: <numeric_value>

Arguments: <numeric_value> may be from 1 to 10.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*
When FGEN1:STATE is on, the multitone waveform is recalculated on the receipt of any FGEN1:MULT:... command.

FGENerator#:MULTitone:OFFSet

Purpose: Set the median voltage of the waveform in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:MULTitone:OFFSet <numeric_value>

Query: FGENerator#:MULTitone:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value> may be from -5 to +5

Notes: *See also FGEN#:STATe and FGEN#:SELect".*
When FGEN1:STATE is on, the multitone waveform is recalculated on the receipt of any FGEN1:MULT:... command.

FGENerator#:MULTitone:TONE#:RAMPlitude

Purpose: Set the relative amplitude of the current tone in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:MULTitone:TONE#:RAMPlitude <numeric_value>

Query: FGENerator#:MULTitone:TONE#:RAMPlitude?

Response: <numeric_value>

Arguments: <numeric_value> may be from -1.000 to +1.000

Notes: *See also FGEN#:STATe and FGEN#:SELect".*
When FGEN1:STATE is on, the multitone waveform is recalculated on the receipt of any FGEN1:MULT:... command.

FGENerator#:MULTitone:TONE#[:FREQuency]

Purpose: Set the frequency of the current tone in the specified channel's function generator (either 1 or 2). TONE# is TONE1 to TONE10.

Command: FGENerator#:MULTitone:TONE# <numeric_value>

Query: FGENerator#:MULTitone:TONE#?

Response: <numeric_value>

Arguments: <numeric_value> Frequency (1Hz - 100 MHz).

Notes: *See also FGEN#:STATe and FGEN#:SELect".*
When FGEN1:STATE is on, the multitone waveform is recalculated on the receipt of any FGEN1:MULT:... command.

FGENerator#:PULSe:AMPLitude

Purpose: Sets the base to top amplitude of the pulse in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:AMPLitude <numeric_value>

Query: FGENerator#:PULSe:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> -10 to +10 V

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:PULSe:BASE

Purpose: Sets the voltage of the non-triggered level of the pulse in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:BASE <numeric_value>

Query: FGENerator#:PULSe:BASE?

Response: <numeric_value>

Arguments: <numeric_value> -5 V to +5 V

Notes: *FGEN#:PULSE:BASE and FGEN#:SQUare:BASE are value coupled. See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:PULSe:ETIMe

Purpose: The 10%-90% edge time of both the rising and falling edges of the pulse in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:ETIMe <numeric_value>

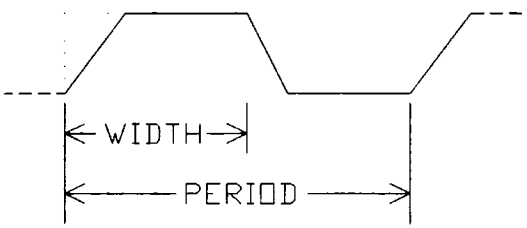
Query: FGENerator#:PULSe:ETIMe?

Response: <numeric_value>

Arguments: <numeric_value> 5 ns to 5 ms

Notes: *FGEN#:PULSe:ETIME and FGEN#:SQUare:ETIME are value coupled.*

*The time to transition from BASE to top (0 to 100%) will be approximately 100/80 *ETIME, or 1.25 *ETIME. As shown in the diagram below, PULSE:WIDTH +1.25 *ETIME must be <= FGEN#PULSE:PERiod, or the pulse cannot be produced.*



*If PULSE:SWEEP[:STATe] is ON, remember that ETIME does not change with frequency. At the STOP frequency width +1/25 *ETIME must fit in 1/STOP frequency.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:PERiod

Purpose: Sets the period (1/frequency) of the pulse in the specified channel's function generator (either 1 or 2), assuming FGEN#:PULSE:TDELay is 0.

Command: FGENerator#:PULSe:PERiod <numeric_value>

Query: FGENerator#:PULSe:PERiod?

Response: <numeric_value>

Arguments: <numeric_value> 20 ns to 1 s

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:PULSe:SWEep:SPACing

Purpose: Selects the type of sweep (either linear or log) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:SWEep:SPACing <character_data>

Query: FGENerator#:PULSe:SWEep:SPACing?

Response: <character_data>

Arguments: LINear or LOG

Notes: *LINear Sweep: the frequency is ...SWEEP:START at the beginning, ...SWEEP:STOP at ...SWEEP:TIME and increases at a constant rate in Hz/unit time in between.*

LOG Sweep: the frequency increases from ...SWEEP:START to ...SWEEP:STOP at a rate which is a constant percentage change in frequency per unit time. The time needed for the frequency to double, t_{x2}, for example, is

$$t_{x2} = \frac{(Sweeptime)}{Log\frac{(freq_{stop})}{(freq_{start})}} * \log(2)$$

This command is value coupled to all FGEN#:<any>:SWEEP:SPACing commands for the specified channel, that is, FGEN1 or FGEN2.

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:SWEep:STARt

Purpose: Sets the start frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:SWEep:STARt <numeric_value>

Query: FGENerator#:PULSe:SWEep:STARt?

Response: <numeric_value>

Arguments: <numeric_value> 1Hz to 50 MHz

Notes: *This command is value coupled to all FGEN#:<any>:SWEep:STARt commands for the specified channel. The upper limit is enforced when the waveform is built.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:SWEep:STOP

Purpose: Sets the stop frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:SWEep:STOP <numeric_value>

Query: FGENerator#:PULSe:SWEep:STOP?

Response: <numeric_value>

Arguments: <numeric_value> 1Hz to 50 MHz

Notes: *This command is value coupled to all FGEN#:<any>:SWEep:STOP commands for the specified channel. The upper limit is enforced when the waveform is built.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:SWEep:TIME

Purpose: Sets the amount of time that it will take to go from the SWEep:STARt to SWEep:STOP in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:SWEep:TIME <numeric_value>

Query: FGENerator#:PULSe:SWEep:TIME?

Response: <numeric_value>

Arguments: <numeric_value> 1 ns to 1 s.

Notes: *This command is value coupled to all FGEN#:<any>:SWEep:TIME commands for the specified channel.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:SWEep[:STATe]

Purpose: Turns the sweep on or off for the PULSE function in the specified channel's function generator (either 1 or 2). When sweep is off the parameters specified by FGENerator#:PULSe:PERiod and FGENerator#:PULSe: TDELay define the output pulse train.

Command: FGENerator#:PULSe:SWEep <Boolean>

Query: FGENerator#:PULSe:SWEep?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn sweep off.
1 Turn sweep on.
OFF Turn sweep off.
ON Turn sweep on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:PULSe:TDELay

Purpose: Sets the amount of time between the beginning of the waveform and the beginning of the first edge of the pulse in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:TDELay <numeric_value>

Query: FGENerator#:PULSe:TDELay?

Response: <numeric_value>

Arguments: <numeric_value>

Notes:

1. TDELay should be set to 0 when the waveform is playing continuously. TDELay adds time before the beginning the 1st pulse period. This is useful in single triggered mode where TRIGger:DELay affects both channels, but FGEN:PULSe:TDELay introduces a delay on only the selected channel.

2. FGEN#:PULSE:TDELAY and FGEN#:SQUare:TDELAY are value coupled.

3. See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:PULSe:WIDTh

Purpose: Sets the width of the pulse from 50% up the rising edge to 50% down the falling edge in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:PULSe:WIDTh <numeric_value>

Query: FGENerator#:PULSe:WIDTh?

Response: <numeric_value>

Arguments: <numeric_value> 5 ns to 1 s

Notes: *If FGEN#:PULSe:SWEep[:STATe] is ON, WIDTh specifies the width at the start frequency. Width decreases as frequency increases in the sweep, so that the duty cycle at the start frequency is maintained throughout the sweep.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:AMPLitude

Purpose: Sets the peak to peak amplitude of the ramp in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:AMPLitude <numeric_value>

Query: FGENerator#:RAMP:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value>, 0 to 10 V

NOTES: *FGEN#:RAMP:AMPLitude, FGEN#:SINE:AMPLitude, FGEN#:TRIangle:AMPLitude, FGEN#:SQUare: AMPLitude and FGEN#:MULTItone:AMPLitude are all value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:FREQuency

Purpose: Sets the frequency of the ramp in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:FREQuency <numeric_value>

Query: FGENerator#:RAMP:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 25 MHz

Notes: *FGEN#:RAMP:FREQuency and FGEN#:TRI:FREQuency are value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:INVert

Purpose: Controls whether the ramp is rising or falling in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:INVert <Boolean>

Query: FGENerator#:RAMP:INVert?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn invert off.
1 Turn invert on.
OFF Turn invert off.
ON Turn invert on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:RAMP:OFFSet

Purpose: Set the median voltage of the waveform in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:OFFSet <numeric_value>

Query: FGENerator#:RAMP:OFFSet?

Response: <numeric_value>

Arguments:: <numeric_value> -5 to +5 V

Notes: *FGEN#:SINE, TRIangle, RAMP and MULTitone:OFFSET are value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:SPOSition

Purpose: Sets the start position of the ramp in percentage of the ramp slope in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:SPOSition <numeric_value>

Query: FGENerator#:RAMP:SPOSition?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 100

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:RAMP:SWEep:SPACing

Purpose: Selects the type of sweep (either linear or log) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:SWEep:SPACing <character_data>

Query: FGENerator#:RAMP:SWEep:SPACing?

Response: <character_data>

Arguments: LINear or LOG

Notes: *See notes for FGEN#:PULSE:SWEep:SPACing*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:SWEep:STARt

Purpose: Sets the start frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:SWEep:STARt <numeric_value>

Query: FGENerator#:RAMP:SWEep:STARt?

Response: <numeric_value>

Arguments: <numeric_value> 1Hz to 25 MHz

Notes: *See notes for FGEN#:PULSE:SWEEP:STARt*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:SWEep:STOP

Purpose: Sets the stop frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:SWEep:STOP <numeric_value>

Query: FGENerator#:RAMP:SWEep:STOP?

Response: <numeric_value>

Arguments: <numeric_value> 1 Hz to 25 MHz

Notes: *See notes for FGEN#:PULSE:SWEEP:STOP*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:SWEep:TIME

Purpose: Sets the amount of time that it will take to go from SWEep:STARt to SWEep:STOP in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:RAMP:SWEep:TIME <numeric_value>

Query: FGENerator#:RAMP:SWEep:TIME?

Response: <numeric_value>

Arguments: <numeric_value>, 1 ns to 1 s

Notes: *See notes for FGEN#:PULSe:SWEep:TIME*
See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:RAMP:SWEep[:STATe]

Purpose: Turns the sweep on or off for the RAMP function in the specified channel's function generator (either 1 or 2). When sweep is off the parameters specified by FGENerator#:RAMP:FREQuency define the output ramp.

Command: FGENerator#:RAMP:SWEep <Boolean>

Query: FGENerator#:RAMP:SWEep?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn sweep off.
1 Turn sweep on.
OFF Turn sweep off.
ON Turn sweep on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:SELect

Purpose: Selects which function the specified channel's function generator outputs. The available functions are: SINE, TRIangle, SQUare, RAMP, PULSe, MULTitone, and DC.

Command: FGENerator#:SELect <character_data>

Query: FGENerator#:SELect?

Response: <character_data>

Arguments: SINE/TRIangle/SQUare/RAMP/PULSe/MULTItone/DC

FGENerator#:SINE:AMPLitude

Purpose: Sets the peak to peak amplitude of the sine wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:AMPLitude <numeric_value>

Query: FGENerator#:SINE:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 10 V

Notes: *FGEN:SINE, RAMP, TRIange, SQUare and MULTItone:AMPLitude are all valued coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:FREQuency

Purpose: Sets the frequency of the sine wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:FREQuency <numeric_value>

Query: FGENerator#:SINE:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>1 Hz to 100 MHz

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:SINE:OFFSet

Purpose: Set the voltage of the zero degree phase of the sinewave (the median voltage) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:OFFSet <numeric_value>

Query: FGENerator#:SINE:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V

Notes: *FGEN#:SINE, TRIangle, RAMP, and MULTitone:OFFset are valued coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:PHASe

Purpose: Sets the start phase of the sine wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:PHASe <numeric_value>

Query: FGENerator#:SINE:PHASe?

Response: <numeric_value>

Arguments: <numeric_value>, 0 to 360 (degrees).

Notes: *FGEN#:SINE:PHASe and FGEN#:TRIangle:PHASe are value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:SWEep:SPACing

Purpose: Selects the sweep type (either linear or log) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:SWEep:SPACing <character_data>

Query: FGENerator#:SINE:SWEep:SPACing?

Response: <character_data>

Arguments: LINear or LOG

Notes: *See notes for FGEN#:PULSe:SWEep:SPACing*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:SWEep:STARt

Purpose: Sets the start frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:SWEep:STARt <numeric_value>

Query: FGENerator#:SINE:SWEep:STARt?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 100 MHz

Notes: *See notes for FGEN#:PULSe:SWEep:STARt.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:SWEep:STOP

Purpose: Sets the stop frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:SWEep:STOP <numeric_value>

Query: FGENerator#:SINE:SWEep:STOP?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 100 MHz

Notes: *See notes for FGEN#:PULSe:SWEep:STOP*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:SWEep:TIME

Purpose: Sets the amount of time that it will take to go from SWEep:STARt to SWEep:STOP in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SINE:SWEep:TIME <numeric_value>

Query: FGENerator#:SINE:SWEep:TIME?

Response: <numeric_value>

Arguments: <numeric_value>, 1 ns to 1 S

Notes: *This command is value coupled to all FGEN#:,any>:SWEep:TIME commands for the specified channel.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SINE:SWEep[:STATe]

Purpose: Turns the sweep on or off for the SINE function in the specified channel's function generator (either 1 or 2). When sweep is off the parameters specified by FGENerator#:SINE:FREQuency defines the output sine wave.

