AWG5000 and AWG7000 Series
Arbitrary Waveform Generators

Programmer Manual

Tektronix

**AWG5000 and AWG7000 Series
Arbitrary Waveform Generators**

Programmer Manual

Tektronix

AWG5000 and AWG7000 Series Programmer Online Help, Supports AWG software V4.2, part number 076-0146-05. August 22, 2011

Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:
- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Warranty

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product. Parts, modules and replacement products used by Tektronix for warranty work may be new or reconditioned to like new performance. All replaced parts, modules and products become the property of Tektronix.

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

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

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

[W2 – 15AUG04]

Table of Contents

Getting Started

Syntax and Commands

Status and Events

Appendices

Getting Started

Introduction

This online programmer guide provides you with the information to use commands for remotely controlling your instrument. With this information, write computer programs that will perform functions such as setting the front-panel controls, selecting clock source, setting sampling rate, and exporting data for use in other programs. In addition to the traditional GPIB electronic interface, (referred to as the physical GPIB interface), your instrument is provided with a *TekVISA* GPIB-compatible interface, (referred to as the virtual GPIB interface).

Refer to Documentation for information on related manuals and documents.

The programmer guide is divided into the following major topics (books):

- Getting Started: This topic introduces you to the online help and provides basic information about setting up your instrument for remote control.

- Command Syntax: This topic provides an overview of the command syntax that you will use to communicate with the instrument and other general information about commands, such as how commands and queries are constructed, how to enter commands, constructed mnemonics, and argument types.

- Command Groups: This topic contains all the commands listed in functional groups. Each group consists of an overview of the commands in that group and a table that lists all the commands and queries for that group. Click a command in the listing to display a detailed description of the command.

- Status and Events: This topic discusses the status and event reporting system for the GPIB interface. This system informs you of certain significant events that occur within the instrument. Topics that are discussed include registers, queues, event handling sequences, synchronization methods, and messages that the instrument may return, including error messages.

- Appendices: This topic contains miscellaneous information, such as a table of the factory initialization (default) settings, and GPIB interface specifications that may be helpful when using remote commands to control the instrument.

Remote Control

The instrument support GPIB interface and LAN interface. To set the GPIB address, use the **System Menu > GPIB/LAN Configuration** menu.
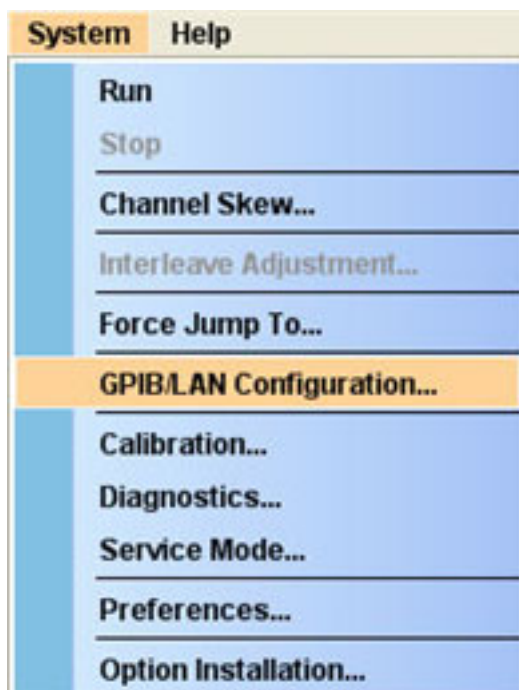
GPIB Interface   The GPIB enables up to 15 devices (including the controller) to be connected for concurrent use. With the arbitrary waveform generator connected to an external computer via GPIB, use the computer to remotely control your instrument. With the instrument, use the GPIB interface as a controller. See the GPIB Parameters for information on GPIB parameters.

LAN Interface   The instrument accept two types of Ethernet LAN connections; one is simple ("Raw Socket") connection, and the other is VXI-11 protocol. See the LAN Parameters for information on LAN parameters.

GPIB Parameters

To use the GPIB, the instrument require you to configure the GPIB mode and the GPIB address.

- Talk/Listen: Select this mode to remotely control your instrument using an external computer as the controller.

- Off Bus: Select this mode to electronically disconnect the instrument from the GPIB bus.

- Address: This address is a number that allows the software to identify each device connected to the GPIB bus. Specify a unique number from 0 to 30 for each device.
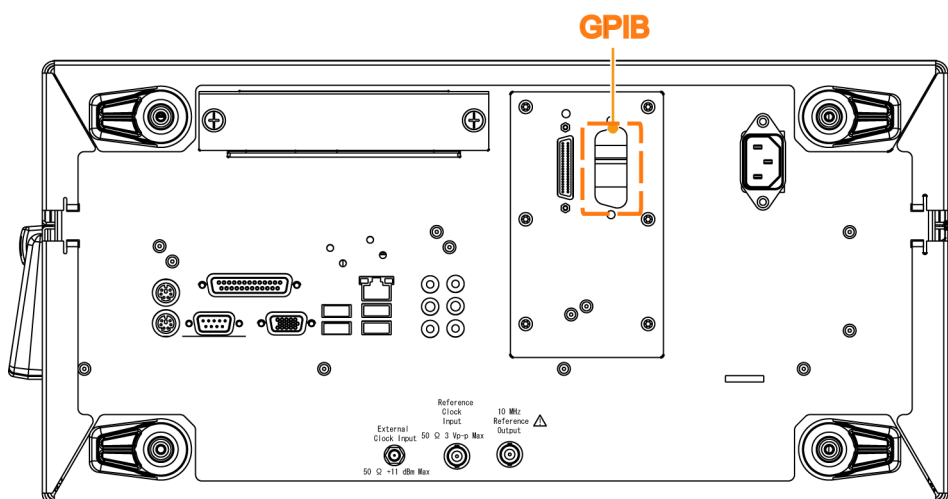
LAN Parameters

In the instrument, set parameters to start or stop a process that communicates through LAN. The instrument can communicate with LAN using the following methods:

- VXI-11 Server (LAN): VXI-11 protocol is used through TekVISA. To use this protocol, TekVISA must also be installed on the remote controller (PC).

- Raw Socket (LAN): TCP/IP protocol is used. Use the GPIB/LAN Configuration option to set the socket communication On and Off. Specify the port number for the Raw Socket interface. This port number must be assigned to the application software or the Ethernet driver on the external controller.

By default, the instrument are specified to automatically acquire an IP address by DHCP. Refer to Windows documentation regarding network-related parameters. For TekVISA, refer to the TekVISA manual.
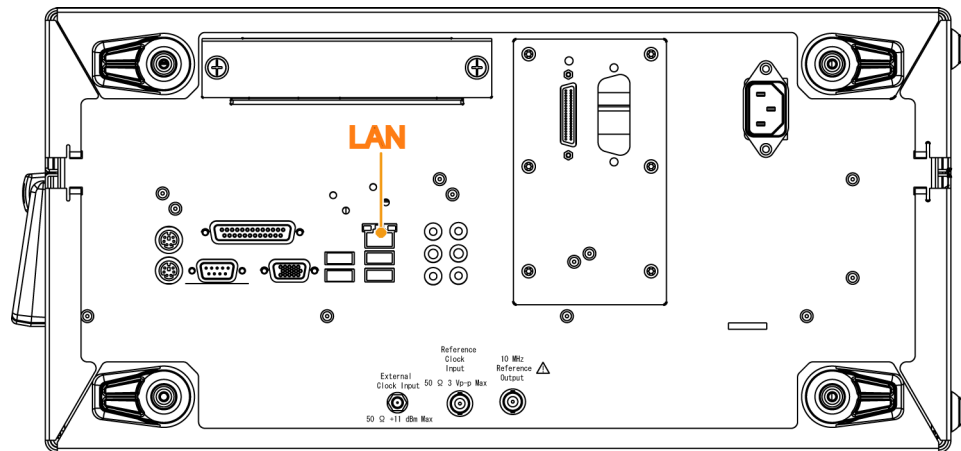
Connecting to the Instrument using GPIB

Your instrument has a 24-pin GPIB connector on its rear panel. This connector has a D-type shell and conforms to IEEE Std 488.1–1987. Attach an IEEE Std 488.1–1987 GPIB cable to this connector and to your controller as shown in the following figure.
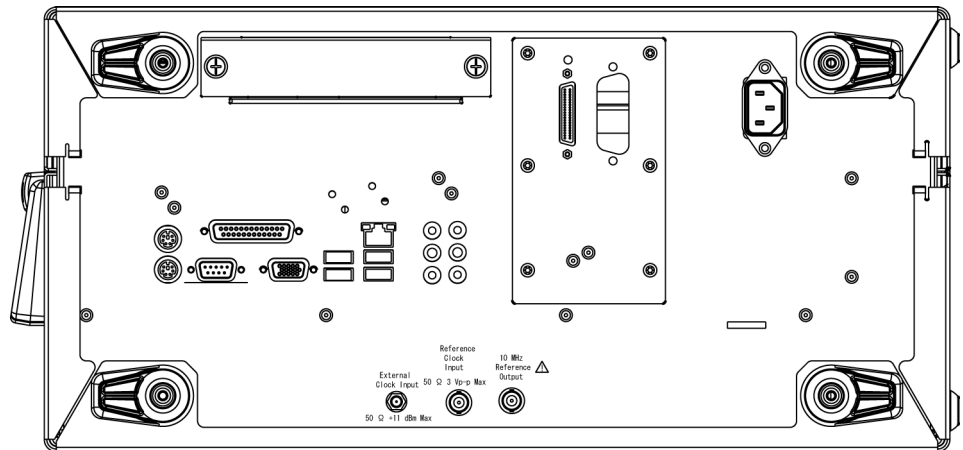
Setting Up GPIB Communication

Before setting up your instrument for remote communications using the electronic (physical) GPIB interface, you should familiarize yourself with the following GPIB requirements:

- A unique device address must be assigned to each device on the bus. No two devices can share the same device address.

- No more than 15 devices can be connected to any one line.

- One device should be connected for every 6 feet (2 meters) of cable used. No more than 65 feet (20 meters) of cable should be used to connect devices to a bus.

- At least two-thirds of the devices on the network should be powered on while using the network.

- Connect the devices on the network in a star or linear configuration. Do not use loop or parallel configurations.

Setting the GPIB Address   To function correctly, your instrument must have a unique device address. The default settings for the GPIB configuration are:

- GPIB Address: 1

- GPIB Mode: Talk/Listen

To change the GPIB address settings, do the following:

1. Select GPIB/LAN Configuration… from the System menu.
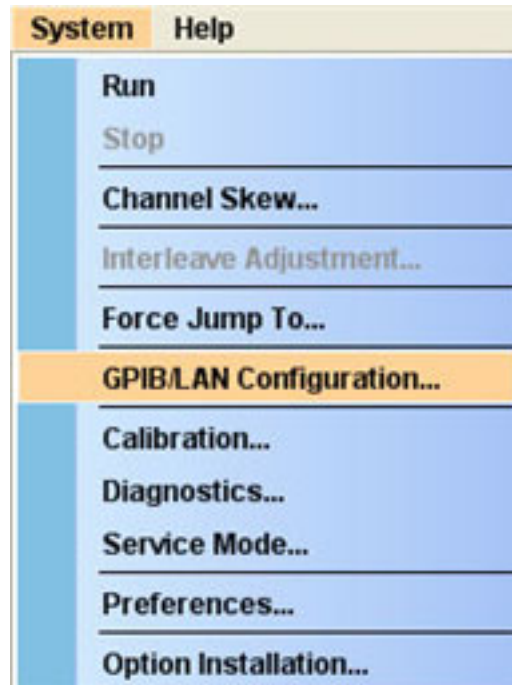
2. The GPIB/LAN Configuration dialog box is displayed.
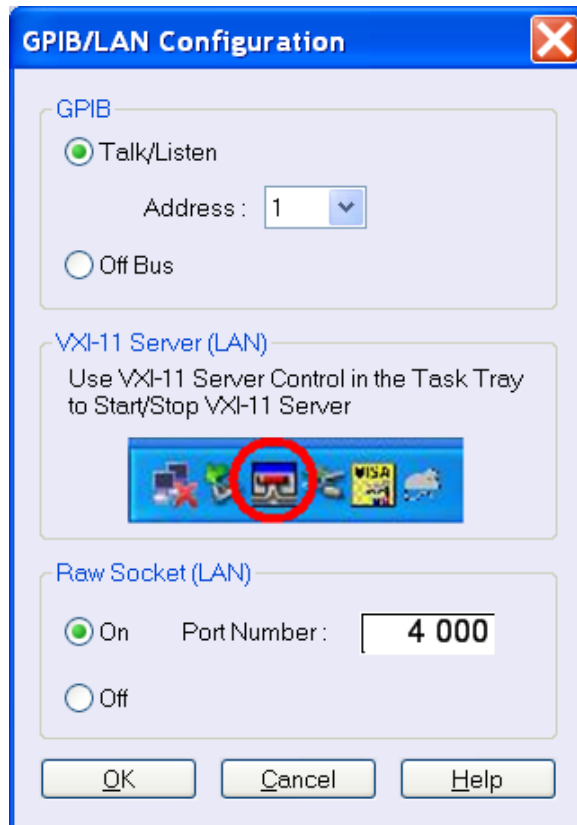
3. Change the GPIB Address to a unique address.

4. Click **OK** button.

Documentation

Review the following table to locate more information about this product.

To read about	Use these documents
Installation and Operation (overviews)	Read the Quick Start User Manual for general information about how to use your instrument.
In-depth Operation and User Interface Help	Access the user online help from the Help menu for information on virtually all controls and elements on screen. Online help includes detailed instructions for using instrument functions.
Programmer Commands	Access the programmer online guide from the Help menu. The programmer guide includes the syntax of remote commands.
Specifications and Performance Verification Procedures	Read the Technical Reference documents for specifications and the performance verification procedures. These documents are available on the Documentation CD.
Service Procedures	Read the Service Manuals to service the instrument to the module level. The manuals are available on the Tektronix Web site (www.Tektronix.com/manuals).

Sample Program

The sample program illustrates methods use to control the arbitrary waveform generator. This program sends waveform data and then starts waveform generation. Access the sample program from Windows Start menu. Select **All Programs > Tektronix > AWG > Examples**.

This program is also included on the Document CD.

Syntax and Commands

Command Syntax

Syntax Overview

Control the operations and functions of the instrument through the GPIB and LAN interface using commands and queries. The related topics listed below describe the syntax of these commands and queries. The topics also describe the conventions that the instrument uses to process them. See the Command Groups topic for a listing of the commands by command group or use the index to locate a specific command.

Refer to the following table for the symbols that are used.

Table 2-1: Syntax symbols and their meanings

Symbol	Meaning
< >	Defined element
::=	Is defined as
|	Exclusive OR
{ }	Group; one element is required
[ ]	Optional; can be omitted
...	Previous elements can be repeated
( )	Comment

Command and Query Structure

Overview Commands consist of set commands and query commands (usually called commands and queries). Commands modify instrument settings or tell the instrument to perform a specific action. Queries cause the instrument to return data and status information.

Most commands have both a set form and a query form. The query form of the command differs from the set form by its question mark on the end. For example, the set command `AWGControl:RRATE` has a query form `AWGControl:RRATE?`. Not all commands have both a set and a query form. Some commands have only set and some have only query.

Messages A command message is a command or query name followed by any information the instrument needs to execute the command or query. Command messages may contain five element types, defined in the following table.

Table 2-2: Message symbols and their meanings

Symbol	Meaning
<Header>	This is the basic command name. If the header ends with a question mark, the command is a query. The header may begin with a colon (:) character. If the command is concatenated with other commands, the beginning colon is required. Never use the beginning colon with command headers beginning with a star (*).
<Mnemonic>	This is a header subfunction. Some command headers have only one mnemonic. If a command header has multiple mnemonics, a colon (:) character always separates them from each other.
<Argument>	This is a quantity, quality, restriction, or limit associated with the header. Some commands have no arguments while others have multiple arguments. A <space> separates arguments from the header. A <comma> separates arguments from each other.
<Comma>	A single comma is used between arguments of multiple-argument commands. Optionally, there may be white space characters before and after the comma.
<Space>	A white space character is used between a command header and the related argument. Optionally, a white space may consist of multiple white space characters.

Commands   Commands cause the instrument to perform a specific function or change one of the settings. Commands have the structure:

`[:]<Header>[<Space><Argument>[<Comma><Argument>]...]`

A command header consists of one or more mnemonics arranged in a hierarchical or tree structure. The first mnemonic is the base or root of the tree and each subsequent mnemonic is a level or branch off the previous one. Commands at a higher level in the tree may affect those at a lower level. The leading colon (:) always returns you to the base of the command tree.

Queries   Queries cause the instrument to return status or setting information. Queries have the structure:

`[:]<Header>?`

`[:]<Header>?[<Space><Argument>[<Comma><Argument>]...]`

Clearing the Instrument

Use the Device Clear (DCL) or Selected Device Clear (SDC) GPIB functions to clear the Output Queue and reset the instrument to accept a new command or query. Refer to your GPIB library documentation for further details about the Device Clear operation.

Command Entry

Rules   The following rules apply when entering commands:

- You can enter commands in upper or lower case.

- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 decimal).

- The instrument ignores commands consisting of any combination of white space characters and line feeds.

Abbreviating   You can abbreviate many instrument commands. Each command in this documentation shows the abbreviations in capitals. For example, enter the command `MMEMory:CATalog` simply as `MMEM:CAT`.

Concatenating
Use a semicolon (;) to concatenate any combination of set commands and queries. The instrument executes concatenated commands in the order received. When concatenating commands and queries, follow these rules:

1. Separate completely different headers by a semicolon and by the beginning colon on all commands except the first one. For example, the commands `TRIGger:IMPedance 50` and `AWGControl:RMODe TRIGgered`, can be concatenated into the following single command:

   `TRIGger:IMPedance 50;::AWGControl:RMODE TRIGgered`

2. If concatenated commands have headers that differ by only the last mnemonic, abbreviate the second command and eliminate the beginning colon. For example, concatenate the commands `TRIGger:SOURCE EXTernal` and `TRIGger:POLarity NEGative` into a single command:

   `SOURce EXTernal, NEGative`

   The longer version works equally well:

   `TRIGger:SOURCE EXTernal;::TRIGger:POLarity NEG`

3. Never precede a star (*) command with a semicolon (;) or colon (:).

4. When you concatenate queries, the responses to all the queries are concatenated into a single response message. For example, if the high level of the marker1 of channel one is 1.0 V and the low level of that is 0.0 V, the concatenated query `SOURce1:MARKer:VOLTage:HIGH?`; `SOURce1:MARKer:VOLTage:LOW?` will return the following:

   1.0;0.0

5. Set commands and queries may be concatenated in the same message. For example, `AWGControl:RMODe SEQuence;SEQuence:LENGth?` is a valid message that sets the run mode to Sequence. The message then queries the length of the sequence. Concatenated commands and queries are executed in the order received.

Here are some invalid concatenations:

- `TRIGger:SOURce INTernal;AWGControl:RMODe TRIGgered` (no colon before AWGControl)

- `TRIGger:SOURce INTernal;::TRIGger:POLarity NEG` (extra colon before TRIGger:SOURce INTernal;POLarity NEG instead)

Terminating
This documentation uses <EOM> (end of message) to represent a message terminator.

Table 2-3: Message terminator and meaning

Symbol	Meaning
<EOM>	Message terminator

For messages sent to the instrument, the end-of-message terminator must be the END message (EOI asserted concurrently with the last data byte). The instrument always terminates messages with LF and EOI. It allows white space before the terminator. For example, it allows CR LF.

Parameter Types

Parameters are indicated by angle brackets, such as <file_name>. There are several different types of parameters, as listed in the following table. The parameter type is listed after the parameter. Some parameter types are defined specifically for the instrument command set and some are defined by SCPI.

Table 2-4: Parameter types, their descriptions, and examples

Parameter type	Description	Example
Arbitrary block	A block of data bytes	#512234xxxxx... where 5 indicates that the following 5 digits (12234) specify the length of the data in bytes; xxxxx... indicates actual data or #0xxxxx...<LF><&EOI>
Boolean	Boolean numbers or values	ON or ≠ 0 OFF or 0
Discrete	A list of specific values	MINimum, MAXimum
NR1 numeric	Integers	0, 1, 15, –1
NR2 numeric	Decimal numbers	1.2, 3.141,–6.5
NR3 numeric	Floating point numbers	3.1415E+9
NRf numeric	Flexible decimal numbers that may be type NR1, NR2, or NR3	See NR1, NR2, and NR3 examples in this table
String	Alphanumeric characters (must be within quotation marks)	"Testing 1, 2, 3"

About MIN, MAX You can also use MINimum and MAXimum keywords in the commands with the "Numeric" parameter. Set the minimum value or the maximum value using these keywords and query these values.

Block   Several instrument commands use a block argument form (see the following table).

Table 2-5: Block symbols and their meanings

Symbol	Meaning
<NZDig>	A nonzero digit character in the range of 1–9
<Dig>	<Dig> A digit character, in the range of 0–9
<DChar>	A character with the hexadecimal equivalent of 00 through FF (0 through 255 decimal) that represents actual data
<Block>	A block of data bytes defined as: <Block> ::={#<NZDig><Dig>[<Dig>...][<DChar>...] |#0[<DChar>...]<terminator>}

Arbitrary Block   An arbitrary block argument is defined as:

#<NZDig><Dig>[<Dig>...][<DChar>...]

or

#0[<DChar>...]<terminator>

<NZDig> specifies the number of <Dig> elements that follow. Taken together, the <NZDig> and <Dig> elements form a decimal integer that specifies how many <DChar> elements follow.

#0 means that the <Block> is an indefinite length block. The <terminator> ends the block.

NOTE. The arbitrary waveform generators do not support the indefinite format (a block starts with #0).