Command: FGENerator#:SINE:SWEep <Boolean>

Query: FGENerator#:SINE:SWEep?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn sweep off.
1 Turn sweep on.
OFF Turn sweep off.
ON Turn sweep on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:SQUare:AMPLitude

Purpose: Sets the base to top amplitude of the square wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:AMPLitude <numeric_value>

Query: FGENerator#:SQUare:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value>, 0 to 10 V

Notes: *FGEN#:SINE, RAMP, TRIangle, SQUare and MULTItone:AMPLitude are value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:BASE

Purpose: Sets the voltage of the non-triggered level of the waveform in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:BASE <numeric_value>

Query: FGENerator#:SQUare:BASE?

Response: <numeric_value>

Arguments: <numeric_value>, -5 to +5 V

Notes: *FGEN#:SQUare:BASE and FGEN#:PULSe:BASE are valued coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:ETIMe

Purpose: The 10%-90% edge time of both the rising and falling edges of the square wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:ETIMe <numeric_value>

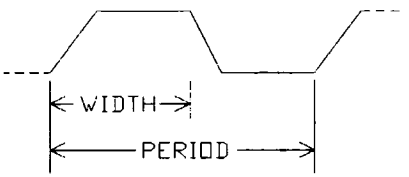
Query: FGENerator#:SQUare:ETIMe?

Response: <numeric_value>

Arguments: <numeric_value>, 5 ns to 1 s

Notes: *The time to transition from BASE to top (0 to 100%) will be approximately 100/80 X ETIME, or 1.25 x ETIME. 1.25 x ETIME must be less than 0.5/FGEN#:SQUare:FREQuency, or the square wave cannot be produced. If SWEEP is on, remember that ETIME does not change with frequency, so 1.25 x ETIME must be less than 0.5/STOP frequency.*

See also FGEN#:STATe and FGEN#:SELect".



FGENerator#:SQUare:FREQuency

Purpose: Sets the frequency of the square wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:FREQuency <numeric_value>

Query: FGENerator#:SQUare:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 50 MHz

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:SQUare:SWEep:SPACing

Purpose: Selects the sweep type (either linear or log) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:SWEep:SPACing <character_data>

Query: FGENerator#:SQUare:SWEep:SPACing?

Response: <character_data>

Arguments: LINear or LOG

Notes: *See notes for FGEN#:PULse:SWEep:SPACing*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:SWEep:STARt

Purpose: Sets the start frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:SWEep:STARt <numeric_value>

Query: FGENerator#:SQUare:SWEep:STARt?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 50 MHz

Notes: *See notes for FGEN#:PULSe:SWEep:STARt*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:SWEep:STOP

Purpose: Sets the stop frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:SWEep:STOP <numeric_value>

Query: FGENerator#:SQUare:SWEep:STOP?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 50 MHz

Notes: *See notes for FGEN#:PULse:SWEep:STOP*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:SWEep:TIME

Purpose: Sets the amount of time that it will take to go from SWEep:STARt to SWEep:STOP in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:SWEep:TIME <numeric_value>

Query: FGENerator#:SQUare:SWEep:TIME?

Response: <numeric_value>

Arguments: <numeric_value>, 1 ns to 1 s

Notes: *See notes for FGEN#:PULSe:SWEep:TIME*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:SQUare:SWEep[:STATe]

Purpose: Turns the sweep on or off for the square wave function in the specified channel's function generator (either 1 or 2). When sweep is off the parameter specified by FGENerator#:SQUare:FREQuency defines the output square wave.

Command: FGENerator#:SQUare:SWEep <Boolean>

Query: FGENerator#:SQUare:SWEep?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn sweep off.
1 Turn sweep on.
OFF Turn sweep off.
ON Turn sweep on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#:SQUare:TDELay

Purpose: Sets the amount of time before the first edge of the square wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:SQUare:TDELay <numeric_value>

Query: FGENerator#:SQUare:TDELay?

Response: <numeric_value>

Arguments: <numeric_value>

Notes: *See notes for FGEN#:PULSe:TDELay*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:AMPLitude

Purpose: Sets the peak to peak amplitude of the triangle wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:AMPLitude <numeric_value>

Query: FGENerator#:TRIangle:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value>, 0 to 10 V

Notes: *See note for FGEN#:RAMP:AMPLitude*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:FREQuency

Purpose: Sets the frequency of the triangle wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:FREQuency <numeric_value>

Query: FGENerator#:TRIangle:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 25 MHz

Notes: *FGEN#:TRIangle:FREQuency and FGEN#:RAMP:FREQuency are value coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:OFFSet

Purpose: Set the median voltage of the triangle waveform in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:OFFSet <numeric_value>

Query: FGENerator#:TRIangle:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value>, -5 to +5 V

Notes: *FGEN#:SINE, TRIangle, RAMP and MULTitone:OFFSet are valued coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:PHASe

Purpose: Sets the phase of the triangle wave in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:PHASe <numeric_value>

Query: FGENerator#:TRIangle:PHASe?

Response: <numeric_value>

Arguments: <numeric_value>, 0 to 360 (degrees)

Notes: *FGEN#:SINE:PHASE and FGEN#:TRIangle:PHASE are valued coupled.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:SWEep:SPACing

Purpose: Selects the sweep type (either linear or log) in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:SWEep:SPACing <character_data>

Query: FGENerator#:TRIangle:SWEep:SPACing?

Response: <character_data>

Arguments: LINear or LOG

Notes: *See notes for FGEN#:PULSE:SWEep:SPACing.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:SWEep:STARt

Purpose: Sets the start frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:SWEep:STARt <numeric_value>

Query: FGENerator#:TRIangle:SWEep:STARt?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 25 MHz

Notes: *See notes for FGEN#:PULSe:SWEep:STARt.*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:SWEep:STOP

Purpose: Sets the stop frequency of the sweep in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:SWEep:STOP <numeric_value>

Query: FGENerator#:TRIangle:SWEep:STOP?

Response: <numeric_value>

Arguments: <numeric_value>, 1 Hz to 25 MHz

Notes: *See notes for FGEN#:PULSe:SWEep:STOP*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:SWEep:TIME

Purpose: Sets the amount of time that it will take to go from SWEep:STARt to SWEep:STOP in the specified channel's function generator (either 1 or 2).

Command: FGENerator#:TRIangle:SWEep:TIME <numeric_value>

Query: FGENerator#:TRIangle:SWEep:TIME?

Response: <numeric_value>

Arguments: <numeric_value>, 1 ns to 1 s

Notes: *See notes for FGEN#:PULSe:SWEep:TIME*

See also FGEN#:STATe and FGEN#:SELect".

FGENerator#:TRIangle:SWEep[:STATe]

Purpose: Turns the sweep on or off for the TRIangle function in the specified channel's function generator (either 1 or 2). When sweep is off the parameter specified by FGENerator#:TRIangle:FREQuency defines the output triangle wave.

Command: FGENerator#:TRIangle:SWEep <Boolean>

Query: FGENerator#:TRIangle:SWEep?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn sweep off.
1 Turn sweep on.
OFF Turn sweep off.
ON Turn sweep on.

Notes: *See also FGEN#:STATe and FGEN#:SELect".*

FGENerator#[:STATe]

Purpose: Turns the function generator on or off in the specified channel (either 1 or 2).

Command: FGENerator# <Boolean>

Query: FGENerator#?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn off function generator.
1 Turn on function generator.
OFF Turn off function generator.
ON Turn on function generator.

Notes: *When FGEN#["STATE] is ON, FGEN#:SELECT and FGEN#:<selected function>: <any command> cause the wave to be immediately recalculated. When FGEN#[:STATE] is changed from on to off the channel continues to play the same waveform until either a different wave is opened or FGEN#[:STATE] is set to ON again. When FGEN#[:STATE] is changed from OFF to ON the function generator waveform is immediately recalculated. This behavior can be used to advantage. For example if FGEN1 Is playing a swept sine:*

FGEN1:STATE OFF
FGEN1:SINE:SWEEP:START 1 MHz; STOP 10 MHz; SPAC LOG
FGEN1:STATE ON
will only calculate one sweep instead of three.

HCOPy:AUToincr

Purpose: Enable / disable automatic increment of the filename when a hardcopy is stored to a file. With automatic increment enabled the hardcopy files will be stored in a sequence as follows: HCOPY001.PRN, HCOPY002.PRN, etc..

Command: HCOPy:AUToincr <Boolean>

Query: HCOPy:AUToincr?

Response: <Boolean>

Arguments: 1 of: 0, 1, OFF, ON

0 OFF
1 ON

HCOPy:FILename

Purpose: Set or query the current hardcopy file name.

Command: HCOPy:FILename <string>

Query: HCOPy:FILename?

Response: <string>

Arguments: A quoted string containing up to 5 alpha characters. A three digit HCOPy:INDex is appended to this to form the file name. HCOPy:FILename can be changed only by this command, not from the front panel. Default is "HCOPY".

HCOPy:INDex

Purpose: Set the index number used when the hardcopy filename is automatically incremented. For the file name HCOPY001.PRN the index is 1. The index may range from 0 to 999.

Command: HCOPy:INDex <numeric_value>

Query: HCOPy:INDex?

Response: <numeric_value>

Arguments: <numeric_value>, 0 TO 999

HCOPy:TARGet:GRAPhics:DESTination

Purpose: Set or query the destination for the hardcopy graphics file. This command is meant to be used with possible future options. At the moment the only destination is FLOPPY.

Command: HCOPy:TARGet:GRAPhics:DESTination <character_data>

Query: HCOPy:TARGet:GRAPhics:DESTination?

Response: <character_data>

Arguments: FLOPPY

HCOPy:TARGet:GRAPhics:FORMat

Purpose: Set or query the hardcopy graphics file format. Graphics files may be exported in formats that allow them to be read by common word processors, paint, and graphics packages. The arguments list all available formats.

Command: HCOPy:TARGet:GRAPhics:FORMat <character_data>

Query: HCOPy:TARGet:GRAPhics:FORMat?

Response: <character_data>

Arguments: PCX/TIF/BMP

HCOPy:TARGet:PRINter:DESTination

Purpose: Set or query the destination of the hardcopy printer data. The destination may be a port to which the printer is attached or it may be a disk drive where a file in printer format will be stored. The arguments list all possible destinations.

Command: HCOPy:TARGet:PRINter:DESTination <character_data>

Query: HCOPy:TARGet:PRINter:DESTination?

Response: <character_data>

Arguments: CENTronics/FLOPpy/GPIB

HCOPy:TARGet:PRINter:FFEed

Purpose: Set or query whether a form feed is automatically generated following a hardcopy. To place only one hardcopy on a page FORM FEED should be enabled.

Command: HCOPy:TARGet:PRINter:FFEed <Boolean>

Query: HCOPy:TARGet:PRINter:FFEed?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Turn form feed off.
1 Turn form feed on.
OFF Turn form feed off.
ON Turn form feed on.

HCOPy:TARGet:PRINter:MODel

Purpose: Set or query the selected printer model. The printer type set here should match the printer on which the hardcopy will be printed.

Command: HCOPy:TARGet:PRINter:MODel <character_data>

Query: HCOPy:TARGet:PRINter:MODel?

Response: <character_data>

Arguments: EMX - Epson MX/FX
ELQ - Epson LQ
HPLaserJet - HP Laserjet II
HPTHinkjet - HP Thinkjet

HCOPy:TARGet:PRINter:QUALity

Purpose: Set or query the print quality. Draft provides faster, but lower resolution printing. Proof provides higher resolution and higher quality printing. This setting is not available for all supported printers.

Command: HCOPy:TARGet:PRINter:QUALity <character_data>

Query: HCOPy:TARGet:PRINter:QUALity?

Response: <character_data>

Arguments: DRAFt/PROof

HCOPy:TARGet:PRINter:SIZE

Purpose: Set the size of the hardcopy. Notebook is a smaller size that is suitable for including into a lab notebook. Presentation provides a lager size print. The size is not setable for all printer types.

Command: HCOPy:TARGet:PRINter:SIZE <character_data>

Query: HCOPy:TARGet:PRINter:SIZE?

Response: <character_data>

Arguments: PRESentation
NOTebook

HCOPy:TARGet:TYPE

Purpose: Set or query the hardcopy format. Hardcopies may be formatted to provide data suitable for a printer, or data in a graphics file format.

Command: HCOPy:TARGet:TYPE <character_data>

Query: HCOPy:TARGet:TYPE?

Response: <character_data>

Arguments: PRINter/GRAPhics

HCOPy[:IMMediate]

Purpose: Begin a hardcopy to a printer or file.

Command: HCOPy

Query: None.

Response: None

Arguments: None

INITiate[:IMMediate]

Purpose: This command is used to trigger the system (INITiate the trigger system). The INITiate command is equivalent to the 488.2 command *TRG or the MANUAL button on the TRIGGER menu. If the system is not in a triggered mode or not waiting for a trigger this command has no effect.

Command: INITiate

Query: None.

Response: None

Arguments: None

MMEMory:CATalog:{:ALL}

Purpose: Read out information about waveform, sequence, and equation files in the current project.

Command: MMEMory:CATalog

Query: MMEMory:CATalog?

Response: Each file is listed in an entry formatted as follows:

DEFAULT , <15 char name blank filled to 15 chars.	EQUATION, type of file	272, size in bytes	1993/08/03, date	07:49 time

Arguments: None

MMEMory:CATalog:EQUation

Purpose: Read out directory information about equation files in the current project.

Command: None

Query: MMEMory:CATalog:EQUation?

Response: See MMEMory:CATalog:ALL

Arguments: None

Notes: *This is not the same as SCPI MMEM:CAT? query.*

MMEMory:CATalog:IMAGe

Purpose: Read out directory information about the image files in the current project.

Command: None

Query: MMEMory:CATalog:IMAGe?

Response: #0filename, type, size, date, time

Arguments: None

Example: MMEMory:CATalog:IMAGe? - gets a directory listing of image files stored in the current project

MMEMory:CATalog:SEQuence

Purpose: Read out information about sequence file in the current project.

Command: None

Query: MMEMory:CATalog:SEQuence?

Response: See MMEMory:CATalog:ALL

Arguments: None

MMEMory:CATalog:WAVeform

Purpose: Read out information about waveform files in the current project.

Command: None

Query: MMEMory:CATalog:WAVeform?

Response: See MMEMory:CATalog:ALL

Arguments: None

MMEMory:DATA

Purpose: Retrieve a waveform file from the project via GPIB.

Command: MMEMory:DATA filename,data

Query: MMEMory:DATA? filename

Response: A DIF expression

Arguments: Filename is a quoted string of up to 15 characters.
Data is DIF expression.

MMEMory:DATA:PREamble

Purpose: Retrieve the DIF header but not the VALUES of a waveform file in the current project.

Command: None

Query: MMEMory:DATA:PREamble? filename

Response: A DIF preamble see Section 5 for details.

Arguments: Filename is a quoted string of up to 15 characters.

MMEMory:DELete:EQUation

Purpose: Remove an equation file from the current project.

Command: MMEMory:DELete:EQUation filename

Query: None

Response: None

Arguments: Filename is a quoted string of up to 15 characters.

MMEMory:DELete:IMAGe

Purpose: Remove an image file from the current project.

Command: MMEMory:DELete:IMAGe filename

Query: None

Response: None

Arguments: filename - image file to be deleted, in quotes

Example: MMEMory:DELete:IMAGe "test.img" - deletes test.img from the current project

MMEMory:DELete:PROJect

Purpose: Discard an entire project (waves, equations sequences) from the LW400.

Command: MMEMory:DELete:PROJect filename

Query: None

Response: None

Arguments: Filename is a quoted string of up to 15 characters.

MMEMory:DELete:SEQuence

Purpose: Remove a sequence file from the current project.

Command: MMEMory:DELete:SEQuence filename

Query: None

Response: None

Arguments: Filename is a quoted string of up to 15 characters.

MMEMory:DELete [WAVeform]

Purpose: Discard a specific waveform from the LW400.

Command: MMEMory:DELete: WAVeform filename

Query: None

Response: None

Arguments: Filename is a quoted string of up to 15 characters.

OUTPut#:FILTer[:LPASs]:FREQuency

Purpose: Sets the bandwidth for the selected channel.

Command: OUTPut#:FILTer:FREQuency <numeric_value>

Query: OUTPut#:FILTer:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value> 100e6, 10e6, 1e6, 100e3, and 10e3.

Notes: *When a waveform file is opened or a function generator wave is created, the filters are automatically set to match the clock decades. They can subsequently be changed with this command. If values other than specified are sent the LW400 will round to the nearest available value.*

OUTPut#:NOISe:LEVel

Purpose: Sets the level of noise that is inserted into the waveform for the selected channel (1 or 2). Noise of this level will be produced if OUTPut#:NOISe:[STATe] is ON.

Command: OUTPut#:NOISe:LEVel <numeric_value>

Query: OUTPut#:NOISe:LEVel?

Response: <numeric_value>

Arguments: <numeric_value 0 to 50

Notes: *The noise has energy from about 750 Hz to 100 MHz. If the OUTPut#:FILTer[:LPASs]:FREQuency is set below 100 MHz much of the noise energy will be filtered out*

OUTPut#:NOISe:PATH

Purpose: Determines whether noise is routed through rear panel connectors for external filtering, or not.

Command: OUTPut#:NOISe:PATH <character data>

Query: OUTPut#:NOISe:PATH?

Response: EXTERNAL or INTERNAL

Arguments:: EXTernal or INTernal

Notes: *OUTP1:NOISE:PATH and OUTP2:NOISE:PATH are coupled. There is one internal noise source, which feeds both channels.*

OUTPut#:NOISe[:STATe]

Purpose: Enables or disables inserting uncorrelated pseudo-random noise into the waveform for the selected channel (1 or 2).

Command: OUTPut#:NOISe <Boolean>

Query: OUTPut#:NOISe?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disables noise.
1 Enables noise.
OFF Disables noise.
ON Enables noise.

OUTPut2[:RESample]

Purpose: Both channels must be at the same clock rate for proper waveform timing. If channel 2 is at a different clock rate then channel 1, this command can used to resample channel 2 so the clock rates are equal. The status operation register can be queried to determine if resampling of channel 2 is necessary. You can determine if resampling is necessary by execution the command STATUS:OPERation:CONDition?

Command: OUTPut2:RESample

Query: None

Response: None

Arguments: None

OUTPut#[:STATe]

Purpose: Enables or disables the output for the selected channel (1 or 2).

Command: OUTPut# <Boolean>

Query: OUTPut#?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Disable the output.
1 Enable the output.
OFF Disable the output.
ON Enable the output.

PROJect:NEW

Purpose: Creates a new project with the specified name. The current project is closed and the new project is opened.

Command: PROJect:NEW <string>

Query: PROJect:NEW?

Response: <string>, the up to 15 character name previously entered for PROJ:NEW. If nothing entered, no response.

Arguments: A quoted string of up to 15 characters

PROJect:OPEN

Purpose: Opens the specified project and closes current project if the specified project exists (no action is taken if it doesn't exist).

Command: PROJect:OPEN <string>

Query: PROJect:OPEN?

Response: <string>

Arguments: <string>

PROJect:SAVE

Purpose: Saves the current project. If a project name other than the current one is given then the current project is save with the new name. The old project is left unchanged. If a name (other than the current project) is given that already exists, then an error message will be displayed and the project will not be saved.

Command: PROJect:SAVE <string>

Query: PROJect:SAVE?

Response: <string>, the up to 15 character name of the current project, or the name entered into PROJECTSAVE from the menus.

Arguments: A quoted string of up to 15 characters.

SEQuence:ADVance

Purpose: To advance to the next sequence in the list. The current sequence will stop where ever it currently is and the next sequence will begin playing. If there is no next sequence the last sequence will continue to play. The channel that is advanced is selected by SEQuence:AON.

Command: SEQuence:ADVance

Query: None

Response: None

Arguments: None

Example: SEQuence:ADVance - advances to the next sequence in the list

SEQuence:AON

Purpose: To select on which channel the SEQuence:ADVance and SEQuence:JUMP commands will operate..

Command: SEQuence:AON channel

Query: SEQuence:AON?

Response: channel, either CH1 or CH2

Arguments: channel, either CH1 or CH2

Example: SEQuence:AON CH1 - sets the advance on to channel 1

SEQuence:COMPile

Purpose: Compiles and executes the sequence in the currently selected editor (Channel 1 or Channel2)

Command: WAVE:SEQuence:COMPile

Query: None

Response: None

Arguments: None

SEQuence:DATA

Purpose: Transfers a sequence file identified by a filename to or from the LW400 via GPIB in #0 block format

Command: SEQuence:DATA "filename", <block>

Query: SEQuence:DATA? "filename"

Response: <indefinite length block>

Arguments: filename is a quoted string of up to 14 characters
<indefinite length block>

Note: An indefinite length block: #0 followed by a sequence list consisiting of up to 512 lines (2048 lines for LW4x0-ME2,1M memory).

SEQuence:GDATa

Purpose: Transfers a group sequence file identified by a filename to or from the LW400 via GPIB in #0 block format.

Command: SEQuence:GDATa "filename", <block>

Query: SEQuence:GDATa? "filename"

Response: <indefinite length block>

Arguments: filename is a quoted string of up to 14 characters
<indefinite length block>

Example: SEQuence:GDATa "example", #0sequence1, sequence2 - Creates a new group sequence named example with two sequences: sequence1 and sequence2.

Note: *An indefinite length block: #0 followed by a sequence list consisting of up to 512 lines (2048 lines for LW400-ME2, 1M memory)*

SEQuence:GLINk

Purpose: This command adds a new line to the end of the sequence list in the currently selected editor.

Command: SEQuence:GLINk filename

Query: SEQuence:GLINk?

Response: filename, returns the last linked sequence name (string).

Arguments: filename, a sequence name to be linked in quotes

Example: SEQuence:GLINk "sequence1" - adds sequence1 to the end of the currently selected group sequence.

SEQuence:GNEW

Purpose: Creates a new group sequence in the currently selected editor with the specified name.

Command: SEQuence:GNEW filename

Query: SEQuence:GNEW?

Response: filename, returns the last name specified by this command.

Arguments: filename, a file name for the new group sequence in quotes.

Example: SEQuence:GNEW "example"

Note: *SEQ:GNEW has no effect on the output of the LW400 until the next :SEQ:COMPile. The new sequence is not saved in the current LW400 project until :SEQ:SAVE is issued.*

SEQuence:IRECall

Purpose: Recalls a stored image file to the specified channel. The image file must have been previously stored using SEQuence:ISAVe. This command decreases the setup time for loading sequences into hardware. Since we're saving the state of hardware there is no compile of the sequence or conversion from floating point to dac codes. Remember this is the state of the hardware when saved. If the sequence that was saved changes, the changes won't be reflected in the image until a new image is stored.

Command: SEQuence:IRECall channel, filename

Query: None

Response: None

Arguments: channel: 1 = channel 1
2 = channel 2
filename: file name of image to recall in quotes. The file name can have eight characters followed by .IMG (must be .IMG).

Example: SEQuence:IRECall 1, "test.img", loads test.img on channel 1.

SEQuence:ISAVe

Purpose: Saves the control memory and high speed memory of the specified channel to disk under the specified filename. The hardware must have either a sequence or a group sequence loaded. Before the image is saved the sequence is compiled to make sure the hardware is up to data. The command SEQuence:IRECall is used to recall the image.

Command: SEQuence:ISAVe channel, filename

Query: None

Response: None

Arguments: channel: 1 = channel 1
2 = channel 2
filename: file name to store the binary image under, in quotes. The file name can have eight characters followed by .IMG (must be .IMG).

Example: SEQuence:ISAVe 1, "test.img", saves test.img from channel 1.

SEQuence:JUMP

Purpose: To jump to a specific sequence in the list. The current sequence will stop where ever it currently is and the sequence specified by the argument will begin playing. The channel that is advanced is selected by SEQuence:AON.

Command: SEQuence:JUMP value

Query: SEQuence:JUMP?

Response: value, returns the current value of jump (not necessarily the sequences being played)

Arguments: value, valid numbers are 1,2, ... up to the number of sequences in the list.

Example: SEQuence:JUMP 2 - jumps to the second sequence in the group sequence

SEQuence:LINK

Purpose: This command adds a new line to the end of the sequence list in the currently selected editor listing the specified waveform and the specified number of repetitions.

Command: WAVE:SEQuence:LINK <string>,<numeric_value>

Query: WAVE:SEQuence:LINK?

Response: <string>,<numeric_value>
returns the last linked waveform name (string) and repetition count (numeric).

Arguments: <string>: a waveform file name in quotes
<numeric_value>: number of repetitions

SEQuence:NEW

Purpose: Clears the sequence list for the currently selected editor and gives it the specified name.

Command: WAVE:SEQuence:NEW <string>

Query: WAVE:SEQuence:NEW?

Response: <string>

Arguments: <string>: a file name for the new sequence file, in quotes

Notes: *WAVE:SEQ:NEW has no effect on the output of the LW4xx until the next :SEQ:COMPile. The new sequence is not saved in the current LW4xx project until :SEQ:SAVE is issued.*

SEQuence:OPEN

Purpose: Opens the specified sequence file in the current project and reads it into the currently selected editor's sequence list.

Command: WAVE:SEQuence:OPEN <string>

Query: WAVE:SEQuence:OPEN?

Response: <string>

Arguments: <string> the name of a sequence file in the current project.

Notes: *The sequence is automatically compiled, therefore :SEQ:COMPile need not be issued.*

SEQuence:SAVE

Purpose: Save a sequence list from the currently selected editor to the current project.

Command: WAVE:SEQuence:SAVE <string>

Query: WAVE:SEQuence:SAVE?

Response: <string>

Arguments: <string> the name of the sequence list to save.

STATus:OPERation:CONDition?

Purpose: Query the contents of the Operation Status condition register. Reading the condition register is nondestructive.

Operation Status Register Bit Assignments

Bit 14: Not Used	(Decimal 16384)
Bit 13: Not Used	(Decimal 8192)
Bit 12: Resample Ch–2 Required	(Decimal 4096)
Bit 11: Not Used	(Decimal 2048)
Bit 10: Sequence compile complete	(Decimal 1024)
Bit 9: Reserved for future use	(Decimal 512)
Bit 8: Reserved for future use	(Decimal 256)
Bit 7: Not Used	(Decimal 128)
Bit 6: Not Used	(Decimal 64)
Bit 5: Waiting for Trigger	(Decimal 32)
Bit 4: Not Used	(Decimal 16)
Bit 3: Not Used	(Decimal 8)
Bit 2: Not Used	(Decimal 4)
Bit 1: Not Used	(Decimal 2)
Bit 0: Not Used	(Decimal 1)

Command: None.

Query: STATus:OPERation:CONDition?

Response: <numeric_value>

Arguments: None

Notes: *The "waiting for trigger" bit is updated by software. It is not guaranteed to transition on every trigger to which the LW400 responds.*

STATus:OPERation:ENABle

Purpose: Set or query the enable mask which allows masked conditions in the event register to be reported in the summary bit. If a bit is 1 (true) in the enable register AND its associated event bit transitions to 1 (true) the associated summary bit will transition to 1 (true).

Operation Status Register Bit Assignments

Bit 14: Not Used	(Decimal 16384)
Bit 13: Not Used	(Decimal 8192)
Bit 12: Resample Ch–2 Required	(Decimal 4096)
Bit 11: Not Used	(Decimal 2048)
Bit 10: Sequence compile complete	(Decimal 1024)
Bit 9: Reserved for future use	(Decimal 512)
Bit 8: Reserved for future use	(Decimal 256)
Bit 7: Not Used	(Decimal 128)
Bit 6: Not Used	(Decimal 64)
Bit 5: Waiting for Trigger	(Decimal 32)
Bit 4: Not Used	(Decimal 16)
Bit 3: Not Used	(Decimal 8)
Bit 2: Not Used	(Decimal 4)
Bit 1: Not Used	(Decimal 2)
Bit 0: Not Used	(Decimal 1)

Command: STATus:OPERation:ENABle <numeric_value>

Query: STATus:OPERation:ENABle?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 32767

STATus:OPERation[:EVENt]?