Quoted String   Some commands accept or return data in the form of a quoted string, which is simply a group of ASCII characters enclosed by a single quote (') or double quote ("). For example: "this is a quoted string". This documentation represents these arguments as follows:

Table 2-6: String symbol and meaning

Symbol	Meaning
<QString >	Quoted string of ASCII text

A quoted string can include any character defined in the 7-bit ASCII character set. Follow these rules when you use quoted strings:

1. Use the same type of quote character to open and close the string. For example: "this is a valid string".

2. You can mix quotation marks within a string as long as you follow the previous rule. For example, "this is an 'acceptable' string".

3. You can include a quote character within a string simply by repeating the quote.

   For example: "here is a "" mark".

4. Strings can have upper or lower case characters.

5. If you use a GPIB network, you cannot terminate a quoted string with the END message before the closing delimiter.

6. A carriage return or line feed embedded in a quoted string does not terminate the string, but is treated as just another character in the string.

7. The maximum length of a quoted string returned from a query is 1000 characters.

Here are some invalid strings:

■ "Invalid string argument' (quotes are not of the same type)

■ "test<EOI>" (termination character is embedded in the string)

Units and SI Prefix   If the decimal numeric argument refers to voltage, frequency, impedance, or time, express it using SI units instead of using the scaled explicit point input value format <NR3>. (SI units are units that conform to the System International d'Unites standard.) For example, use the input format 200 mV or 1.0 MHz instead of 200.0E-3 or 1.0E+6, respectively, to specify voltage or frequency.

Omit the unit when you describe commands, but include the SI unit prefix. Enter both uppercase and lowercase characters. The following list shows examples of units you can use with the commands.

■ V for voltage (V).

■ HZ for frequency (Hz).

■ OHM for impedance (ohm).

■ S for time (s).

■ DBM for power ratio.

■ PCT for %.

■ VPP for Peak-to-Peak Voltage (V p-p).

■ UIPP for Peak-to-Peak, Unit is UI (UI p-p).

■ UIRMS for RMS, Unit is UI (UIrms).

■ SPP for Peak-to-Peak, Unit is second (s p-p).

■ SRMS for RMS, Unit is second (srms).

■ V/NS for SLEW's unit (V/ns).

In the case of angles, use RADian and DEGree. The default unit is RADian. The SI prefixes, which must be included, are shown in the following table. You can enter both uppercase and lowercase characters.

Table 2-7: SI prefixes and their indexes

SI prefix [1]	Corresponding power
EX	10^{18}
PE	10^{15}
T	10^{12}
G	10^{9}
MA	10^{6}
K	10^{3}
M	10^{-3}
U [2]	10^{-6}
N	10^{-9}
P	10^{-12}
F	10^{-15}
A	10^{-18}

[1] Note that the prefix m/M indicates 10^{-3} when the decimal numeric argument denotes voltage or time, but indicates 10^{6} when it denotes frequency.

[2] Note that the prefix u/U is used instead of "μ".

Since M (m) can be interpreted as 1E-3 or 1E6 depending on the units, use mV for V, and MHz for Hz.

The SI prefixes need units.

correct: 10MHz, 10E+6Hz, 10E+6

incorrect: 10M

SCPI Commands and Queries

The arbitrary waveform generator uses a command language based on the SCPI standard. The SCPI (Standard Commands for Programmable Instruments) standard was created by a consortium to provide guidelines for remote programming of instruments. These guidelines provide a consistent programming environment for instrument control and data transfer. This environment uses defined programming messages, instrument responses and data formats that operate across all SCPI instruments, regardless of manufacturer.

The SCPI language is based on a hierarchical or tree structure that represents a subsystem (see following figure). The top level of the tree is the root node; it is followed by one or more lower-level nodes.
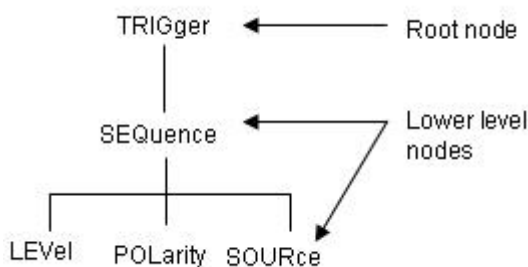
You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

Command Groups

Control group commands

Use the following commands to control operating modes:

Table 2-8: Control group commands and their descriptions

Command	Description
AWGControl:APPLication:RUN	Executes the specified application
AWGControl:APPLication:STATe?	Returns the running state of the specified application
AWGControl:CLOCk:DRATe	Sets or returns the divider rate for the external oscillator
AWGControl:CLOCk:PHASe[:ADJust]	Sets or returns the clock phase adjust
AWGControl:CLOCk:SOURce	Sets or returns the clock source
AWGControl:COMPile	Executes the commands in the specified equation file
AWGControl:CONFigure:CNUMber?	Returns the number of channels available on the instrument
AWGControl:DC[n][:STATe]	Sets or returns the DC state
AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet	sets or returns the DC output level
AWGControl:DOUTput[n][:STATe]	Outputs the raw waveform in the DAC of the specified channel
AWGControl:ENHanced:SEQuence:JMODe	Sets or returns the jump mode
AWGControl:EVENt:DJUMp:DEFine	Associates an event pattern with the jump target for Dynamic Jump
AWGControl:EVENt:JMODe	Sets or returns the event jump mode
AWGControl:EVENt:SOFTware[:IMMediate]	Executes the sequencer jump to the specified element index
AWGControl:EVENt:TABLe[:IMMediate]	Generates an event forcibly in the table jump mode
AWGControl:INTerleave:ADJustment:AMPLitude	Sets or returns the interleave adjustment amplitude
AWGControl:INTerleave:ADJustment:PHASe	Sets or returns the interleave adjustment phase
AWGControl:INTerleave[:STATe]	Enables or disables the interleave state for channels
AWGControl:INTerleave:ZERoing	Sets or removes the zeroing option for the interleave mode
AWGControl:RMODe	Sets or returns the run mode of the arbitrary waveform generator

Table 2-8: Control group commands and their descriptions (cont.)

Command	Description
AWGControl:RRATe	Sets or returns the repetition rate of the arbitrary waveform generator
AWGControl:RRATe:HOLD	Sets or returns the hold property of repetition rate
AWGControl:RSTate?	Returns the state of the arbitrary waveform generator or sequencer
AWGControl:RUN[:IMMediate]	Initiates the output of a waveform or a sequence
AWGControl:SEQuencer:POSition?	Returns the current position of the sequencer
AWGControl:SEQuencer:TYPE?	Returns the type of the arbitrary waveform generator's sequencer
AWGControl:SNAMe?	Returns the current setup file name of the arbitrary waveform generator
AWGControl:SREStore	Restores the arbitrary waveform generator's setting from a specified settings file
AWGControl:SSAVe	Saves the arbitrary waveform generator's setting to a specified settings file
AWGControl:STOP[:IMMediate]	Stops the output of a waveform or a sequence

Calibration Group Commands

Use the following calibration commands to calibrate the arbitrary waveform generator:

Table 2-9: Calibration group commands and their descriptions

Command	Description
*CAL?	Performs an internal calibration of the arbitrary waveform generator and returns the status
CALibration[:ALL]	Performs a full calibration of the arbitrary waveform generator

Diagnostic Group Commands

Use the following diagnostic commands to control self-test diagnostic routines:

Table 2-10: Diagnostic group commands and their descriptions

Command	Description
DIAGnostic:DATA?	Returns the result of a self test
DIAGnostic[:IMMediate]	Executes selected self test routines
DIAGnostic:SELect	Selects the self-test routines
*TST?	Executes a self test

Display Group Commands

Use the following display commands to set the display state of waveform and sequence windows on the instrument:

Table 2-11: Display group commands and their descriptions

Command	Description
DISPlay[:WINDow[1|2]][:STATe]	Minimizes or restores the sequence or waveform window of the arbitrary waveform generator

Event Group Commands

Use the following event commands to configure external event input and generate an event:

Table 2-12: Event group commands and their descriptions

Command	Description
EVENt[:IMMediate]	Generates a forced event
EVENt:IMPedance	Sets or returns the impedance of the external event input
EVENt:JTIMing	Sets or returns the jump timing
EVENt:LEVel	Sets or returns the event level
EVENt:POLarity	Sets or returns the polarity of event signal

Instrument Group Commands

Use the following instrument commands to set or return the coupled state of instrument models:

Table 2-13: Instrument group commands and their descriptions

Command	Description
INSTrument:COUPle:SOURce	Sets or returns the coupled state for a channel

Mass Memory Group Commands

Use the following mass memory commands to read/write data from/to hard disk on the instrument:

Table 2-14: Mass Memory group commands and their descriptions

Command	Description
MMEMory:CATalog?	Returns the current contents and state of the mass storage media
MMEMory:CDIRectory	Sets or returns the current directory of the file system on the arbitrary waveform generator
MMEMory:DATA	Sets or returns block data to/from the file in the current mass storage device
MMEMory:DELete	Deletes a file or directory from the instrument's hard disk
MMEMory:IMPort	Imports a file into arbitrary waveform generator's setup as a waveform
MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe]	Sets or queries FREQuency parameter that decides whether frequency is modified during waveform import
MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel	Sets or queries the channel of which the amplitude and offset values are selected to be updated during import
MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]	Sets or queries LEVel parameter that decides whether amplitude and offsets are modified during waveform import
MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE	Sets or queries the data to be imported. It also sets or queries which data's amplitude and offset values are selected for update during RSA file import.
MMEMory:IMPort:PARameter:NORMalize	Sets or queries whether waveform data are to be normalized
MMEMory:IMPort:PARameter:RESampling:FREQuency	Sets or queries the sampling rate parameter for resampling
MMEMory:IMPort:PARameter:RESampling[:STATe]	Sets or queries the resampling state for waveform import
MMEMory:MDIRectory	Creates a new directory in the current path on the mass storage system
MMEMory:MSIS	Selects a mass storage device used by all MMEMory commands

Output Group Commands

Use the following output commands to set or return the characteristics of the output port of the arbitrary waveform generator:

Table 2-15: Output group commands and their descriptions

Command	Description
OUTPut[n]:FILTer[:LPASs]:FREQuency	Sets or returns the low pass filter frequency of the filter
OUTPut[n][:STATe]	Sets or returns the output state of the arbitrary waveform generator

Sequence Group Commands

Use the following sequence commands to define and edit a sequence:

Table 2-16: Sequence group commands and their descriptions

Command	Description
SEQuence:ELEMent[n]:GOTO:INDex	Sets or retrieves the target index for the GOTO command of the sequencer
SEQuence:ELEMent[n]:GOTO:STATe	Sets or retrieves the GOTO state of the sequencer
SEQuence:ELEMent[n]:JTARget:INDex	Sets or retrieves the target index for the sequencer's event jump operation
SEQuence:ELEMent[n]:JTARget:TYPE	Sets or queries the target type for the jump
SEQuence:ELEMent[n]:LOOP:COUNt	Sets or queries the loop count
SEQuence:ELEMent[n]:LOOP:INFinite	Sets or returns the infinite looping state for a sequence element
SEQuence:ELEMent[n]:TWAit	Sets or returns the wait trigger state for an element on or off
SEQuence:ELEMent[n]:WAVeform[m]	Sets or returns the waveform for a sequence element
SEQuence:JUMP[:IMMediate]	Executes the sequencer jump to the specified element index
SEQuence:LENGth	Sets or returns the sequence length

Sequence Commands
The following set of commands provides ways to create and edit the waveform sequences in the instruments. When the instrument runs a sequence, it outputs the waveforms in the order defined in the sequence.