Purpose: Query the contents of the Operation Status Event register. Reading the Event register will clear the register.

Operation Status Register Bit Assignments

Bit 14: Not Used	(Decimal 16384)
Bit 13: Not Used	(Decimal 8192)
Bit 12: Resample Ch–2 Required	(Decimal 4096)
Bit 11: Not Used	(Decimal 2048)
Bit 10: Sequence compile complete	(Decimal 1024)
Bit 9: Reserved for future use	(Decimal 512)
Bit 8: Reserved for future use	(Decimal 256)
Bit 7: Not Used	(Decimal 128)
Bit 6: Not Used	(Decimal 64)
Bit 5: Waiting for Trigger	(Decimal 32)
Bit 4: Not Used	(Decimal 16)
Bit 3: Not Used	(Decimal 8)
Bit 2: Not Used	(Decimal 4)
Bit 1: Not Used	(Decimal 2)
Bit 0: Not Used	(Decimal 1)

Command: None.

Query: STATus:OPERation?

Response: <numeric_value>

Arguments: None

Notes: *The Waiting for Trigger bit is updated by software and is not guaranteed to detect every occurrence of Waiting for Trigger if the wait is short. Status is checked between command executions and several hundred times a second when the LW400 is idle (that is, not processing commands).*

STATus:PRESet

Purpose: Clears the Operation and Questionable Enable registers and sets positive transactions as the detected events. During power-on the enable registers are set to their STATus:PREset states.

Command: STATus:PRESet

Query: None.

Response: None

Arguments: None

Notes: *See also *CLS*

STATus:QUEStionable:CONDition?

Purpose: Query the contents of the Questionable Status Condition register. Reading the Condition register is non-destructive.

Questionable Status Register Bit Assignments

Bit 14:	Command Warning	(Decimal 16384)
Bit 13:	Not Used	(Decimal 8192)
Bit 12:	Not Used	(Decimal 4096)
Bit 11:	Not Used	(Decimal 2048)
Bit 10:	Not Used	(Decimal 1024)
Bit 9:	Not Used	(Decimal 512)
Bit 8:	Not Used	(Decimal 256)
Bit 7:	Not Used	(Decimal 128)
Bit 6:	Not Used	(Decimal 64)
Bit 5:	Not Used	(Decimal 32)
Bit 4:	Not Used	(Decimal 16)
Bit 3:	Not Used	(Decimal 8)
Bit 2:	Not Used	(Decimal 4)
Bit 1:	Not Used	(Decimal 2)
Bit 0:	Not Used	(Decimal 1)

Command: None.

Query: STATus:QUEStionable:CONDition?

Response: <numeric_value>

Arguments: None

STATus:QUEStionable:ENABle

Purpose: Set or query the enable mask which allows masked conditions in the event register to be reported in the summary bit. If a bit is 1 (true) in the enable register AND its associated event bit transitions to 1 (true) the associated summary bit will transition to 1 (true).

Questionable Status Register Bit Assignments

Bit 14:	Command Warning	(Decimal 16384)
Bit 13:	Not Used	(Decimal 8192)
Bit 12:	Not Used	(Decimal 4096)
Bit 11:	Not Used	(Decimal 2048)
Bit 10:	Not Used	(Decimal 1024)
Bit 9:	Not Used	(Decimal 512)
Bit 8:	Not Used	(Decimal 256)
Bit 7:	Not Used	(Decimal 128)
Bit 6:	Not Used	(Decimal 64)
Bit 5:	Not Used	(Decimal 32)
Bit 4:	Not Used	(Decimal 16)
Bit 3:	Not Used	(Decimal 8)
Bit 2:	Not Used	(Decimal 4)
Bit 1:	Not Used	(Decimal 2)
Bit 0:	Not Used	(Decimal 1)

Command: STATus:QUEStionable:ENABle <numeric_value>

Query: STATus:QUEStionable:ENABle?

Response: <numeric_value>

Arguments: <numeric_value>

STATus:QUEStionable[:EVENt]?

Purpose: Query the contents of the Questionable Status Event register. Reading the Event register clears the register.

Questionable Status Register Bit Assignments

Bit 14:	Command Warning	(Decimal 16384)
Bit 13:	Not Used	(Decimal 8192)
Bit 12:	Not Used	(Decimal 4096)
Bit 11:	Not Used	(Decimal 2048)
Bit 10:	Not Used	(Decimal 1024)
Bit 9:	Not Used	(Decimal 512)
Bit 8:	Not Used	(Decimal 256)
Bit 7:	Not Used	(Decimal 128)
Bit 6:	Not Used	(Decimal 64)
Bit 5:	Not Used	(Decimal 32)
Bit 4:	Not Used	(Decimal 16)
Bit 3:	Not Used	(Decimal 8)
Bit 2:	Not Used	(Decimal 4)
Bit 1:	Not Used	(Decimal 2)
Bit 0:	Not Used	(Decimal 1)

Command: None.

Query: STATus:QUEStionable?

Response: <numeric_value>

Arguments: None

SYSTem:CLOCk:EREFerence

Purpose: Enable or disable External Reference in (10 MHz clock reference)

Command: SYSTem:CLOCk:EREFerence <Boolean>

Query: SYSTem:CLOCk:EREFerence?

Response: <Boolean>

Arguments: INT, EXT

INT Enables internal reference.
EXT Enables external reference.

SYSTem:COMMunicate:GPIB[:SELF]:ADDRESS

Purpose: Set or query the GPIB address setting of the arbitrary function generator. The default address setting is 1 and any setting in the range from 1 - 30 may be specified.

Command: SYSTEM:COMMunicate:gpib:address <numeric_value>

Query: SYSTem:COMMunicate:GPIB:ADDRess?

Response: <numeric_value>

Arguments: <numeric_value> GPIB address of the instrument. Must be between (1-30).

Notes: *This command takes effect immediately. Any further communication over GPIB must address the LW400 at its new address.*

SYSTem:ERRor?

Purpose: Query up to the last three system errors, most recent first. The result of the Query is the error number followed by the error text for the next most recent system error.

Command: None.

Query: SYSTem:ERRor?

Response: <numeric_value>,<string> for example, 0, "No Error"

Arguments: None

SYSTem:HELP:SYNTax?

Purpose: Find out the full command header and argument types for a known command header.

Command: None.

Query: SYSTem:HELP:SYNTax? <string>

Response: <string>

Arguments: None

Notes: *Example: system:help:syntax? "OUTP1:FILT:FREQ"*
returns ":OUTPut1:FILTer[:LPASs]:FREQuency <numeric_value>"
For query only headers the '?' must be included, for example,
system:help:syntax? "system:help:syntax?"

SYSTem:VERSion?

Purpose: Read out what version of SCPI the instrument uses.

Command: None.

Query: SYSTem:VERSion?

Response: 1993.0

Arguments: None

TRIGger[:SEQuence]:BCOunt

Purpose: Sets the number of repetitions of the waveform that will be played after a trigger is received in burst mode.

Command: TRIGger:BCOunt <numeric_value>

Query: TRIGger:BCOunt?

Response: <numeric_value>

Arguments: <numeric_value> Burst count (1 - 4095)

TRIGger[:SEQuence]:DELay

Purpose: Sets the delay from trigger to start of output of the waveform.

Command: TRIGger:DELay <numeric_value>

Query: TRIGger:DELay?

Response: <numeric_value>

Arguments: <numeric_value> Trigger delay (0s - max).

Notes: *The maximum delay depends on the clock frequency. It is over 4.29 billion clocks.*

TRIGger[:SEQuence]:LEVel

Purpose: Set or query the trigger level. The trigger level is specified in volts.

Command: TRIGger:LEVel <numeric_value>

Query: TRIGger:LEVel?

Response: <numeric_value>

Arguments: <numeric_value> Trigger level (± 2.5 volts).

Notes: *The resolution of the trigger level is 20 mV steps. The value will be rounded to the nearest multiple of 20 mV.*

TRIGger[:SEQuence]:MODE

Purpose: Set or query the trigger mode. The trigger mode may set to CONTinuous, SINGle, BURSt, or GATE. Continuous will continually play the waveform regardless of trigger state. Single will play one repetition of the waveform after a trigger is received. Burst will play a burst count number of repetitions of the waveform after a trigger is received. Gate will continuously play the waveform as long as the trigger input is true.

Command: TRIGger:MODE <character_data>

Query: TRIGger:MODE?

Response: <character_data>

Arguments: one of: CONTinuous, SINGle, BURSt, GATE

BURSt	Select burst trigger.
CONTinuous	Select continuous trigger.
GATE	Select gate trigger
SINGle	Select single trigger.

TRIGger[:SEQuence]:SLOPe

Purpose: Set or query the trigger slope. If the trigger slope is set to positive a trigger is generated when the signal crosses the trigger threshold (level) in a positive going direction. If the trigger slope is set to negative a trigger is generated when the signal crosses the trigger threshold in a negative going direction. Gate is true if trigger input is above the trigger level for positive slope or below trigger level for negative slope.

Command: TRIGger:SLOPe <character_data>

Query: TRIGger:SLOPe?

Response: <character_data>

Arguments: one of: POSitive, NEGative

NEGative	Trigger on negative going edge.
POSitive	Trigger on positive going edge.

TRIGger[:SEQuence]:SOURCE

Purpose: Set or query the trigger source. The trigger source selection is internal or external.

Command: TRIGger:SOURCE <character_data>

Query: TRIGger:SOURCE?

Response: <character_data>

Arguments: one of: YES, NO

NO	Internal trigger selected
YES	External trigger selected

WAVE:AMPLitude:AMPLitude

Purpose: Sets the peak to peak amplitude of the region between the left and right time cursors. The amplitude is grown around a baseline that is defined by a line drawn from the voltage point under the left cursor to the voltage point under the right cursor.

Command: WAVE:AMPLitude:AMPLitude <numeric_value>

Query: WAVE:AMPLitude:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value>, 100 uV - 10 V

WAVE:AMPLitude:INVert

Purpose: Inverts the portion of the selected waveform between the time cursors

Command: WAVE:AMPLitude:INVert

Query: None

Response: None

Arguments: None

Note: *Added in firmware version 2.0 or higher*

WAVE:AMPLitude:MEDian

Purpose: Sets the median voltage level of the region between the left and right time cursor, where median is defined as Vbottom + 1/2 the peak to peak amplitude of the region.

Command: WAVE:AMPLitude:MEDian <numeric_value>

Query: WAVE:AMPLitude:MEDian?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5

WAVE:AMPLitude:VMAX

Purpose: Sets the maximum voltage of the region between the left and right time cursors. The VMAX only changes the maximum voltage the minimum voltage is left unchanged.

Command: WAVE:AMPLitude:VMAX <numeric_value>

Query: WAVE:AMPLitude:VMAX?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5

WAVE:AMPLitude:VMIN

Purpose: Sets the minimum voltage of the region between the left and right time cursors. The VMIN only changes the minimum voltage the maximum voltage is left unchanged.

Command: WAVE:AMPLitude:VMIN <numeric_value>

Query: WAVE:AMPLitude:VMIN?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5

Clock and Filter Ranges

This table of clock and filter ranges is for reference when using the WAVE: CLOCK commands.

Decade	Lower Limit	Upper Limit	Filter
400 MHz	355 MHz	400 MHz	100 MHz
40 MHz	35.5 MHz	40 MHz	10 MHz
4 MHz	3.55 MHz	4 MHz	1 MHz
400 kHz	355 kHz	400 kHz	100 kHz
40 kHz	35.5 kHz	40 kHz	10 kHz

Note: The LW400A series provides a continuously variable clock from 6 kHz to 400 MHz when the WAVE:CLOCK:LIMit NO is issued.

WAVE:CLOCk:ACSet

Purpose: Set to YES and the WaveStation automatically selects the best sample clock rate to achieve the required duration of the waveform. Set to NO and the clock is held at the user set frequency while the number of samples is varied to set the waveform duration.

Command: WAVE:CLOCk:ACSet <character_data>

Query: WAVE:CLOCk:ACSet?

Response: <character_data>

Arguments: YES or NO

YES—automatic selection of sample clock rate.
NO—held at user set frequency.

WAVE:CLOCk:DECade

Purpose: Selects the clock decade in which the internal clock runs. The choices are 40 kHz, 400 kHz, 4 MHz, 40 MHz, 400 MHz.

Command: WAVE:CLOCk:DECade <numeric_value>

Query: WAVE:CLOCk:DECade?

Response: <numeric_value>

Arguments: <numeric_value> 400e6, 40e6, 4e6, 400e3, 40e3.

Note: *This command is included for backwards compatability. It is recommended that the WAVE:CLOCk:MAX command be used.*

WAVE:CLOCk:FIXed

Purpose: Selects whether the clock is fixed or variable. If the clock is variable then the system may change the clock. If the clock is fixed then the system will not change the clock. *Note: see 'Clock and Filter Ranges' on page 10-8.*

Command: WAVE:CLOCk:FIXed <character_data>

Query: WAVE:CLOCk:FIXed?

Response: <character_data>

Arguments: VARiable/FIXed

VAR—allows WaveStation to change clock.
FIX—WaveStation not allowed to change clock.

Note: *This command is included for backwards compatibility. It is recommended that the* WAVE:CLOCk:ACSet *command be used.*

WAVE:CLOCk:FREQuency

Purpose: This is the frequency—clock rate—at which the clock is fixed (see WAVE:CLOCk:FIXed). If WAVE:CLOCk:FIXed is VARiable, this sets the clock frequency but subsequent edit operations may change the clock frequency.

Command: WAVE:CLOCk:FREQuency <numeric_value>

Query: WAVE:CLOCk:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>

WAVE:CLOCk:LIMit

Purpose: To limit the clock setting to the frequency ranges covered by the internal filter ranges or allow full range of the clock. Setting to YES limits the clock to the frequency ranges covered by the internal filters. Setting to NO allows control of the clock over the full range.

Command: WAVE:CLOCk:LIMit <character_data>

Query: WAVE:CLOCk:LIMit?

Response: <character_data>

Arguments: YES or NO
YES Limit to internal filters.
NO Allow continuously variable clock.

Note: This command applies to the LW400A series not the LW400 series.

WAVE:CLOCk:MAX

Purpose: When Limit Clock field is set to YES, this command is used to select the clock decade in which the internal clock runs. The choices are 40 kHz, 400 kHz, 4 MHz, 40 MHz, 400 MHz. When the Limit Clock field is set to NO, this is a query only command.

Command: WAVE:CLOCk:MAXt <numeric_value>

Query: WAVE:CLOCk:MAX?

Response: <numeric_value>

Arguments: <numeric_value> 400e6, 40e6, 4e6, 400e3, 40e3

Note: This command applies to the LW400A series not the LW400 series.

WAVE:CUT:COPY

Purpose: Copies the region between the right and left time cursors and store the data to the cut buffer. All data under and between the time cursors is copied.

Command: WAVE:CUT:COPY

Query: None

Response: None

Arguments: None

WAVE:CUT:DELete

Purpose: Copies the data between the left and right time cursors to the cut buffer and deletes the data from the waveform. All data under and between the time cursors is deleted.

Command: WAVE:CUT:DELete

Query: None

Response: None

Arguments: None

WAVE:CUT:EXTRact

Purpose: Copies the value of the waveform minus the value of the baseline to the cut buffer. What is left in the waveform is the value of the baseline. The baseline is defined by a line drawn from the voltage point under the left cursor to the voltage point under the right cursor. When pasted back into the waveform, extracted data is always summed with the selected region of the waveform.

Command: WAVE:CUT:EXTRact

Query: None

Response: None

Arguments: None

WAVE:DATA

Purpose: Used to read out the currently selected waveform or to read in a new waveform as a DIF expression.

Command: WAVE:DATA <block>

Query: WAVE:DATA?

Response: <block>

Arguments: <block>

Notes: *The "<block>" is in Data Interchange Format (DIF). See Chapter 5 for a detailed description.*

When using the WAVE:DATA <block> command to transfer a waveform to the WaveStation, data must follow immediately(hold off eoi). See section 7—Remote Programming examples.

WAVE:DATA:PREamble?

Purpose: Read out the DIF expression describing the currently selected waveform, containing everything except the data values.

Command: None

Query: WAVE:DATA:PREamble?

Response: <block>

Arguments: None

Notes: *The "<block>" is in Data Interchange Format (DIF). See Chapter 5 for a detailed description.*

WAVE:DIGital:DURation: POINts

Purpose: Sets the duration of the inserted waveform in sample points

Command: WAVE:DIGital:DURation:POINts <numeric_value>

Query: WAVE:DIGital:DURation:POINts ?

Response: <numeric_value>

Arguments: <numeric_value> 1 to the maximum memory length

WAVE:DIGital:DURation [:TIME]

Purpose: Sets the duration of the inserted waveform in time

Command: WAVE:DIGital:DURation:TIME <numeric_value>

Query: WAVE:DIGital:DURation:TIME ?

Response: <numeric_value>

Arguments: <numeric_value> Clock period to the maximum memory length times the clock period

WAVE:DIGital:FMASk

Purpose: Sets the mask value used to select desired bits which are then set using the WAVE:DIGital:SMValue command.

Command: WAVE:DIGital:FMASk <numeric_value>

Query: WAVE:DIGital:FMASk ?

Response: <numeric_value>

Arguments: <numeric_value> 0-255

Notes: *The numeric value may be specified in decimal, hexadecimal, or binary: 0-255 in decimal, #h00 - #hFF in hexadecimal, or #b00000000 - #b11111111 in binary.*

WAVE:DIGital:LCURsor:POINts

Purpose: Sets the position of the Time Left cursor in sample points. This is the position at which the digital values (SVALue or SMValue) will be inserted.

Command: WAVE:DIGital:LCURSOR:POINts <numeric_value>

Query: WAVE:DIGital:LCURSOR:POINts?

Response: <numeric_value>

Arguments: <numeric_value> 0 to maximum number of points in the waveform

WAVE:DIGital:LCURsor:[TIME]

Purpose: Sets the position of the Time Left cursor in time. This is the position at which the digital values (SVALue or SMValue) will be inserted.

Command: WAVE:DIGital:LCURSOR:TIME <numeric_value>

Query: WAVE:DIGital:LCURSOR:TIME?

Response: <numeric_value>

Arguments: <numeric_value> 0 to the waveform time duration

WAVE:DIGital:MODE

Purpose: Sets the mode in which sections of waveforms are inserted. The two modes are insert and overwrite. Insert places the new section at the left time cursor and moves all the data to the right of the cursor (not including the point under the left cursor) by the length of the inserted section. Overwrite places the new section at the left cursor and overwrites existing data in the waveform.

Command: WAVE:DIGital:MODE <character_data>

Query: WAVE:DIGital:MODE?

Response: <character_data>

Arguments: INSert or OVERwrite

WAVE:DIGital:MVALue

Purpose: Sets the value of the masked bits selected by the WAVE:DIGital:FMASk command.

Command: WAVE:DIGital:MVALue <numeric_value>

Query:: WAVE:DIGital:MVALue?

Response: <numeric_value>

Arguments: <numeric_value> 0 to $2^{(\#\text{of 1's in FMASk value})} - 1$

Notes: *If the mask value were 200 (11001000) then the masked value can be set in the range 0 to 7 (2^3 -1). The numeric value can be entered in decimal, hexadecimal or binary formats: 0-max in decimal, #h00 - #hmax in hexadecimal, or #b00000000 - #bmax in binary, where max represents the maximum range in the selected format.*

WAVE:DIGital:SMValue

Purpose: In the overwrite mode the specified masked value, set using the WAVE:DIGitial:MVALue command, is "ORed" into the waveform, for the duration set in WAVE:DIGital:Duration, starting at the location specified in WAVE:DIGital:LCURsor. In the insert mode the masked value is simply inserted into the waveform starting at the Time Left cursor location. In either mode the binary weight of the inserted bits is restored to their original values.

Command: WAVE:DIGital:SMValue

Query: None

Response: None

Arguments: None

Notes: *If the mask was 200 (11001000) and the masked value was 5, then in insert mode the value 136 (128 + 8) or 10001000 would be inserted into the waveform at the left cursor.*

WAVE:DIGital:SVALue

Purpose: Inserts or overwrites the value, set using the WAVE:DIGital:VALue command, into the waveform, for the duration set in WAVE:DIGital:Duration, starting at the location specified in WAVE:DIGital:LCURsor.

Command: WAVE:DIGital:SVALue

Query: None

Response: None

Arguments: None

WAVE:DIGital:VALue

Purpose: Sets the value to be inserted using the WAVE:DIGital:SMValue command.

Command: WAVE:DIGital:VALue <numeric_value>

Query: WAVE:DIGital:VALue?

Response: <numeric_value>

Arguments: <numeric_value> 0-255

Notes: *The numeric value may be entered in decimal, hexadecimal, or binary: 0 -255 in decimal, #h00 - #hFF in hexadecimal, or #b00000000 - #b11111111 in binary.*

WAVE:INSert:CURSor

Purpose: Set to insert new waveform sections before or after the left time cursor. If BEFore is selected, after the new section is inserted the left cursor is moved to the end of the inserted section, leaving the inserted section before the left cursor. If AFTer is selected, the left cursor is not moved after the section is inserted.

Command: WAVE:INSert:CURSor <character_data>

Query: WAVE:INSert:CURSor?

Response: <character_data>

Arguments: BEFore or AFTer

WAVE:INSert:MODE

Purpose: Sets the mode in which sections of waveforms are inserted. The two modes are insert and overwrite. Insert places the new section at the left time cursor and moves all the data to the right of the cursor (not including the point under the left cursor) by the length of the inserted section. Overwrite places the new section at the left cursor and overwrites existing data in the waveform.

Command: WAVE:INSert:MODE <character_data>

Query: WAVE:INSert:MODE?

Response: <character_data>

Arguments: INSert or OVERwrite

WAVE:INSert:OVERsample

Purpose: Used to select the setting for the "Oversample Wave" option. If set to YES the currently selected waveform will be checked for discontinuities, and if found, will be fixed by passing the discontinuity through a low pass filter. If NO is chosen, the data will not be checked for discontinuities.

Command: WAVE:INSert:OVERsample <character_data>

Query: WAVE:INSert:OVERsample?

Response: <character_data>

Arguments: YES or NO

YES oversample
NO do not oversample

WAVE:INSert:PASTe:COUNt

Purpose: Sets the number of times that the data in the cut buffer is inserted into the waveform.

Command: WAVE:INSert:PASTe:COUNt <numeric_value>

Query: WAVE:INSert:PASTe:COUNt?

Response: <numeric_value>

Arguments: <numeric_value> 1 to 32767

WAVE:INSert:PASTe[:IMMediate]

Purpose: Inserts the cut buffer into waveform WAVE:INSert:PASTe:COUNt times at the left time cursor in the edit mode described by WAVE:INSert:MODE. If the data was extracted the it is summed back into the waveform.

Command: WAVE:INSert:PASTe

Query: None

Response: None

Arguments: None

WAVE:INSert:SCOPe:ADDRess

Purpose: Sets which GPIB address the digital oscilloscope that data is to be downloaded from is using.

Command: WAVE:INSert:SCOPe:ADDRess <numeric_value>

Query: WAVE:INSert:SCOPe:ADDRess?

Response: <numeric_value> 0 - 30

Arguments: <numeric_value> 0 - 30

WAVE:INSert:SCOPe:BWLimit

Purpose: Set to YES and the LW400 will check for, and oversample to eliminate discontinuities on the currently selected waveform.

Command: WAVE:INSert:SCOPe:BWLimit <character_data>

Query: WAVE:INSert:SCOPe:BWLimit?

Response: <character_data>

Arguments: YES or NO
YES to Bandwidth limit
NO do not bandwidth limit

WAVE:INSert:SCOPe:CONTrol

Purpose: Tells the LW400 whether there is a controller active on the bus from which it must request control to do :WAVE:INSert:SCOPe[:IMMediate].

Command: WAVE:INSert:SCOPe:CONTrol <Boolean>

Query: WAVE:INSert:SCOPe:CONTrol?

Response: <Boolean>

Arguments: YES, NO

NO no active controller.
YES active controller.

Notes: *If set to 'YES'; the LW400 will request control when :WAVE:INS:SCOPE:IMMediate is executed and will return control when it is done. The controller must be capable of supporting 488.2' pass control protocol (IEEE Std 488.2-1992 (Section 17.4).*

WAVE:INSert:SCOPe:PREServe

Purpose: Sets how the data from the digital oscilloscope is preserved. The data can be preserved in time or by points. If time is selected then the data will be resampled to preserve the overall time. If points is selected then the data are not resampled and the points are inserted into the waveform at the current clock. Timing at the output will probably be incorrect.

Command: WAVE:INSert:SCOPe:PREServe <character_data>

Query: WAVE:INSert:SCOPe:PREServe?

Response: <character_data>

Arguments: POINts or TIme

WAVE:INSert:SCOPe:SOURce

Purpose: Sets the location in the digital oscilloscope to download the data from. The available choices depend on the oscilloscope's capabilities. Please refer to the choices on the FROM SCOPE menu under Trace Source for the available source for your scope (make sure the oscilloscope in question is selected in the DSO Type list). The trace source must be typed exactly as shown in the menu field (including spaces) and enclosed in quotes.

Command: WAVE:INSert:SCOPe:SOURce <string>

Query: WAVE:INSert:SCOPe:SOURce?

Response: <string>

Arguments: <string>

WAVE:INSert:SCOPe:TYPE

Purpose: Sets which digital oscilloscope the data will be downloaded from. The LW400 initially supports the scopes listed under Arguments. Additional oscilloscopes may be added through project import. Use the name found in the FROM SCOPE menu under DSO Type to select a different oscilloscope than the ones listed below.

Command: WAVE:INSert:SCOPe:TYPE <string>

Query: WAVE:INSert:SCOPe:TYPE?

Response: <string>

Arguments: <string>

WAVE:INSert:SCOPe[:IMMediate]

Purpose: Downloads the data from the digital oscilloscope defined by WAVE:INSert:SCOPe:TYPE, at GPIB address WAVE:INSert:SCOPe:ADDRess and retrieved from source WAVE:INSert:SCOPe:SOURce. The data will be preserved using WAVE:INSert:SCOPe:PREServe. The captured data will then be inserted into the waveform at the left time cursor using the edit mode WAVE:INSert:MODE. The LW400 must become controller to perform this operation, see WAVE:INSert:SCOPe:CONTrol.

Command: WAVE:INSert:SCOPe

Query: None

Response: None

Arguments: None

WAVE:INSert:SHAPe:DC:DURation

Purpose: Set the length of time of the DC function will be inserted by WAVE:INSert:SHAPe[:IMMediate], if DC has been selected by WAVE:INSert:SHAPe:SELect.

Command: WAVE:INSert:SHAPe:DC:DURation <numeric_value>

Query: WAVE:INSert:SHAPe:DC:DURation?

Response: <numeric_value>

Arguments: <numeric_value> 10 ns to 1 S

WAVE:INSert:SHAPe:DC:LEVel

Purpose: Set the DC voltage level which will be inserted by WAVE:INSert:SHAPe[:IMMediate] if DC is selected by WAVE:INSert:SHAPe:SELect.

Command: WAVE:INSert:SHAPe:DC:LEVel <numeric_value>

Query: WAVE:INSert:SHAPe:DC:LEVel?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V, resolution 1 mV

WAVE:INSert:SHAPe:PULSe:AMPLitude

Purpose: Sets the base to top amplitude of the pulse.

Command: WAVE:INSert:SHAPe:PULSe:AMPLitude <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> -10 to +10

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate.*

WAVE:INSert:SHAPe:PULSe:BASE

Purpose: Sets the voltage of the non-triggered level of the pulse.