To run a sequence, the instrument must be first put in the Sequence mode. This can be done by using either the instrument interface or the AWGControl:RMODe SEQuence command. Once the instrument is in the Sequence mode, it uses either the hardware or the software sequencer to execute the sequence. Query the current sequencer type using the AWGControl:SEQuencer:TYPE? command. However, it is not possible to select the sequencer type.

There is only one sequence defined for an instrument. This is common to all channels. Refer to the AWG7000 and AWG5000 Series Arbitrary Waveform Generators Quick Start User Manuals for a discussion on sequencing waveforms.

Creating and Working with Sequences

To create a sequence programmatically, first set the sequence length using `SEQuence:LENGth(?)` command. This creates a sequence of specified length. At this stage all elements of the sequence will have their parameters set to default values. The default values are as follows:

Table 2-17: Sequence element parameters and their default values

Sequence element parameter name	Default value	Remote command to query or set the parameter
CH 1 Waveform	""	SEQuence:ELEMent[n]:WAVeform[m]
CH 2 Waveform	""	SEQuence:ELEMent[n]:WAVeform[m]
Trigger Wait State	0	SEQuence:ELEMent[n]:TWAit
Infinite loop flag	0	SEQuence:ELEMent[n]:LOOP: INFinite
Loop count	1	SEQuence:ELEMent[n]:LOOP: COUNt
Event Jump Type	OFF	SEQuence:ELEMent[n]:JTARget: TYPE
Event Jump target index	1	SEQuence:ELEMent[n]:JTARget: INDex
Go To target Index	1	SEQuence:ELEMent[n]:GOTO:INDex

To learn how to use the commands to create a sequence, refer to the individual command descriptions.

Source Group Commands

Use the following source commands to set and query the waveform or marker output parameter:

Table 2-18: Source group commands and their descriptions

Command	Description
[SOURce[1]]:FREQuency[:CW|:FIXed]	Sets or returns the sampling frequency of the arbitrary waveform generator
[SOURce[1]]:ROSCillator:FREQuency	Selects the reference oscillator frequency
[SOURce[1]]:ROSCillator:MULTiplier	Sets or returns the reference oscillator multiplier rate
[SOURce[1]]:ROSCillator:SOURce	Selects the reference oscillator source
[SOURce[1]]:ROSCillator:TYPE	Selects the type of the reference oscillator

Table 2-18: Source group commands and their descriptions (cont.)

Command	Description
[SOURce[n]]:COMBine:FEED	Adds the signal from an external input to the output of the channel
[SOURce[n]]:DAC:RESolution	Sets or returns the DAC resolution
[SOURce[n]]:DELay[:ADJust]	Sets or returns the delay (in seconds) of the analog output
[SOURce[n]]:DELay:POINts	Sets or returns the delay (in points) of the analog output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude]	Sets or returns the amplitude of digital output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH	Sets or returns the high digital output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW	Sets or returns the low digital output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet	Sets or returns the offset of digital output
[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]	Sets or returns the amplitude of digital output
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH	Sets or returns the high digital output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW	Sets or returns the low digital output
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet	Sets or returns the offset of digital output
[SOURce[n]]:FUNCtion:USER	Sets or returns the waveform to waveform memory
[SOURce[n]]:MARKer[1|2]:DELay	Sets or returns the marker delay
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude]	Sets the marker amplitude
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH	Sets the marker high level
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW	Sets the marker low level
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet	Sets the marker offset
[SOURce[n]]:PDELay:HOLD	Sets or returns which parameter is retained when sampling rate or waveform length is changed
[SOURce[n]]:PHASe[:ADJust]	Sets or returns the phase of the analog output
[SOURce[n]]:SKEW	Sets or returns the skew for the waveform associated with a channel

Table 2-18: Source group commands and their descriptions (cont.)

Command	Description
[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]	Sets or returns the amplitude for the waveform associated with a channel
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH	Sets or returns the high voltage level for the waveform associated with a channel
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW	Sets or returns the low voltage level for the waveform associated with a channel
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet	Sets or returns the offset for the waveform associated with a channel
[SOURce[n]]:WAVeform	Sets or returns the output waveform from the current waveform list for each channel when Run Mode is not Sequence

Status Group Command

The external controller uses the status commands to coordinate operation between the arbitrary waveform generator and other devices on the bus. The status commands set and query the registers/queues of the arbitrary waveform generator event/status reporting system. For more information about registers and queues, see Status and Event reporting section.

Table 2-19: Status group commands and their descriptions

Command	Description
*CLS	Clears all event registers and queues
*ESE	Sets or queries the status of Event Status Enable Register (ESER)
*ESR?	Returns the status of Standard Event Status Register (SESR)
*SRE	Sets or queries the bits in Service Request Enable Register (SRER)
*STB?	Returns the contents of Status Byte Register (SBR)
STATus:OPERation:CONDition?	Returns the contents of the Operation Condition Register (OCR)
STATus:OPERation:ENABle	Sets or returns the mask for the Operation Enable Register (OENR)
STATus:OPERation[:EVENt]?	Returns the contents of Operation Event Register (OEVR)
STATus:PRESet	Sets the OENR and QENR registers
STATus:QUEStionable:CONDition?	Returns the status of the Questionable Condition Register (QCR)

Table 2-19: Status group commands and their descriptions (cont.)

Command	Description
STATus:QUEStionable:ENABle	Sets or returns the mask for Questionable Enable Register (QENR)
STATus:QUEStionable[:EVENt]?	Returns the status of the Questionable Event (QEVR) Register and clears it

Subsequence Group Commands

Use the following subsequence commands to define and edit a subsequence:

Table 2-20: Subsequence group commands and their descriptions

Command	Description
SEQuence:ELEMent[n]:SUBSequence	Sets or returns the subsequence for a sequence element
SLISt:SUBSequence:DELete	Deletes the subsequence from the currently loaded setup
SLISt:SUBSequence:NEW	Creates a new subsequence
SLISt:SUBSequence:LENGth	Sets or returns the size of the subsequence
SLISt:SUBSequence:TSTamp?	Returns the time stamp of the subsequence
SLISt:NAME?	Returns the name of the subsequence corresponding to the specified index in the subsequence list
SLISt:SIZE?	Returns the size of the subsequence list
SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt	Sets or returns the loop count for the specified subsequence element
SLISt:SUBSequence:ELEMent[n]:WAVeform[n]	Sets or returns the waveform for an element of the subsequence

Synchronization Group Commands

The external controller uses the synchronization commands to prevent external communication from interfering with arbitrary waveform generator operation.

Table 2-21: Synchronization group commands and their descriptions

Command	Description
*OPC	Ensures the completion of the first command before the second command is issued
*WAI	Prevents the arbitrary waveform generator from executing further commands until all pending commands are executed

System Group Commands

Use the following system commands to control miscellaneous instrument functions:

Table 2-22: System group commands and their descriptions

Command	Description
*IDN?	Returns identification information for the arbitrary waveform generator
*OPT?	Returns the implemented options for the arbitrary waveform generator
*RST	Resets the arbitrary waveform generator to its default state
SYSTem:DATE	Sets or returns the system date
SYSTem:ERRor[:NEXT]?	Retrieves and returns data from the error and event queues
SYSTem:KLOCk	Locks or unlocks the keyboard and front panel of the arbitrary waveform generator
SYSTem:TIME	Sets or returns the system time
SYSTem:VERSion?	Returns the SCPI version number to which the command conforms

Trigger Group Commands

Use the following trigger commands synchronize the arbitrary waveform generator actions with events:

Table 2-23: Trigger group commands and their descriptions

Command	Description
*TRG	Generates a trigger event ABORt Stops waveform generation when the AWG is in gated mode
ABORt	Stops waveform generation when the AWG is in gated mode
TRIGger[:SEQuence][:IMMediate]	Generates a trigger event
TRIGger[:SEQuence]:IMPedance	Sets or returns the trigger impedance
TRIGger[:SEQuence]:LEVel	Sets or returns the trigger input level (threshold)
TRIGger[:SEQuence]:MODE	Sets or returns the trigger timing
TRIGger[:SEQuence]:POLarity	Sets or returns the trigger input polarity
TRIGger[:SEQuence]:SLOPe	Sets or returns the trigger slope
TRIGger[:SEQuence]:SOURce	Sets or returns the trigger source

Table 2-23: Trigger group commands and their descriptions (cont.)

Command	Description
TRIGger[:SEQuence]:TIMer	Sets or returns the internal trigger rate (trigger interval)
TRIGger[:SEQuence]:WVALue	Sets or returns the output data position of a waveform while the instrument is in the waiting-for-trigger state

Waveform Group Commands

Use the following waveform commands to create and transfer waveforms between the instrument and the external controller:

Table 2-24: Waveform group commands and their descriptions

Command	Description
WLISt:NAME?	Returns the waveform name of an element in the waveform list
WLISt:SIZE?	Returns the size of the waveform list
WLISt:WAVeform:DATA	Transfers waveform data from external controller into the waveform list or from the waveform list to the external control program
WLISt:WAVeform:DELete	Deletes the waveform from the currently loaded setup
WLISt:WAVeform:LENGth?	Returns the size of the waveform
WLISt:WAVeform:MARKer:DATA	Sets or queries the waveform marker data
WLISt:WAVeform:NEW	Creates a new empty waveform in the waveform list of current setup
WLISt:WAVeform:NORMalize	Normalizes a waveform that exists in the waveform list of the current setup.
WLISt:WAVeform:PREDefined?	True or false based on whether the waveform is predefined
WLISt:WAVeform:RESAmple	Resamples a waveform that exists in the waveform list of the current setup.
WLISt:WAVeform:TSTamp?	Returns the time stamp of the waveform
WLISt:WAVeform:TYPE?	Returns the type of the waveform

Waveform Data Format   The instrument support two types of waveform data – Integer format and Floating Point format.

Integer format is useful when you want to transfer data faster. It also speeds up restoring data from AWG setup file (`.AWG` file) thereby making loading faster. Loading data into hardware memory is also faster in the integer format because the integer format is the same as the hardware data format and no conversion is necessary.

Floating point format is helpful while editing the waveform because it gives more resolution for editing operations.

The integer data format is shown in the following table. It occupies two bytes per waveform data point. In the figure, "D" refers a data bit and "M" refers to a marker bit. Note that in the 10-bit DAC resolution, marker bits are ignored. However, the bit settings of the marker are not altered and are restored when you switch back to the 8-bit mode.

Table 2-25: Integer data format

	Byte offset 1								Byte offset 0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
8-bit DAC	M2	M1	D7	D6	D5	D4	D3	D2	D1	D0						
10-bit DAC			D9	D8	D7	D6	D5	D4	D3	D2	D1	D0				
14-bit DAC	M2	M1	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Floating data format is the same as the IEEE 754 single precision format. It occupies 4 bytes per waveform data point. It stores normalized data without any scaling. When the waveform in real data format is output, the data is rounded off to the nearest integer value and clipped to fit the DAC range.

The waveforms in the real format retains normalized values. The format for the waveform analog data in the real format is IEEE754 single precision.

The real data format is shown in the following table.

Table 2-26: Real data format

Byte offset 3	Byte offset 2	Byte offset 1	Byte offset 0
IEEE754 single precision format (32 bits)			

Byte offset 4							
7	6	5	4	3	2	1	0
M2	M1						

DAC resolution affects the way hardware interprets the bits in the waveform. Therefore it is necessary to reload waveforms once the DAC resolution is modified. To understand how to change the DAC resolution, see the [SOURce[n]]:DAC:RESolution command. To understand how to load a waveform into hardware memory see the [SOURce[n]]:WAVeform command.

Byte Order During Transfer

Waveform data is always transferred in LSB first format.

Transferring Waveforms in Chunks

When transferring large waveforms, it is convenient to send waveform data in chunks. This allows better memory management and enables you to stop the transfer before it is completed. It also helps the external controller to report the progress of the operation to the user.

The WLISt:WAVeform:DATA command accepts parameters that makes it possible for control programs to send data in any chunk size. The Size parameter of this command sets the chunk size. The StartIndex parameter sets the first data point of each chunk. Note that using StartIndex and Size, it is also possible to transfer only a part of the waveform. Though it is possible to transfer any arbitrary-sized waveform data to an AWG5000 or AWG7000 Series instrument (up to an allowed upper limit), there are certain conditions to load the waveform to hardware waveform memory or sequence memory. See the [SOURce[n]]:WAVeform and SEQuence:ELEMent[n]:WAVeform[m] commands to understand the waveform sizes that are allowed in each case.

Command Descriptions

ABORt (No Query Form)

This command stops waveform generation when the arbitrary waveform generator is in gated mode. This is equivalent to releasing the Trig button on the front panel when the instrument is in gated mode.

Group   Trigger

Syntax   `ABORt`

Related Commands   TRIGger[:SEQuence][:IMMediate], *TRG

Examples   `ABORT` resets the trigger system.

AWGControl:APPLication:RUN (No Query Form)

This command executes the specified application.

Group   Control

Syntax   `AWGControl:APPLication:RUN <application_name>`

Related Commands   AWGControl:APPLication:STATe?

Arguments   `<application_name>`::=<string> specifies the application to be executed.

Examples   `AWGCONTROL:APPLICATION:RUN "SERIALXPRESS"` runs the SerialXpress application.

AWGControl:APPLication:STATe? (Query Only)

This query returns the running state of the specified application.

Group   Control

Syntax   `AWGControl:APPLication:STATe?  <application_name>`

Related Commands   AWGControl:APPLication:RUN

Arguments   `<application_name>`::=`<string>`

Returns   <Boolean>

0 indicates False

1 indicates True

Examples   `AWGCONTROL:APPLICATION:STATE?` "SERIALXPRESS" might return 1 indicating that the SerialXpress application is running.

AWGControl:CLOCk:DRATe

This command and query sets or returns the divider rate for the external oscillator. Divider rate is applicable only when the reference oscillator source is external. Only 1, 2, 4, 8… are valid values.

Errors for the AWG5000 series are –222 and –224. The –222, which is out of range, is produced when a value is greater than 32 and less than or equal to 256. Any non-power of 2 value creates a –224 error. For the AWG7000 series, there is no out of range error and any non-power of 2 and greater than 256 produces a –224.

Group   Control

Syntax   `AWGControl:CLOCk:DRATe <divider_rate>`
`AWGControl:CLOCk:DRATe?`

Related Commands   AWGControl:CLOCk:SOURce

Arguments   `<divider_rate>`::=`<NR1>`

At *RST, this returns the minimum value.

Returns   <NR1>

Examples AWGCONTROL:CLOCK:DRATE 8 sets the divider rate to 8.

AWGCONTROL:CLOCK:DRATE? returns 8.

AWGControl:CLOCk:PHASe[:ADJust] (AWG7000B and AWG7000C Series only)

This command and query sets or returns the clock phase adjust. It is used to adjust the internal clock phase of the instrument to synchronize or align timing with external devices.

When the sampling rate is below 375 MS/s, the instrument may take a few minutes to execute the command or to set the sampling rate. Spurious in the output signal may increase if you set the clock phase to any value other than 0 (zero) with the interleave in On state.

Group Control

Syntax AWGControl:CLOCk:PHASe[:ADJust] <NR3>
AWGControl:CLOCk:PHASe[:ADJust]?

Arguments <NR3>

The setting range of NR3 is ±72,000 degrees (±200 clocks), and the resolution is 0.1 degree.

At *RST, this returns 0 degree.

Returns <NR3>

Examples AWGCONTROL:CLOCK:PHASE:ADJUST 120 sets the clock phase adjustment value to 120 degrees.

AWGCONTROL:CLOCK:PHASE:ADJUST? returns 1.20000000E+002, indicating that the clock phase adjustment value is 120 degrees.

AWGControl:CLOCk:SOURce

This command and query sets or returns the clock source. When the clock source is internal, the arbitrary waveform generator's internal clock is used to generate the clock signal. If the clock source is external, the clock signal from an external oscillator is used.

Group Control

Syntax   AWGControl:CLOCk:SOURce <source>
AWGControl:CLOCk:SOURce?

Related Commands   AWGControl:APPLication:STATe?

Arguments   <source>::={EXTernal|INTernal}

EXTernal specifies that the clock signal from external oscillator is used.

INTernal specifies that the clock signal is generated internally.

At *RST, this value is set to INTernal.

Returns   EXT|INT

Examples   AWGCONTROL:CLOCK:SOURCE EXTERNAL sets the clock source to EXTernal.

AWGCONTROL:CLOCK:SOURCE? returns EXT.

AWGControl:COMPile (No Query Form)

This command executes the commands in the specified equation file. This command executes the WPL contents of the specified file name, returning when the commands in the file have been executed.

Group   Control

Syntax   AWGControl:COMPile <filename>

Arguments   <filename>::=<string>

Examples   AWGCONTROL:COMPILE "SIN.EQU" compiles the equation file named "sin.equ".

AWGControl:CONFigure:CNUMber? (Query Only)

This query returns the number of channels available on the instrument. It returns the count of channels even when they are disabled. However, interleaved channels are not included in the count.

Group   Control

Syntax AWGControl:CONFigure:CNUMber?

Related Commands None

Returns <NR1>

Returns 1, 2, or 4 depending on the model.

Examples AWGCONTROL:CONFIGURE:CNUMBER? might return 2.

AWGControl:DC[n][:STATe]

This commands and query sets or returns the output state of one of the four DC outputs. Use this command to turn off or turn on the DC outputs.

The value of n = 1|2|3|4

The output state is common for all DC outputs. Therefore, irrespective of the value used for 'n' in the command, all DC outputs are switched on or switched off at once.

Group Control

Syntax AWGControl:DC[n][:STATe] <state>
AWGControl:DC[n][:STATe]?

Related Commands AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet

Arguments <state>::= <Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

Returns <state>

Examples AWGCONTROL:DC1:STATE 1 sets the DC1 output to On.

AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet

This command and query sets or returns the DC output level.

The value of n = 1|2|3|4.

Group   Control

Syntax   AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet <offset>
AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet?

Related Commands   AWGControl:DC[n][:STATe]

Arguments   <offset>::=<NR3> the value will be between –3.0 V to +5.0 V.

At *RST, this returns 0 V.

Returns   <NR3>

Examples   AWGCONTROL:DC1:VOLTAGE:OFFSET 1.0V sets the DC1 level to 1.0 V.

AWGCONTROL:DC1:VOLTAGE:OFFSET? might return 1.00000000E+000.

AWGControl:DOUTput[n][:STATe]

This command enables the raw DAC waveform outputs for the specified channel.

The query form of this command returns the status of raw DAC waveform output for the specified channel. When the state is ON, offset and filter settings for the channel are ignored.

This command is not supported on the instruments with Option 02 or Option 06.

Group   Control

Syntax   AWGControl:DOUTput[n][:STATe] <state>
AWGControl:DOUTput[n][:STATe]?

Related Commands   [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet, OUTPut[n]:FILTer[:LPASs]:FREQuency

Arguments `<state>`::= <Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

Returns <state>

Examples `AWGCONTROL:DOUTPUT1:STATE 1` causes the instrument to output raw DAC waveform from Channel 1.

AWGControl:ENHanced:SEQuence:JMODe

This command and query sets or returns the jump mode.

This command is available for the AWG5012B, AWG5000C, and AWG7000C with option 09. This command is provided for compatibility with the AWG400/500/600/700 series instruments.

The query form will return TABL when the instrument is in the Table Jump mode, otherwise LOG will be returned.

Group Control

Syntax `AWGControl:ENHanced:SEQuence:JMODe <jump_mode>`
`AWGControl:ENHanced:SEQuence:JMODe?`

Arguments `<jump_mode>`::={LOGic|TABLe|SOFTware}

`LOGic or SOFTware` activates Event Jump. The jump target defined in the sequence definition will be the target of Event Jump.

`TABLe` activates Table Jump. The Table Jump definition is used as the jump target.

At *RST, this returns LOGic.

Returns <jump_mode>

Examples `AWGCONTROL:ENHANCED:SEQUENCE:JMODE TABLe` sets the jump mode to Table.

`AWGCONTROL:ENHANCED:SEQUENCE:JMODE?` might return TABL if the instrument is in the Table Jump mode.

AWGControl:EVENt:DJUMp:DEFine (No Query Form)

This command associates an event pattern with the jump target for Dynamic Jump. The query returns the jump target associated to the specified <event_pattern>.

Group   Control

Syntax   AWGControl:EVENt:DJUMp:DEFine <event_pattern>,<jump_target>
AWGControl:EVENt:DJUMp:DEFine? <event_pattern>

Arguments   event_pattern::=<NR1>. The values ranges between 0 and 511. This parameter specifies the event pattern to make an event jump. The event bits are mapped to the integer value as follows:

MSB   LSB

Event bits ——— ——— ——-8 76543210

jump_target::=<NR1>. The values are –1 and 1 to maximum sequence length. This parameter specifies the sequence index as the jump target. If –1 is specified, the event jump for the specified <event_pattern> is cancelled.

At *RST, all definitions are cancelled.

Examples   AWGCONTROL:EVENT:DJUMP:DEFINE 15,3 sets the jump target index to third sequence element for the event pattern 00001111.

AWGCONTROL:EVENT:DJUMP:DEFINE? 15 might return 3.

AWGControl:EVENt:JMODe

This command and query sets or returns the event jump mode.

Group   Control

Syntax   AWGControl:EVENt:JMODe <jump_mode>
AWGControl:EVENt:JMODe?