Command: WAVE:INSert:SHAPe:PULSe:BASE <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:BASE?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate.*

WAVE:INSert:SHAPe:PULSe:CYCLes

Purpose: Sets the number of cycles that will be inserted into the waveform. The duration of the inserted section will be CYCLes * PERiod + TDELay.

Command: WAVE:INSert:SHAPe:PULSe:CYCLes <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:CYCLes?

Response: <numeric_value>

Arguments: <numeric_value> 0.01 to 65536

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate.*

WAVE:INSert:SHAPe:PULSe:ETIMe

Purpose: The 10%-90% edge time of both the rising and falling edges of the pulse.

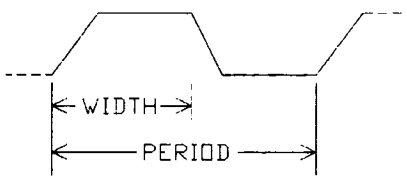
Command: WAVE:INSert:SHAPe:PULSe:ETIMe <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:ETIMe?

Response: <numeric_value>,

Arguments: <numeric_value>, limits depend on clock decade.
400 MHz: 5 ns to 510 ns
40 kHz: 50 µs to 5.1 ms

Notes: *The time to transition from base to top (0 to 100%) will be approximately 100/80 x ETIMe, or 1.25 x ETIMe. As shown in the diagram below, ...PULSe:WIDth +1.25xETIMe must be <=...PULSe:PERiod, or the pulse cannot be produced.*



See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate

WAVE:INSert:SHAPe:PULSe:PERiod

Purpose: Sets the period (1/frequency) of the pulse train which will be inserted into the waveform.

Command: WAVE:INSert:SHAPe:PULSe:PERiod <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:PERiod?

Response: <numeric_value>

Arguments: <numeric_value> limits depend on clock decade
400 MHz: 10 ns to 2.5 ms, 0.1 ns resolution
40 kHz: 100 us to 25 seconds, 1 us resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:PULSe:TDELay

Purpose: Sets the amount of time between the beginning of the waveform and the beginning of the first edge of the pulse.

Command: WAVE:INSert:SHAPe:PULSe:TDELay <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:TDELay?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade
400 MHz: 0 to 2.5 mS, 0.1 ns resolution
40 kHz: 0 to 25 sec, 1 us resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:PULSe:WIDTh

Purpose: Sets the width of the pulse from 50% up the rising edge to 50% down the falling edge.

Command: WAVE:INSert:SHAPe:PULSe:WIDTh <numeric_value>

Query: WAVE:INSert:SHAPe:PULSe:WIDTh?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade
400 MHz: 0 to 2.5 ms, 0.1 ns resolution
40 kHz: 0 to 25 sec, 1 us resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:AMPLitude

Purpose: Sets the peak to peak amplitude of the ramp which will be inserted into the waveform.

Command: WAVE:INSert:SHAPe:RAMP:AMPLitude <numeric_value>

Query: WAVE:INSert:SHAPe:RAMP:AMPLitude?

Response: <numeric_value>

Arguments: 0 to 10 V

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:CYCLes

Purpose: The number of cycles of the ramp that are inserted into the waveform.

Command: WAVE:INSert:SHAPe:RAMP:CYCLes <numeric_value>

Query: WAVE:INSert:SHAPe:RAMP:CYCLes?

Response: <numeric_value>

Arguments: .01 to 65536

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:FREQuency

Purpose: Sets the frequency of the ramp.

Command: WAVE:INSert:SHAPe:RAMP:FREQuency <numeric_value>

Query: WAVE:INSert:SHAPe:RAMP:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade

400 MHz:	400 Hz to 25 MHz,	1 Hz resolution
40 kHz	0.04 Hz to 2.5 kHz	0.0001 Hz

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:INVert

Purpose: Controls whether the ramp is rising or falling.

Command: WAVE:INSert:SHAPe:RAMP:INVert <Boolean>
Query: WAVE:INSert:SHAPe:RAMP:INVert?

Response: <Boolean>

Arguments: one of: 0, 1, OFF, ON

0 Normal.
1 Inverted.
OFF Normal.
ON Inverted.

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:OFFSet

Purpose: Set the voltage of the zero degree phase of the ramp.

Command: WAVE:INSert:SHAPe:RAMP:OFFSet <numeric_value>

Query: WAVE:INSert:SHAPe:RAMP:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V, resolution 1 mV

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:RAMP:SPOSition

Purpose: Sets the start position of the ramp in percentage of the ramp slope.

Command: WAVE:INSert:SHAPe:RAMP:SPOSition <numeric_value>

Query: WAVE:INSert:SHAPe:RAMP:SPOSition?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 100, resolution 0.001

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SELect

Purpose: Selects which shape will be inserted into the waveform by WAVE:INSert:SHAPe:IMMediate.

Command: WAVE:INSert:SHAPe:SELect <character_data>

Query: WAVE:INSert:SHAPe:SELect?

Response: <character_data>

Arguments: DC/PULSE/RAMP/SINE/SQUare/TRIangle

WAVE:INSert:SHAPe:SINE:AMPLitude

Purpose: Sets the peak to peak amplitude of the sine wave.

Command: WAVE:INSert:SHAPe:SINE:AMPLitude <numeric_value>

Query: WAVE:INSert:SHAPe:SINE:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 10 V, 1 mV resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SINE:CYCLes

Purpose: The number of cycles of a sine waves that will be inserted into the waveform.

Command: WAVE:INSert:SHAPe:SINE:CYCLes <numeric_value>

Query: WAVE:INSert:SHAPe:SINE:CYCLes?

Response: <numeric_value>

Arguments: <numeric_value> 0.01 to 65536, resolution 0.01

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SINE:FREQuency

Purpose: Sets the frequency of the sine wave.

Command: WAVE:INSert:SHAPe:SINE:FREQuency <numeric_value>

Query: WAVE:INSert:SHAPe:SINE:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade

400 MHz:	400 Hz to 100 MHz,	1 Hz resolution
40 kHz:	.04 Hz to 10 kHz.	0.0001 Hz resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SINE:OFFSet

Purpose: Set the voltage of the zero degree phase of the sine wave.

Command: WAVE:INSert:SHAPe:SINE:OFFSet <numeric_value>

Query: WAVE:INSert:SHAPe:SINE:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V, 1 mV resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SINE:PHASe

Purpose: Sets the start phase of the sine wave.

Command: WAVE:INSert:SHAPe:SINE:PHASe <numeric_value>

Query: WAVE:INSert:SHAPe:SINE:PHASe?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 360 (degrees), 0.05 degrees resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SQUare:AMPLitude

Purpose: Sets the peak to peak amplitude of the square wave.

Command: WAVE:INSert:SHAPe:SQUare:AMPLitude <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 10 V, resolution 1 mV

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SQUare:BASE

Purpose: Sets the voltage of the non-triggered level of the square wave.

Command: WAVE:INSert:SHAPe:SQUare:BASE <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:BASE?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V, 1 mV resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SQUare:CYCLes

Purpose: The number of cycles of the square wave that will be inserted into the waveform.

Command: WAVE:INSert:SHAPe:SQUare:CYCLes <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:CYCLes?

Response: <numeric_value>

Arguments: <numeric_value> 0.01 to 65536, 0.01 resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SQUare:ETIMe

Purpose: The 10%-90% edge time of both the rising and falling edges of the square wave.

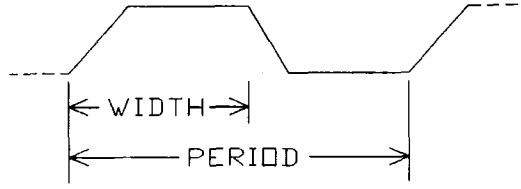
Command: WAVE:INSert:SHAPe:SQUare:ETIMe <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:ETIMe?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade

400 MHz:	5 ns to 510 ns
40 kHz:	50 µs to 5.1 ms



Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate.*

The time to transition from base to top (0 to 100%) will be approximately 100/80 x ETIMe, or 1.25 x ETIMe. +1.25xETIMe must be <= 0.5/SQUARe:FREQuency or the waveform can not be produced.

WAVE:INSert:SHAPe:SQUare:FREQuency

Purpose: Sets the frequency of the square wave. (Period is 1/Frequency)

Command: WAVE:INSert:SHAPe:SQUare:FREQuency <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade

400 MHz:	400 Hz to 50 MHz,	resolution 1 Hz
40 kHz:	.04 Hz to 5 kHz,	resolution 0.0001 Hz

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:SQUare:TDELay

Purpose: Sets the amount of time before the first edge of the square wave.

Command: WAVE:INSert:SHAPe:SQUare:TDELay <numeric_value>

Query: WAVE:INSert:SHAPe:SQUare:TDELay?

Response: <numeric_value>

Arguments: <numeric_value>

Notes *TDELay adds time before the beginning of the first rising edge. After that the number of 50% duty cycle pulses specified by ...SQUare:CYCLes are inserted as specified by ...SQUare:FREQuency, etc.*

See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate

WAVE:INSert:SHAPe:TRIangle:AMPLitude

Purpose: Sets the peak to peak amplitude of the triangle wave.

Command: WAVE:INSert:SHAPe:TRIangle:AMPLitude <numeric_value>

Query: WAVE:INSert:SHAPe:TRIangle:AMPLitude?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 10 V, 1 mV resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:TRIangle:CYCLes

Purpose: The number of cycles that will be inserted into the waveform by WAVE:INSert:SHAPe[:IMMediate].

Command: WAVE:INSert:SHAPe:TRIangle:CYCLes <numeric_value>

Query: WAVE:INSert:SHAPe:TRIangle:CYCLes?

Response: <numeric_value>

Arguments: <numeric_value> .01 to 65536, .01 resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:TRIangle:FREQuency

Purpose: Sets the frequency of the triangle wave which will be inserted by :WAVE:INSert:SHAPe[:IMMediate].

Command: WAVE:INSert:SHAPe:TRIangle:FREQuency <numeric_value>

Query: WAVE:INSert:SHAPe:TRIangle:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value>, limits depend on clock decade

400 MHz:	400 Hz to 25 MHz,	1 Hz resolution
40 kHz:	0.04 Hz to 2.5 kHz	0.0001 Hz resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:TRIangle:OFFSet

Purpose: Set the median voltage of the triangle.

Command: WAVE:INSert:SHAPe:TRIangle:OFFSet <numeric_value>

Query: WAVE:INSert:SHAPe:TRIangle:OFFSet?

Response: <numeric_value>

Arguments: <numeric_value> -5 to +5 V, 1 mV resolution

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe:TRIangle:PHASe

Purpose: Phase of the triangle wave.

Command: WAVE:INSert:SHAPe:TRIangle:PHASe <numeric_value>

Query: WAVE:INSert:SHAPe:TRIangle:PHASe?

Response: <numeric_value>

Arguments: <numeric_value> 0 to 360 (degrees). 0.05 degree resolution.

Notes: *See also WAVE:INSert:SHAPe:SELect and WAVE:INSert:SHAPe:IMMediate*

WAVE:INSert:SHAPe[:IMMediate]

Purpose: Inserts the selected shape (WAVE:INSert:SHAPe:SELect) at the left time cursor using the edit mode defined by WAVE:INSert:MODE.

Command: WAVE:INSert:SHAPe

Query: None

Response: None

Arguments: None

WAVE:INSert:WAVE

Purpose: Insert the named waveform into the current waveform at the TIME LEFT cursor, using the edit mode defined by WAVE:INSert:MODE and WAVE:INSert:CURSor.

Command: WAVE:INSert:WAVE <string>

Query: None

Response: None

Arguments: <string>
Name of the waveform to insert, in quotes.
Example: WAVE:INSert:WAVE "default a"

WAVE:INSert:WRAP

Purpose: Select YES and the waveform will be treated as a continuous wave—the last point wraps to the first point and the waveform is checked for discontinuities between the end and the beginning of the waveform. Select NO if the waveform is only to be played once (single shot) or, is part of a sequence where wrapping the ends might be an incorrect thing to do.

Command: WAVE:INSert: WRAP <character_data>

Query: WAVE:INSert: WRAP?

Response: <character_data>

Arguments: YES or NO

YES treat as continuous waveform
NO treat as single shot waveform

WAVE:MARKer:CLOCk:FIRSt

Purpose: Sets the time at which the first rising edge of the waveform begins. In order for this command to have affect WAVE:MARKer:TYPE must be set to CLOCk.

Command: WAVE:MARKer:CLOCk:FIRSt <numeric_value>

Query: WAVE:MARKer:CLOCk:FIRSt?

Response: <numeric_value>

Arguments: <numeric_value> Time 0 to duration of waveform in seconds

Notes: *Because the marker transitions occur on clock edges, the resolution corresponds to the time between clocks.*
At 400 MHz it is 2.5 ns.

WAVE:MARKer:CLOCk:FREQuency

Purpose: Sets the frequency of the marker output. In order for this command to have affect WAVE:MARKer:TYPE must be set to CLOCk.

Command: WAVE:MARKer:CLOCk:FREQuency <numeric_value>

Query: WAVE:MARKer:CLOCk:FREQuency?

Response: <numeric_value>

Arguments: <numeric_value> 10 Hz to 200 MHz

Notes: *Because marker transitions occur on clock edges, and the marker must be high for the same amount of time that it is low (50% duty cycle), the FREQuency rounds to the nearest value corresponding to an even number of clock periods.*

WAVE:MARKer:EDGE:DEFault

Purpose: Replaces the currently defined edges with two edges: going high at 1 X clock interval, and going low at 32 X clock interval.

The rising edge is not placed at time 0 because in a triggered mode, the marker output would be HIGH while the LW400 was awaiting trigger.

Command: WAVE:MARKer:EDGE:DEFault

Query: None

Response: None

Arguments: None

WAVE:MARKer:EDGE:NDEFind?

Purpose: Find out the number of marker edges defined for WAVE:MARKer:TYPe EDGE

Command: None.

Query: WAVE:MARKer:EDGE:NDEFind?

Response: <numeric_value> 0 to 128

Arguments: None

WAVE:MARKer:EDGE:TIME

Purpose: Sets the time where the next edge of a marker may be inserted. In order for this field to have affect WAVE:MARKer:TYPE must be set to EDGE.

Command: WAVE:MARKer:EDGE:TIME <numeric_value>

Query: WAVE:MARKer:EDGE:TIME?

Response: <numeric_value>

Arguments: <numeric_value> time in seconds

WAVE:MARKer:EDGE[:STATe]

Purpose: Set marker state at MARKer:EDGE:TIME.
This defines a new edge. Max #edges: 128.

Command: WAVE:MARKer:EDGE <character_data>

Query: WAVE:MARKer:EDGE?

Response: <character_data>

Arguments: <character_data> LOW or HIGH

WAVE:MARKer:LEVel

Purpose: Sets the voltage levels of the marker. The marker can be either TTL or ECL.

Command: WAVE:MARKer:LEVel <character_data>

Query: WAVE:MARKer:LEVel?

Response: <character_data> TTL or ECL

Arguments: <character_data> TTL or ECL

WAVE:MARKer:TYPE

Purpose: Selects either a clock marker or an edge marker. A clock marker allows a frequency of the clock to be defined and where the first edge is located. The edge marker allows edges to be set at specific times in the waveform.

Command: WAVE:MARKer:TYPE <character_data>

Query: WAVE:MARKer:TYPE?

Response: <character_data> EDGE or CLOCK

Arguments: <character_data> EDGE or CLOCK

WAVE:MATH:COUPling

Purpose: Affects only integration. If set to DC, a flat non-0 level will integrate to a ramp. If set to AC, signal minus the median is integrated.

Command: WAVE:MATH:COUPling <character_data>

Query: WAVE:MATH:COUPling?

Response: <character_data> AC or DC

Arguments: <character_data> AC or DC

WAVE:MATH:IMMediate

Purpose: Performs the math function specified by WAVE:MATH[:OPERation] on the current waveform (defined by WAVE:OPEN) and WAVE:SOURce2 (if applicable) on the region between the left and right time cursors. The result is placed into the current waveform.

Command: WAVE:MATH:IMMediate

Query: None

Response: None

Arguments: None

WAVE:MATH:SMOoth

Purpose: Sets the width, in number of sample points, for the wave math smoothing computation.

Command: WAVE:MATH:SMOoth <character_data>

Query: WAVE:MATH:SMOoth?

Response: <character_data>

Arguments: <character_data>
THREE
FIVE
SEVEN
NINE

WAVE:MATH:SOURce2

Purpose: Selects the "other" waveform for operations requiring two sources (add, sub, mult, div, conv)

Command: WAVE:MATH:SOURce2 <string>

Query: WAVE:MATH:SOURce2? <string>

Response: <string>

Arguments: <string> The name of the other waveform, in quotes.

Notes: *WaveMath operates on the currently selected waveform and SOURce2, if applicable to the selected operation.*

WAVE:MATH[:OPERation]

Purpose: Specifies which math operation will be performed by WAVE:MATH:IMMediate. The available functions are listed in Arguments.

Command: WAVE:MATH <character_data>

Query: WAVE:MATH?

Response: <character_data>

Arguments: <character_data>

ADD	Selects an add function.
CONVolve	Selects a convolve function.
DIFFerentiate	Selects a differentiate function.
DIVide	Selects a divide function.
INTegrate	Selects an integrate.
MULTiply	Selects a multiply function.
SMOoth	Selects a smooth function.
SUBTract	Selects a subtract function.

WAVE:NEW

Purpose: Creates a new waveform with the name specified by the Arguments.

Command: WAVE:NEW <string>

Query: WAVE:NEW?

Response: <string>

Arguments: <string> Up to 15 characters, in quotes.

Example: WAVE:NEW "IN3 TEST4"
The file names may have embedded spaces, &, _, -, and %.
Some file names are reserved. The reserved names are:
"CH1 FUNC GEN"
"CH2 FUNC GEN"
"DEFAULT A"
"DEFAULT B"
"UNROLLED"

WAVE:OPEN

Purpose: Opens a waveform from the current project.

Command: WAVE:OPEN <string>

Query: WAVE:OPEN?

Response: <string>

Arguments: <string> A waveform file name, in quotes.

WAVE:REGion:LEFT

Purpose: Set the position of the left time cursor. This is a synonym for DISP:TRACE:CURSORs:TIME:LEFT. Either may be used at any time. The left cursor is the position where edit operations begin.

Command: WAVE:REGion:LEFT <numeric_value>

Query: WAVE:REGion:LEFT?

Response: <numeric_value>

Arguments: <numeric_value>

WAVE:REGion:RIGHt

Purpose: Set the position of the right time cursor. This command only has effect if DISPlay[:WINDow]:TRACe:CURSors:TIME:TRACk is off. This is a synonym for DISP:TRACE:CURSORs:TIME:RIGHT. The right cursor delimits a region for those operations that affect a region, i.e., CUT, WAVE:AMPLitude, WAVE:TIME.

Command: WAVE:REGion:RIGHt <numeric_value>

Query: WAVE:REGion:RIGHt?

Response: <numeric_value>

Arguments: <numeric_value>

WAVE:SAVE

Purpose: Saves the current waveform with the name supplied by the Arguments. If a name other than the current name of the waveform is given then the current waveform is saved with the new name. The old waveform is left unchanged. If a name (other than the current waveform) is given that already exists, then an error message will be displayed and the waveform will not be saved.

Command: WAVE:SAVE<string>

Query: WAVE:SAVE?

Response: <string> The name of the last waveform saved by WAVE:SAVE

Arguments: <string> File name is quotes, up to 15 characters.
Example: WAVE:SAVE "NEWWAVENAME"
Creates a file named NEWWAVENAME.

WAVE:SELect

Purpose: Selects which waveform editor will be the target of all :WAVE commands.

Command: WAVE:SELect <character_data>

Query: WAVE:SELect?

Response: <character_data>, one of CH1/CH2/SCR

Arguments: <character_data> CH1/CH2/SCR

Example:
:WAVE:SEL CH1; OPEN "MY WAVE"
Opens "MYWAVE" into Channel 1. Channel 1 is displayed.

WAVE:TIME:DELay

Purpose: Changes the time position of the contents of the waveform at and to the right of the left cursor. The argument specifies the new time position for the left cursor. If the delay is decreased, the left cursor and all data offer it move to the left, and some data to the left of the left cursor is overwritten. If the delay is increased then the left cursor moves to the right and the voltage level under the left time cursor is repeated. Features can be delayed with a resolution of a 100 ps at 400 MHz clock decade.

Command: WAVE:TIME:DELay <numeric_value>

Query: WAVE:TIME:DELay?

Response: <numeric_value> - the value last set by WAVE:TIME:DELAY

Arguments: <numeric_value>, seconds

Notes: *This is an "overlapped" command, that is, subsequent commands can execute before this operation completes. Use *WAI or *OPC to synchronize with completion.*

WAVE:TIME:DURation:MODE

Purpose: Selects the mode for changing the duration of a feature. The two modes are insert and overwrite. Insert changes the duration of the region between the left and right time cursors but does not affect the features outside the time cursors. The region to the right of the right time cursor will only change in time (according to the duration change). Overwrite changes the duration of the region between the left and right time cursors but will not change the overall length of the waveform (unless the duration change is greater than the length of the waveform). The area to the right of the right time cursor will be overwritten if the duration is increased or the last point in the region between the left and right time cursors will be replicated if the duration is decreased.

Command: WAVE:TIME:DURation:MODE <character_data>

Query: WAVE:TIME:DURation:MODE?

Response: <character_data>

Arguments: <character_data> INSert/OVERwrite

WAVE:TIME:DURation[:TIME]

Purpose: Changes the duration of the region between the left and right time cursors. The waveform will be changed using the duration change mode defined by WAVE:TIME:DURation:MODE. The duration of a region can be increased in 100 ps steps.

Command: WAVE:TIME:DURation <numeric_value>

Query: WAVE:TIME:DURation?

Response: <numeric_value>

Arguments: <numeric_value> Duration of region (10 ns - memory length)

Notes: *If the duration is decreased high frequency information can be lost. This is because frequency increases as duration decreases. Repeated duration changes work from a saved copy of the original data so this affect can be reversed.

*This is an "overlapped" command. See ...TIME:DELay.

WAVE:TIME:MOVE

Purpose: Moves the feature between the left and right time cursors. The feature is extracted from the waveform (using a baseline that is defined by a line drawn from the voltage point under the left cursor to the voltage point under the right cursor) and then summed back into the waveform at the new time position. The feature can be moved in 100 ps steps. The argument is the new position of the time left cursor.

Command: WAVE:TIME:MOVE <numeric_value>

Query: WAVE:TIME:MOVE?

Response: <numeric_value>

Arguments: <numeric_value>, seconds

Notes: The destination of move must be such that the entire region can be summed back into the waveform. Therefore, the argument should be less than waveform duration minus (time right - time left).

Repeated moves use the original extracted data, so the feature does not degrade with repeated moves.

This is an "overlapped" command. See the note on "WAVE:TIME:DELay.

This page left intentionally blank

7 REMOTE PROGRAMMING EXAMPLES

Introduction

This section of the manual provides programming examples based on a GPIB remote control program, LWGPIB.BAS, written in Microsoft QuickBASIC (ver 4.5) for 80X86 based personal computers. This is a simple GPIB terminal program which includes a menu based user interface. It allows users to send individual remote commands, send queries and receive replies, and transfer waveforms, in DIF format, to and from the LW400 series AWG. As in all GPIB programs, the commands used are heavily dependent on the interface hardware. LWGPIB.BAS was written for a National Instruments PCII/IIA GPIB interface adapter with its associated NI488.2 interface software (ver 2.1.1). This program is intended to serve as an example of principle. Similar GPIB input/output commands are used by other interface hardware suppliers and can be used to provide equivalent functionality.

Setting Up The Environment For The QuickBASIC Programming

GPIB Interface

The QuickBASIC programming environment must include a library of functions and subrountine calls for the GPIB adapter and its supporting software. The National Instruments NI488.2 software for DOS includes a QuickBASIC language interface in the file, QBIB.OBJ. Any QuickBASIC applications program, represented by the name APPLIC, can be link compiled with this program from DOS using the QuickBASIC linker:

LIB QBIB.LIB + QBIB.OBJ; (Produces the stand alone library QBIB.LIB)

BC APPLIC; (Compiles the application producing APPLIC.OBJ)

LINK APPLIC.OBJ, , ,QBIB.LIB; (Linker creates the executable file APPLIC.EXE)

APPLIC (Executes the application).

Alternatively, the QuickBASIC environment can be set up to run programs by setting up a QuickLibrary using the following DOS commands:

LINK /Q QBIB.OBJ, , ,BQLB45.LIB; (Creates a QuickLibrary, QBIB.QLB).

QB APPLIC /L QBIB.QLB (Run QuickBASIC with the application loaded using the QuickLibrary).

The application must include QBDECL.BAS at the beginning of the program. This program, also supplied by National Instruments, contains constants, declarations, and subroutine prototypes required to control the GPIB interface. QBDECL.BAS can be merged with the application program or the metacommand, $INCLUDE, can be used, within the application, to incorporate QBDECL.BAS during compilation.

Additional information on setting up QuickBASIC to work with the National Instruments PCII/IIA GPIB adapters can be found in chapter 3 of the National Instrument's, "NI-488.2 Software Reference Manual For MS-DOS"