Arguments   jump_mode::={EJUMp | DJUMp}

EJUMp sets the Jump Mode to Event Jump. The jump targets defined in the sequence definition table will be used as the jump target. In this mode, the instrument behavior for the event jump is the same as that of the AWG7000/AWG5000 series.

DJUMp sets the Jump Mode to Dynamic Jump. The Dynamic Jump target definitions are used as the jump target. This is also known as Table Jump.

The jump command is always available in both the modes.

At *RST, this returns EJUMp.

Examples   AWGCONTROL:EVENT:JMODE DJUMP sets the Jump Mode to Dynamic Jump.

AWGControl:EVENt:SOFTware[:IMMediate] (No Query Form)

This command executes the sequencer jump to the specified element index.

Group   Control

Syntax   AWGControl:EVENt:SOFTware[:IMMediate] <target>

Related Commands   SEQuence:JUMP[:IMMediate]

Arguments   <target>::=<NR1>

Examples   AWGCONTROL:EVENT:SOFTWARE:IMMEDIATE 10 forces the sequencer to jump to index number 10.

AWGControl:EVENt:TABLe[:IMMediate] (No Query Form)

This command forcibly generates an event in the Table Jump mode.

This command is available for the AWG5012B, AWG5000C, and AWG7000C with option 09. If the instrument (with option 09) is not in the Table Jump mode, this command will generate a setting conflict error. To set the Jump mode, use the AWGControl:ENHanced:SEQuence:JMODe command.

This command is provided for compatibility with the AWG400/500/600/700 series instruments. The same functionality can be invoked by the EVENt[:IMMediate] command.

NOTE. The Table Jump definition cannot be created or edited using a remote command or the screen interface. The Table Jump definition is derived from the `*.SEQ` *file. Refer to the User Online Help for more information on the Table Jump definition and SEQ file format.*

Group   Control

Syntax   AWGControl:EVENt:TABLe[:IMMediate]

Related Commands   AWGControl:ENHanced:SEQuence:JMODe, EVENt[:IMMediate]

Examples   AWGCONTROL:EVENT:TABLE[:IMMEDIATE] generates an event signal for Table Jump.

AWGControl:INTerleave:ADJustment:AMPLitude

This command and query sets or returns the interleave adjustment amplitude.

This command is available only for Option 06. This setting is only valid when the interleave state is On.

Group   Control

Syntax   AWGControl:INTerleave:ADJustment:AMPLitude <NR3>
AWGControl:INTerleave:ADJustment:AMPLitude?

Related Commands   AWGControl:INTerleave[:STATe], AWGControl:INTerleave:ZERoing

Arguments   <NR3>

Range is between 0.5 Vpp to 1.0 Vpp when Zeroing is OFF.

0.25 Vpp to 0.5 Vpp when Zeroing is ON.

At *RST, this returns 0 Vpp.

Returns   <NR3>

Examples   AWGCONTROL:INTERLEAVE:ADJUSTMENT:AMPLITUDE 1 sets the interleave adjustment amplitude to 1 volts.

AWGCONTROL:INTERLEAVE:ADJUSTMENT:AMPLITUDE? returns 0.00000000E+000, indicating that the interleave adjustment amplitude is 0 volts.

AWGControl:INTerleave:ADJustment:PHASe

This command and query sets or returns the interleave adjustment phase.

This command is available only for Option 06. This setting is only valid when the interleave state is On.

Group   Control

Syntax   AWGControl:INTerleave:ADJustment:PHASe <NR3>
AWGControl:INTerleave:ADJustment:PHASe?

Related Commands   AWGControl:INTerleave[:STATe]

Arguments   <NR3>

At *RST, this returns 0 degree.

Returns   <NR3>

Examples

AWGCONTROL:INTERLEAVE:ADJUSTMENT:PHASE 120 sets the interleave adjustment phase to 120 degrees.

AWGCONTROL:INTERLEAVE:ADJUSTMENT:PHASE? returns 1.20000000E+002, indicating that the interleave adjustment phase is 120 degrees.

AWGControl:INTerleave[:STATe]

This command and query sets or returns the interleave state for channels. This is available only on the AWG7000 series with option 06 instruments.

When Interleave is ON, the output of CH1 and CH2 are mixed at the output circuit to achieve twice the sampling rate. When interleave state is switched on, then:

- ◼ Sampling rate is set to the nearest valid value

- ◼ Waveform remains as before

- ◼ Sequence pointing to CH2 waveform becomes "Empty"

- ◼ Channel count becomes 1

- ◼ Coupled channels lose the coupled state

NOTE. Switching the interleave state from ON to OFF will not restore the sequence CH2 waveforms. Also once the coupled state is lost, it is not restored.

Marker data cannot be interleaved.

Only even marker data is output when the interleave state is ON.

Group   Control

Syntax   AWGControl:INTerleave[:STATe] <state>
AWGControl:INTerleave[:STATe]?

Related Commands   AWGControl:INTerleave:ZERoing

Arguments   <state>::=<Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

Returns   <state>

Examples   AWGCONTROL:INTERLEAVE:STATE 1 sets the instrument to interleave mode.

AWGControl:INTerleave:ZERoing

This command and query sets or returns the state for zeroing when in the interleave mode.

NOTE. This command is available only on the AWG7000 series with option 06 instruments.

Setting Zeroing to ON will change the amplitude setting range when interleaving is done. When Zeroing is OFF, amplitude is not affected by the interleave state.

Setting the zeroing state to ON is a trade-off between bandwidth and signal quality.

Group   Control

Syntax AWGControl:INTerleave:ZERoing <state>
AWGControl:INTerleave:ZERoing?

Related Commands AWGControl:INTerleave[:STATe]

Arguments <state>::=<Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

Returns <state>

Examples AWGCONTROL:INTERLEAVE:ZEROING 1 turns on the zeroing function.

AWGControl:RMODe

This command and query sets or returns the run mode of the arbitrary waveform generator.

Group Control

Syntax AWGControl:RMODe
{CONTinuous|TRIGgered|GATed|SEQuence|ENHanced}
AWGControl:RMODe?

Related Commands AWGControl:RUN[:IMMediate], AWGControl:STOP[:IMMediate], *TRG, [SOURce[n]]:FUNCtion:USER

Arguments CONTinuous sets Run Mode to Continuous.

TRIGgered sets Run Mode to Triggered.

GATed sets Run Mode to Gated.

SEQuence sets Run Mode to Sequence.

ENHanced is provided only for the compatibility with AWG400/500/600/700 series. In the response, SEQ is returned even if ENH is specified in the command.

At *RST, this value is CONTinuous.

Returns CONT|TRIG|GAT|SEQ

Examples `AWGCONTROL:RMODE TRIGGERED` sets the instrument Run mode to Triggered.

`AWGCONTROL:RMODE?` returns CONT if the instrument is in continuous mode.

The following table lists the run modes and their descriptions:

Argument	Description
CONTinuous	Selects the continuous mode, which continuously outputs the waveform. The external trigger, including the FORCE TRIGGER button and the corresponding remote commands, has no effect.
TRIGgered	Sets the triggered mode, which outputs one waveform cycle for each trigger.
GATed	Selects the gated mode, which continuously outputs the waveform or sequence as long as the trigger remains enabled. The trigger remains effective as long as any of the following events occur: ■ The FORCE TRIGGER button remains pressed ■ A valid external gate signal remains input ■ The `TRIGger[:SEQuence][:IMMediate]` or `*TRIG` command has been executed but an ABORt command has not yet been issued
SEQuence	Selects the sequence mode, which outputs the waveform according to the sequence file specified with the `SOURce:FUNCtion:USER` command. If the sequence file is not loaded, this mode is the same as the triggered mode.

AWGControl:RRATe

This command and query sets or returns the repetition rate of the arbitrary waveform generator.

Group Control

Syntax `AWGControl:RRATe <repetition_rate>`
`AWGControl:RRATe?`

Related Commands [SOURce[1]]:FREQuency[:CW|:FIXed]

Arguments   `<repetition_rate>`::=`<NR3>`

At *RST, this value is 10 MHz.

Returns   `<NR3>`

Examples   `AWGCONTROL:RRATE 1000000` sets the repetition rate to 1MHz.

`AWGCONTROL:RRATE?` returns 1E+6.

AWGControl:RRATe:HOLD

This command and query sets or returns the hold property of repetition rate. Setting this to ON keeps the repetition rate of the instrument constant even when the waveform size changes. This causes the sampling rate to change. When this is OFF, the repetition rate changes when the waveform length changes.

Group   Control

Syntax   `AWGControl:RRATe:HOLD <hold_state>`
`AWGControl:RRATe:HOLD?`

Related Commands   AWGControl:RRATe

Arguments   `<hold_state>`::=`<Boolean>`

`0` indicates OFF

`1` indicates ON

At *RST, this returns 0.

Returns   `<NR1>`

Examples   `AWGCONTROL:RRATE:HOLD 1` sets the instrument repetition rate to Hold.

AWGControl:RSTate? (Query Only)

This query returns the run state of the arbitrary waveform generator or the sequencer.

Group	Control
Syntax	`AWGControl:RSTate?`
Related Commands	AWGControl:RMODe, AWGControl:RUN[:IMMediate]
Returns	\<NR1\>
	0 indicates that the instrument has stopped.
	1 indicates that the instrument is waiting for trigger.
	2 indicates that the instrument is running.
Examples	`AWGCONTROL:RSTATE?` might return 0 if the instrument waveform generation is stopped.

AWGControl:RUN[:IMMediate] (No Query Form)

This command initiates the output of a waveform or a sequence. This is equivalent to pressing Run/Stop button on the front panel. The instrument can be put in the run state only when output waveforms are assigned to channels.

Group	Control
Syntax	`AWGControl:RUN[:IMMediate]`
Related Commands	AWGControl:STOP[:IMMediate], [SOURce[n]]:WAVeform
Examples	`AWGCONTROL:RUN` puts the instrument in the run state.

AWGControl:SEQuencer:POSition? (Query Only)

This query returns the current position of the sequencer.

Group	Control
Syntax	`AWGControl:SEQuencer:POSition?`

Related Commands AWGControl:SEQuencer:TYPE?

Returns <NR1>

At *RST, this value is 1.

Examples `AWGCONTROL:SEQUENCER:POSITION?` might return 100.

AWGControl:SEQuencer:TYPE? (Query Only)

This query returns the type of the arbitrary waveform generator's sequencer. The sequence is executed by the hardware sequencer whenever possible.

Group Control

Syntax `AWGControl:SEQuencer:TYPE?`

Related Commands None

Returns HARDware indicates that the instrument is in the hardware sequencer mode.

SOFTware indicates that the instrument is in the software sequencer mode.

At *RST, this value is HARDware.

Examples `AWGCONTROL:SEQUENCER:TYPE?` might return HARD if the instrument is in the hardware sequencer mode.

AWGControl:SNAMe? (Query Only)

This query returns the current setup file name of the arbitrary waveform generator. The response contains the full path for the file including the disk drive.