The LWGPIB.BAS Program

A complete listing of the LWGPIB.BAS program follows. Key elements, related to LW400 remote control operations, are discussed in detail in the following sections.

```
' --- Main - LWGPIB.BAS
' --- Initialize program and declare program subroutines
DECLARE SUB RecallWave (AWG%)
DECLARE SUB SendCommand (AWG%)
DECLARE SUB SendQuery (AWG%)
DECLARE SUB SetLocal (AWG%)
DECLARE SUB HelpScreen ()
DECLARE SUB StoreWave (AWG%)
DECLARE SUB StoreScreenDump (AWG%)
DECLARE SUB InitScreen ()

' --- Merge QBDECL.BAS functions and subroutines for National Instruments GPIB adapter
REM $INCLUDE: 'QBDECL.BAS'

PRINT "              GPIB REMOTE CONTROL PROGRAM"
' --- Prompt for AWG address verify AWG is present
FOUND = 0
WHILE FOUND = 0
        GOSUB InitDevice
WEND
' --- Initialize screen, display selection menu, prompt for selection and branch to function
CALL InitScreen
CONT = 1
WHILE CONT = 1
COLOR 11
LINE INPUT "                          ENTER OPTION: "; OPT$
        OPTION$ = UCASE$(OPT$)
                SELECT CASE OPTION$
                        CASE "D"
                                CALL StoreWave(AWG%)
                        CASE "U"
                                CALL RecallWave(AWG%)
                        CASE "C"
                                CALL SendCommand(AWG%)
                        CASE "Q"
                                CALL SendQuery(AWG%)
                        CASE "L"
                                CALL SetLocal(AWG%)
                        CASE "A"
                                CALL IBLOC(AWG%)
                                GOSUB InitDevice
                                WHILE FOUND = 0
```

```
                        GOSUB InitDevice
                        WEND
                    CASE "E"
                        CONT = 0
                    CASE "H"
                        CALL HelpScreen
                        CALL InitScreen
                    CASE ELSE
                    COLOR 12
                        PRINT "                    INVALID OPTION"
                END SELECT
                COLOR 14
WEND
CALL IBLOC(AWG%)
SYSTEM

InitDevice: ' --- Subroutine to prompt for GPIB address of AWG and verify that it is present
   COLOR 14, 1, 11
   CLS
   PRINT "                    LCGPIB"
   PRINT " "
   COLOR 10
   LINE INPUT "          ENTER GPIB ADDRESS OF LW4XX AWG: "; ADD$
   DEV$ = "DEV" + ADD$
        AWG% = ILFIND(DEV$)
                IF AWG% < 0 THEN
                COLOR 12
                   PRINT "              COULD NOT FIND AWG AT ADDRESS "; ADD$
                   LINE INPUT "          CHECK ADDRESS SETTING AND CABLE
THEN HIT ANY KEY"; X$
                   FOUND = 0
                ELSE
                 CALL IBTMO(AWG%, 10)
                 CMD$ = "*IDN?"
                 STA% = ILWRT(AWG%, CMD$, 5)
                 RD$ = SPACE$(100)
                 STA% = ILRD(AWG%, RD$, 100)
                    IF (STA% AND &H4000) THEN
                    COLOR 12
                        PRINT "          COULD NOT FIND AWG AT ADDRESS ";
ADD$
                        LINE INPUT "          CHECK ADDRESS SETTING AND CABLE
THEN HIT ANY KEY"; X$
```

```
                    FOUND = 0
                ELSE
                    FOUND = 1
                    TMO% = 12: CALL IBTMO(AWG%, TMO%)
                END IF
                END IF
RETURN

' --- Subroutine to display help screen
SUB HelpScreen
  VIEW PRINT
  CLS
  PRINT " "
  COLOR 15
  PRINT "    EXPLANATION OF AVAILABLE OPTIONS: "
  PRINT " "
  COLOR 14
  PRINT "  A : GPIB Address: Prompts the user for the GPIB address of the AWG."
  PRINT "  C : Send Command: Prompts the user for a remote command then sends"
  PRINT "                    the command to the AWG."
  PRINT "  Q : Send Query:   Prompts the user for a remote query, sends this query,"
  PRINT "                    and displays the response from the AWG."
  PRINT "  L : Local:        Returns AWG to local operation."
  PRINT "  D : Download:     Prompts for a filename and stores current waveform from"
  PRINT "                    AWG to a DIF file on the PC. The default path is the same"
  PRINT "                    drive and directory where this program resides. A full path"
  PRINT "                    can be specified. For example, to store a waveform called"
  PRINT "                    TEST.WAV to a directory named WAVES on the B drive, the"
  PRINT "                    following should be entered when prompted for a filename:"
  PRINT "                    B:\WAVES\TEST.WAV"
  PRINT "  U : Upload:       Prompts the user for a filename restores"
  PRINT "                    the specified DIF waveform file to AWG."
  PRINT "  H : Help:         Displays this screen."
  PRINT "  E : Exit:         Exits program and returns to DOS."
  LINE INPUT "    Hit enter key to continue", help$
  END SUB
```

```
' --- Subroutine to display selection menu
SUB InitScreen
        CLS
        COLOR 12, 1, 4
        CLS
        COLOR 15
        PRINT "                              LWGPIB"
        PRINT "    GPIB REMOTE CONTROL PROGRAM FOR LECROY LW4XX AWG's"
        PRINT "    FOR USE WITH NATIONAL INSTRUMENTS GPIB INTERFACE"
        COLOR 11
        PRINT " "
        PRINT "                    AVAILABLE OPTIONS ARE:"
        PRINT " "
        COLOR 14
        PRINT "                    A  = CHANGE GPIB ADDRESS"
        PRINT "                    C  = SEND REMOTE COMMAND"
        PRINT "                    Q  = SEND REMOTE QUERY"
        PRINT "                    L  = RETURN TO LOCAL OPERATION"
        PRINT "                    D  = DOWNLOAD WAVEFORM TO DIF FILE"
        PRINT "                    U  = UPLOAD WAVEFORM FROM DIF FILE"
        PRINT "                    H  = HELP"
        PRINT "                    E  = EXIT"
        VIEW PRINT 18 TO 24
END SUB

' --- Subroutine to upload waveform from disk for AWG
SUB RecallWave (AWG%)

   COLOR 12
   LINE INPUT "                    ENTER FILENAME: "; FILENAME$
   file$ = UCASE$(FILENAME$)


  COLOR 15

          PRINT "                  "; file$; " IS BEING UPLOADED TO THE AWG "


   CALL ibeot(AWG%, 0) ' NI488.2 subroutine to prevent EOI being asserted.
   CMD$ = "WAVE:DATA" ' LW400 remote command to accept waveform data
   CALL IBWRT(AWG%, CMD$) ' NI488.2 subroutine to write command string (CMD$) to
                                        'device (AWG%)
   CALL ibeot(AWG%, 1) ' NI488.2 subroutine to assert EOI and end of command
```

```
   CALL IBWRTF(AWG%, file$) ' NI488.2 subroutine to write a binary file (file$)to device
                            '(AWG%)

END SUB

' --- Subroutine to send a remote command
SUB SendCommand (AWG%)
   COLOR 10
   LINE INPUT "                    ENTER COMMAND: "; CMD$
   CALL IBWRT(AWG%, CMD$)  ' NI488.2 subroutine to write command string (CMD$)
                           ' to device (AWG%)
   CLS
END SUB

' --- Subroutine to send a remote query and receive and display the reply
SUB SendQuery (AWG%)
   COLOR 10
   LINE INPUT "                     ENTER QUERY: "; CMD$
   CALL IBWRT(AWG%, CMD$) ' NI488.2 subroutine to write command string (CMD$)
                                    ' to device (AWG%)
   COLOR 13
   PRINT "              AWG REPLY:",
   TMO% = 10
   STA% = ILTMO(AWG%, TMO%) ' NI488.2 function sets timeout to TMO% seconds
                                    ' returns the status word ibsta
   REPLY$ = SPACE$(1)
GetReply:
   STA% = ILRD(AWG%, REPLY$, 1) ' NI488.2 function read string REPLY$ from device
                                                                    ' AWG% and
returns the status word, ibsta
   IF REPLY$ = CHR$(10) THEN GOTO GetReply
   COLOR 14
   PRINT REPLY$;
   STA% = ILRSP(AWG%, SPR%) ' NI488.2 function returns contents of device AWG%'s
                                    'serial poll byte

   IF SPR% AND 16 THEN
   GOTO GetReply
   ELSE
   PRINT ""
   END IF

END SUB
```

```
' Subroutine to return the AWG to local operation
SUB SetLocal (AWG%)
   COLOR 10
   PRINT "         LOCAL OPERATION IS ENABLED UNTIL NEW OPTION IS SELECTED"
   CALL IBLOC(AWG%) ' NI488.2 subroutine to unassert the remote enable line
END SUB

' Subroutine to download and store a waveform from the AWG, in DIF format, to disk
SUB StoreWave (AWG%)

   COLOR 12
   LINE INPUT "                    ENTER FILENAME: "; FILENAME$
   file$ = UCASE$(FILENAME$)
   COLOR 15
   PRINT "                  "; "CURRENT WAVEFORM "; "BEING STORED TO "; file$
   CMD$ = "WAVE:DATA?": CALL IBWRT(AWG%, CMD$)

   STA% = ILRDF(AWG%, file$)  ' NI488.2 function to read the current waveform the device
                                        'AWG% into the file, file$

END SUB
```

End Or Identify (EOI) Operation

Except where specifically noted, all commands to and from the LW400 series AWG's are terminated by asserting the EOI signal line simultaneously with the last byte transmitted. No other command terminators are required.

Initializing GPIB Communication With The AWG

The National Instrument GPIB interface must be opened to communicate with a selected device by using the IBFIND interface subroutine or the ILFIND function as shown in the following example from the Initdevice subroutine in the LWGPIB.BAS program:

```
LINE INPUT "          ENTER GPIB ADDRESS OF LW4XX AWG: "; ADD$ ' Enter GPIB addr.
  DEV$ = "DEV" + ADD$
       AWG% = ILFIND(DEV$)  'determine unit descriptor of selected instrument at address ADD$
```

IBFIND and ILFIND return a positive number, called the unit descriptor, used to identify the selected device in all other GPIB transactions. If the call fails, a negative number is returned in place of the unit descriptor and provides an indication of an interface error.

Sending A Command To The LW400 Series AWG

The subroutine SendCommand provides an example of using National Instrument's output command, IBWRT, to send a remote command, in the form of the ASCII string CMD$, to the AWG.

```
' --- Subroutine to send a remote command
SUB SendCommand (AWG%)
   COLOR 10  ' Set trace color to green
   LINE INPUT "                    ENTER COMMAND: "; CMD$
   CALL IBWRT(AWG%, CMD$)  ' NI488.2 subroutine to write command string (CMD$)
                           ' to device (AWG%)
   CLS
END SUB
```

IBWRT is called as a subroutine and requires the unit descriptor (AWG%), to identify the device being addressed, and the command string, CMD$, as arguments. Any of the LW400 remote commands can be sent to the AWG using this subroutine.

Sending a Query, Reading the Response, and Using Status to Determine When the Operation is Done

The query form of a remote command is used to obtain information about the state of the AWG. The query is sent to the AWG and it responds with the desired information. The subroutine SendQuery handles this operation in the LWGPIB.BAS program.

The query command string, CMD$, is entered and output to the AWG using the National Instruments IBWRT subroutine, which was previously described:

```
LINE INPUT "                ENTER QUERY: "; CMD$ ' Enter the desired command
CALL IBWRT(AWG%, CMD$) ' NI488.2 subroutine to write command string (CMD$) to device
                       ' to device (AWG%)
```

The next section of code displays the response header and sets the GPIB interface time out. Depending on the information requested, the AWG response may be delayed. The National Instrument's function ILTMO is used to increase the time out delay to the value set by the variable TMO%, in this case 10 seconds, to allow for the worst case response time response time. Some queries, such as *TST?, can require timeouts in the range of minutes and will not work with this program. Alternative techniques such as using service request interrupts provide more flexible response.

```
COLOR 13      ' Set trace color to violet
  PRINT "             AWG REPLY:", ' Print response header
  TMO% = 10
  STA% = ILTMO(AWG%, TMO%) ' NI488.2 function sets timeout to TMO% seconds
                           ' returns the status word ibsta to the variable STA%
```

The response is read and displayed one character at a time using the National Instrument's GPIB read function, ILRD. This process continues until the AWG's output buffer is empty. This is determined by using the serial poll function, ILRSP, to read the status byte. The message available

(MAV) bit, bit 4, is tested to determine if the query response is complete.

```
  REPLY$ = SPACE$(1) ' Dimension the response string, REPLY$, as 1 character long GetReply:
  STA% = ILRD(AWG%, REPLY$, 1) ' NI488.2 function read string REPLY$ from device
' AWG% and returns the status word, ibsta
  IF REPLY$ = CHR$(10) THEN GOTO GetReply ' loop to GetReply if response is a line feed
  COLOR 14 ' Set trace color to yellow
  PRINT REPLY$; 'Build a response string by concatenating single characters until the query output buffer
                ' is empty
  STA% = ILRSP(AWG%, SPR%) ' NI488.2 function returns contents of device AWG%'s
                                        'serial poll byte
  IF SPR% AND 16 THEN  GOTO GetReply ' If message available (MAV) bit is set in status byte get
                                                  'additional characters
  ELSE ' If no additional characters are available print a blank line and exit
  PRINT ""
  END IF
END SUB
```

Downloading A Waveform From The AWG To A File

The selected waveform in the AWG can be output in data interchange format (DIF) via GPIB by sending the LW400 the WAVE:DATA? query. The subroutine StoreWave is used to handle this operation. It prompts the user to enter a file name, file$, under which the waveform data will be stored. It then issues the WAVE:DATA? query using the National Instrument IBWRT subroutine. The waveform is read directly into the desired file using the National Instruments read file function, ILRDF.

```
' Subroutine to download and store a waveform from the AWG, in DIF format, to disk
SUB StoreWave (AWG%)
   COLOR 12  ' Set the trace color to red
   LINE INPUT "                    ENTER FILENAME: "; FILENAME$
   file$ = UCASE$(FILENAME$)  ' Convert filename to uppercase
   COLOR 15  ' Set the trace color to white
   PRINT "                    "; "CURRENT WAVEFORM "; "BEING STORED TO "; file$
   CMD$ = "WAVE:DATA?": CALL IBWRT(AWG%, CMD$) ' Output WAVE:DATA? Query to AWG

   STA% = ILRDF(AWG%, file$) ' NI488.2 function to read the current waveform the device
                              'AWG% into the file, file$

END SUB
```

Uploading A Waveform A DIF File To The AWG

Waveform files, in data interchange format (DIF), are accepted by the AWG after it receives the WAVE:DATA remote command. The subroutine RecallWave sends a selected waveform file to the AWG. As in the case of the StoreWave subroutine, the user is prompted to enter the desired filename. Prior to sending the command WAVE:DATA the National Instrument's subroutine IBEOT is used to disable EOI. This suppresses command termination at the end of the WAVE:DATA command. The AWG waits for the waveform file which is sent with the following write file subroutine (IBWRTF). After the transfer is complete EOI is again enabled.

```
' --- Subroutine to upload waveform from disk for AWG
SUB RecallWave (AWG%)

   COLOR 12 ' Set the trace color to red
   LINE INPUT "                    ENTER FILENAME: "; FILENAME$ ' Enter waveform filename
   file$ = UCASE$(FILENAME$) ' Convert filename to uppercase for display


 COLOR 15  'Set the trace color to white

          PRINT "                    "; file$; " IS BEING UPLOADED TO THE AWG "
```

```
CALL IBEOT(AWG%, 0) ' NI488.2 subroutine to prevent EOI being asserted until transfer is complete.
  CMD$ = "WAVE:DATA" ' LW400 remote command to accept waveform data
  CALL IBWRT(AWG%, CMD$) ' NI488.2 subroutine to write command string (CMD$) to
                'device (AWG%)
  CALL IBEOT(AWG%, 1)  ' NI488.2 subroutine to enable EOI at the end of following commands.
  CALL IBWRTF(AWG%, file$) ' NI488.2 subroutine to write a binary file (file$)to device
                '(AWG%)

END SUB
```

THIS PAGE LEFT INTENTIONALLY BLANK

Index

*

C

D

H

I

M

O

P

S

T

W