Group Control

Syntax `AWGControl:SNAMe?`

Related Commands AWGControl:SSAVe, AWGControl:SREStore

Returns   <file_name>,<msus>

<file_name>::=<string>

<msus> (mass storage unit specifier)::=<string>

At *RST, this values is "","C:"

Examples   `AWGCONTROL:SNAME?` might return the following response:

`"\my\project\awg\setup\a1.awg","D:"`

AWGControl:SREStore (No Query Form)

This command restores the arbitrary waveform generator's setting from a specified settings file. The drive may be a local or a network drive. If the full path is not specified, the file will be stored in the current path.

Group   Control

Syntax   `AWGControl:SREStore <file_name>[,<msus>]`

Related Commands   AWGControl:SNAMe?, AWGControl:SSAVe

Arguments   `<file_name>`::=<string>

`<msus>` (mass storage unit specifier)::=<string>

Examples   `AWGCONTROL:SRESTORE "Setup1.Awg"`

AWGControl:SSAVe (No Query Form)

This command saves the arbitrary waveform generator's setting to a specified settings file. The drive may be a local or a network drive. If full path is not specified, the file will be stored in the current path.

Group   Control

Syntax   `AWGControl:SSAVe <file_name>[,<msus>]`

Related Commands   AWGControl:SREStore, AWGControl:SNAMe?

Arguments `<file_name>`::=<string>

`<msus>` (mass storage unit specifier)::=<string>

Examples `AWGCONTROL:SSAVE` "`\my\project\awg\setup\x.awg`","`D:`" will save the current setup to "`D:\my\project\awg\setup\x.awg`".

AWGControl:STOP[:IMMediate] (No Query Form)

This command stops the output of a waveform or a sequence.

Group Control

Syntax `AWGControl:STOP[:IMMediate]`

Related Commands AWGControl:RUN[:IMMediate]

Examples `AWGCONTROL:STOP:IMMEDIATE` stops the output of a waveform.

*CAL? (Query Only)

This query does an internal calibration of the arbitrary waveform generator and returns a status that indicates whether the calibration was completed successfully.

Group Calibration

Syntax `*CAL?`

Related Commands CALibration[:ALL]

Returns <NR1>

0 indicates no error.

Examples `*CAL?` performs an internal calibration and returns results. For example, it might return 0, which indicates that the calibration completed without any errors.

CALibration[:ALL]

This command does a full calibration of the arbitrary waveform generator. In its query form, the command does a full calibration and returns a status indicating the success or failure of the operation. `CALibration[:ALL]?` is equivalent to `*CAL?`

Group   Calibration

Syntax   `CALibration[:ALL]`
`CALibration[:ALL]?`

Related Commands   *CAL?

Returns   <calibration error code> ::= <NR1>

0 indicates no error

–340: error

Examples   `CALIBRATION:ALL` performs an internal calibration.

`CALIBRATION:ALL?` performs an internal calibration and returns results. For example, it might return 0, which indicates that the calibration completed without any errors.

*CLS (No Query Form)

This command clears all event registers and queues.

Group   Status

Syntax   `*CLS`

Related Commands   None

Examples   `*CLS` clears all the event registers and queues.

DIAGnostic:DATA? (Query Only)

This command returns the results of a self test.

Group   Diagnostic

Syntax   `DIAGnostic:DATA?`

Related Commands   DIAGnostic[:IMMediate], DIAGnostic:SELect

Returns   <NR1>

0 indicates no error.

–330 indicates that the self test failed.

Examples   `DIAGNOSTIC:DATA?` might return 0, which indicates that the diagnostics completed without any errors.

DIAGnostic[:IMMediate]

This command executes the selected self-test routines. The query form of this command executes the selected self-test routines and returns the results.

Group   Diagnostic

Syntax   `DIAGnostic[:IMMediate]`
`DIAGnostic[:IMMediate]?`

Related Commands   DIAGnostic:DATA?, DIAGnostic:SELect

Returns   <NR1>

0 indicates no error.

–330 indicates that the self test failed.

Examples   `DIAGNOSTIC:IMMEDIATE` executes the self test routines.

`DIAGNOSTIC:IMMEDIATE?` executes the self test routines. After the self test routines finish, the results of the self tests are returned.

DIAGnostic:SELect

This command selects the self-test routines. The query form of this command returns the selected test routine.

The following selections are available:

- ALL

- FPANel - Front panel read/write access test

- DTIMing – Data timing measurement (for AWG5000 series only)

- AREGister – AWG register read back

- A1Memory – CH1 waveform memory test

- A2Memory – CH2 waveform memory test

- A3Memory – CH3 waveform memory test (for AWG5000 series only)

- A4Memory – CH4 Waveform memory test (for AWG5000 series only)

- CREGister – Clock register read back

- CPLock – PLL Lock/unlock

- O1Register – Output1 register read back

- O1ALevel – Output1 analog level

- O1MLevel – Output1 Marker level (for AWG7000 series only)

- O2Register – Output2 register read back

- O2ALevel – Output2 analog level

- O2MLevel – Output2 marker level (for AWG7000 series only)

NOTE. Some of the selections are not available depending on the available options and the number of channels.

Group Diagnostic

Syntax `DIAGnostic:SELect`
`{ALL|FPANel|AREGister|DTIMing|A1Memory|A2Memory|A3Memory|`
`A4Memory|CREGister|CPLock|O1Register|O1ALevel|O1MLevel|`
`O2Register|O2ALevel|O2MLevel}`
`DIAGnostic:SELect?`

Related Commands DIAGnostic[:IMMediate]

Returns   ALL|FPAN|AREG|DTIM|A1M|A2M|A3M|A4M|CREG|CPL|O1R|O1AL|O1ML| O2R|O2AL|O2ML

Examples   `DIAGNOSTIC:SELECT FPANEL` selects the front panel read/write access test.

DISPlay[:WINDow[1|2]][:STATe]

This command minimizes or restores the sequence or waveform window of the arbitrary waveform generator. This command only minimizes or restores the display area; it does not close the window. There is no maximizing.

WINDow1 – Sequence window

WINDow2 – Waveform window

Group   Display

Syntax   `DISPlay[:WINDow[1|2]][:STATe] <display_state>`
              `DISPlay[:WINDow[1|2]][:STATe]?`

Related Commands   None

Arguments   `<display_state>`::=<Boolean>

0 indicates False, minimizes the window display.

1 indicates True, restores the window display.

At *RST, this value is 0 for window1 and 1 for window2.

Returns   <NR1>

Examples   `DISPLAY:WINDOW1:STATE 0` minimizes the sequence window.

*ESE

This command sets or queries the status of Event Status Enable Register.

Group   Status

Syntax	`*ESE <NR1>` `*ESE?`
Related Commands	*CLS, *ESR?, *SRE, *STB?
Arguments	<NR1>
Returns	<NR1>
Examples	`*ESE 177` sets the ESER to 177 (binary 10110001), which sets the PON, CME, EXE and OPC bits.

*ESR? (Query Only)

This query returns the status of Standard Event Status Register.

Group	Status
Syntax	`*ESR?`
Related Commands	*CLS, *ESE, *SRE, *STB?
Returns	<NR1>
Examples	`*ESR?` might return 181, which indicates that the SESR contains the binary number 10110101.

EVENt[:IMMediate] (No Query Form)

This command generates a forced event. This is used to generate the event when the sequence is waiting for an event jump (See SEQuence:ELEMent[n]:JTARget:TYPE).

This is equivalent to pressing the Force Event button on the front panel of the instrument.

Group	Event

Syntax   EVENt[:IMMediate]

Related Commands   EVENt:IMPedance, EVENt:JTIMing, EVENt:LEVel, EVENt:POLarity

Examples   EVENT:IMMEDIATE generates the event signal.

EVENt:IMPedance

This command and query sets or returns the impedance of the external event input. Valid values are 50 ohm or 1 kohm.

Group   Event

Syntax   EVENt:IMPedance <ohms>
EVENt:IMPedance?

Related Commands   EVENt[:IMMediate], EVENt:JTIMing, EVENt:LEVel, EVENt:POLarity

Arguments   <ohms>::=<NR3>

Valid values are 50 ohm or 1 kohm.

At *RST, this value is 1e3 ohm.

Returns   <NR3>

Examples   EVENT:IMPEDANCE 50 sets the impedance to 50 ohms.

EVENt:JTIMing

This command and query sets or returns the jump timing. Refer to the User Online Help for more information on jump timing.

Group   Event

Syntax   EVENt:JTIMing <jump_type>
EVENt:JTIMing?

Related Commands   EVENt[:IMMediate], EVENt:IMPedance, EVENt:LEVel, EVENt:POLarity

Arguments   `<jump_type>`::={SYNChronous|ASYNchronous}

SYNChronous indicates jump occurs immediately.

ASYNchronous indicates jump occurs after the signal generation is finished.

At *RST, this returns ASYNchronous.

Returns   SYNC|ASYN

Examples   `EVENT:JTIMING ASYNCHRONOUS` sets the jump to asynchronous type.

EVENt:LEVel

This command and query sets or returns the event level.

Group   Event

Syntax   `EVENt:LEVel <level>`
`EVENt:LEVel?`

Related Commands   EVENt[:IMMediate], EVENt:IMPedance, EVENt:JTIMing, EVENt:POLarity

Arguments   `<level>`::=`<NR3>`

Range is between 5 V and −5 V.

At *RST, this returns 1.4 V

Returns   `<NR3>`

Examples   `EVENT:LEVEL 1.0V` sets the level to 1 volt.

EVENt:POLarity

This command and query sets or returns the polarity of event signal. The Event Jump is the function to change the sequencing of the waveform by an event signal.

Group   Event

Syntax   EVENt:POLarity {POSitive|NEGative}
EVENt:POLarity?

Related Commands   EVENt[:IMMediate], EVENt:IMPedance, EVENt:JTIMing, EVENt:LEVel

Arguments   POSitive indicates that event jump occurs when the instrument receives a positive pulse.

NEGative indicates that event jump occurs when the instrument receives a negative pulse.

At *RST, this returns POSitive.

Returns   POS|NEG

Examples   EVENT:POLARITY NEGATIVE sets the event polarity to negative.

*IDN? (Query Only)

This command returns identification information for the arbitrary waveform generator.

Group   System

Syntax   *IDN?

Related Commands   None

Returns   <Manufacturer>, <model>, <serial number>, <Firmware version>

<Manufacturer>:: = TEKTRONIX

<Model>:: = AWG7122C, AWG7082C, AWG7121B, AWG7122B, AWG7061B, AWG7062B, AWG5012C, AWG5014C, AWG5002C, AWG5012B, AWG5014B, AWG5002B, AWG5004B

<Serial number>:: = XXXXXXX (indicates an actual serial number)

<Firmware version>:: = SCPI: 99.0 FW:x.x.x.x (x.x.x.x is system software version)

Examples   `*IDN?` might return the following response:

`TEKTRONIX,AWG7122B,B010123,SCPI:99.0 FW:3.0.136.602`

INSTrument:COUPle:SOURce

This command and query sets or returns the coupled state for a channel.

NOTE. When coupling is done, CH1 can be coupled to CH2, CH3 and CH4. CH3 can be coupled to CH4. Other combinations are not allowed.

When ALL is used, all other channels get the parameters of CH1.

When coupling is done, CH1 parameters are copied to CH2 parameters and CH3 parameters to CH4 parameters. This cannot be changed.

On two channel models, ALL is equivalent to PAIR.

On one channel models, only NONE is available.

In four channel models when PAIR is used, CH1 is coupled to CH2 and CH3 is coupled to CH4 in one action. Not all parameters are coupled.

When the coupling is active, setting the coupling state to NONE will remove the coupling.

Group   Instrument

Syntax   `INSTrument:COUPle:SOURce <state>`
`INSTrument:COUPle:SOURce?`

Related Commands   None

Arguments   `<state>`::={NONE|PAIR|ALL}

`NONE`

`PAIR` – CH1 to CH2 and CH3 to CH4

`ALL` – CH1 to CH2, CH3, and CH4

Returns   `<state>`

Examples   `INSTRUMENT:COUPLE:SOURCE ALL` couples the CH1 parameters and CH2 parameters if the instrument is a two-channel model.

MMEMory:CATalog? (Query Only)

This query returns the current contents and state of the mass storage media.

Group   Mass memory

Syntax   MMEMory:CATalog? [<msus>]

Related Commands   MMEMory:CDIRectory, MMEMory:MSIS

Arguments   <msus> (mass storage unit specifier)::=<string>

Returns   <NR1>,<NR1>[,<file_entry>]

The first <NR1> indicates the total amount of storage currently used in bytes.

The second <NR1> indicates the free space of mass storage in bytes.

<file_entry>>::= "<file_name>,<file_type>,<file_size>"

<file_name>:: = is the exact name of the file.

<file_type>::= is DIR for directory, otherwise it is blank.

<file_size>::=<NR1> is the size of the file in bytes.

Examples   MMEMORY:CATALOG? might return the following response:

484672,3878652,"SAMPLE1.AWG,,2948"

"aaa.txt,,1024","ddd,DIR,0","zzz.awg,,2948"

MMEMory:CDIRectory

This command and query sets or returns the current directory of the file system on the arbitrary waveform generator. The current directory for the programmatic interface is different from the currently selected directory in the Windows Explorer on the instrument.

Group   Mass memory

Syntax   MMEMory:CDIRectory [<directory_name>]
MMEMory:CDIRectory?

Related Commands   None

Arguments   `<directory_name>::=<string>`

Returns   <directory_name>

Examples   `MMEMORY:CDIRECTORY "/AWG/WORK0"` changes the current directory to /AWG/WORK0.

MMEMory:DATA

This command and query sets or returns block data to/from the file in the current mass storage device.

NOTE. The file is always transferred to the path mentioned along with the file name on the target.

If no path is specified with the file name, the current directory is used.

When path contains only the file name, current path is assumed.

This command has a limit of 650,000,000 bytes of data. If this limit is insufficient, consider the following alternatives:

■ Use a more efficient file encoding (WFM or PAT) when sending data.

■ Use instrument commands for direct control (WLISt:WAVeform:DATA, FREQ, VOLT, and so on).

■ Use Ethernet (ftp, http, or file sharing) to transfer the file.

Group   Mass memory

Syntax   `MMEMory:DATA <file_name>,<block_data>`
`MMEMory:DATA? <file_name>`

Related Commands   MMEMory:CDIRectory, MMEMory:MSIS

Arguments   `<file_name>,<block_data>`

Returns   Block_data – IEEE 488.2 data block

file_name – string having file name and path.

Examples MMEMORY:DATA "FILE1",#41024XXXXX... loads data into the file FILE1.

MMEMory:DELete (No Query Form)

This command deletes a file or directory from the instrument's hard disk. When used on a directory, this command succeeds only if the directory is empty.

Group Mass memory

Syntax MMEMory:DELete <file_name>[,<msus>]

Related Commands MMEMory:CDIRectory, MMEMory:MSIS

Arguments <file_name>::=<string>

<msus> (mass storage unit specifier)::=<string>

Examples MMEM:DEL "SETUP1.AWG" deletes SETUP1.AWG in the current directory.

MMEM:DEL "\my\proj\awg\test.awg","D:" deletes
D:\my\proj\awg\test.awg, regardless of the current directory
and the current msus.

MMEMory:EXPort (No Query Form)

This command exports a waveform file from the arbitrary waveform generator setup.

NOTE. If the waveform name already exists, it will be overwritten without warning.

The file name can contain a path and drive letter.

The supported file formats are:

TXT – Text file with analog data

TXT8 – Text file with 8-bit DAC resolution

TXT10 – Text file with 10-bit DAC resolution

TXT14 – Text file with 14-bit DAC resolution (AWG5000 Series only)

WFM – AWG400/AWG500/AWG600/AWG700 Series waveform

Group   Mass memory

Syntax   `MMEMory:EXPort <wfm_name>,<filename>,<type>`

Related Commands   None

Arguments   `<wfm_name>,<filename>,<type>`

`<wfm_name>::=<string>`

`<filename>::=<string>`

`<type>` = {TXT|TXT8|TXT10|TXT14|WFM}

Examples   `MMEMORY:EXPORT` "sine1024","sine1024.txt",txt exports a waveform named "sine1024" to a text file "sine1024.txt" with analog data.

MMEMory:IMPort (No Query Form)

This command imports a file into the arbitrary waveform generator's setup as a waveform.

NOTE. If the waveform name already exists, it will be overwritten without warning.

The file name can contain a path and drive letter.

The supported file formats are:

ISF – TDS3000 and DPO4000 waveform format

TDS – TDS5000/TDS6000/TDS7000, DPO7000/DPO70000/DSA70000 Series waveform

TXT – Text file with analog data

TXT8 – Text file with 8-bit DAC resolution

TXT10 – Text file with 10-bit DAC resolution

TXT14 – Text file with 14-bit DAC resolution

WFM – AWG400/AWG500/AWG600/AWG700 Series waveform

PAT – AWG400/AWG500/AWG600/AWG700 Series pattern file

TFW – AFG3000 Series waveform file format

IQT – RSA3000 Series waveform file format

TIQ – RSA6000 Series waveform file format

Group Mass memory

Syntax MMEMory:IMPort <wfm_name>,<filename>,<type>

Related Commands MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe], MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel, MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe], MMEMory:IMPort:PARameter:NORMalize

Arguments <wfm_name>,<filename>,<type>

<wfm_name>::=<string>

<filename>::=<string>

<type> = {ISF|TDS|TXT|TXT8|TXT10|TXT14|WFM|PAT|TFW}

Examples MMEMORY:IMPORT "sine1024","sine1024.txt",txt imports a waveform file named "sine1024", whose file format is text with normalized analog value.

MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe]

This command sets or queries the FREQuency parameter which determines whether frequency is modified during waveform import. If this value is set, the sampling rate is automatically updated during waveform import.

Group Mass memory

Syntax MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe] <state>
MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe]?

Related Commands MMEMory:IMPort

Arguments <state>::=<Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 1.

Returns <state>

Examples MMEMORY:IMPORT:PARAMETER:FREQUENCY:UPDATE:STATE 1 the instrument will automatically modify the sampling rate when importing the waveform data.

MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel

This command sets or queries the channel for which the amplitude and offset values will be updated during import.

NOTE. Channel number starts from 1 for CH1, 2 for CH2

Valid input depends on model number and interleave state

This command is effective only when MMEMory:IMPort:PARameter:LEVel[: UPDate][:STATe] is set to 1

Group Mass memory

Syntax MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel <NR1>
MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel?

Related Commands MMEMory:IMPort, MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]

Arguments <NR1>

At *RST, the value is 1.

Returns 1|2|3|4

Examples MMEMORY:IMPORT:PARAMETER:LEVEL:UPDATE:CHANNEL 1 sets the channel 1 amplitude and offset values to be updated when importing waveform data.

MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]

This command sets or queries the LEVel parameter which determines whether amplitude and offsets are modified during waveform import. If this value is set, the instrument amplitude and offset are automatically updated during waveform import.

Group	Mass memory

Syntax   MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe] <state>
MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]?

Related Commands   MMEMory:IMPort

Arguments   <state>::=<Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 1.

Returns   <Boolean>

Examples   MMEMORY:IMPORT:PARAMETER:LEVEL:UPDATE:STATE 1 the instrument will automatically modify the amplitude and offset when importing the waveform data.

MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE

This commands sets or queries the data to be imported. It also sets or queries which data's amplitude and offset values are selected for update during RSA file import.

This command is effective only when MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe] is set to True and IQT or TIQ is selected as the <Type> for the MMEMory:IMPort command.

Group   Mass memory

Syntax   MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE <Type>
MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE?

Related Commands   MMEMory:IMPort, MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]

Arguments   `<Type>::={IDATa|QDATa}`

IDATa indicates that the instrument imports I data.

QDATa indicates that the instrument imports Q data.

At *RST, the value is IDATa.

Returns   IDAT|QDAT

Examples   `MMEMORY:IMPORT:PARAMETER:LEVEL:UPDATE:TYPE IDATA` sets I data to be imported while importing an RSA file.

MMEMory:IMPort:PARameter:NORMalize

This command sets or queries if the imported data is normalized during text data import operation.

- The imported waveform data is normalized based on the option set in this command.

- When ZREFerence is selected, the offset is preserved during normalization operation.

- If FSCale is selected, offset is lost and full scale of the DAC is used for normalization.

- NONE will not normalize the data.

Group   Mass memory

Syntax   `MMEMory:IMPort:PARameter:NORMalize {NONE|FSCale|ZREFerence}`
`MMEMory:IMPort:PARameter:NORMalize?`

Related Commands   MMEMory:IMPort

Arguments   `<Normalization_type>`

NONE indicates that the imported data is not normalized.

FSCale indicates that the imported data is normalized with full DAC range.

ZREFerence indicates that the imported data is normalized with offset preserved.

At *RST, this returns NONE.

Returns   NONE|FSC|ZREF

Examples   MMEMORY:IMPORT:PARAMETER:NORMALIZE NONE will not normalize the imported data.

MMEMory:IMPort:PARameter:RESampling:FREQuency

This command sets or queries the sampling rate parameter for resampling. The specified sampling rate is applied to imported waveform.

NOTE. This command will take effect only when the MMEMory:IMPort: PARameter:FREQuency[:UPDate][:STATe] command is set to True.

Resampling setting is ignored if the MMEMory:IMPort:PARameter:FREQuency[: UPDate][:STATe] command is set to False.

Group   Mass memory

Syntax   MMEMory:IMPort:PARameter:RESampling:FREQuency <NR3>
MMEMory:IMPort:PARameter:RESampling:FREQuency?

Related Commands   MMEMory:IMPort, MMEMory:IMPort:PARameter:FREQuency[:UPDate][: STATe], MMEMory:IMPort:PARameter:RESampling[:STATe]

Arguments   <NR3>

At *RST, this returns the maximum sampling rate for the non-interleaved mode.

Returns   <NR3>

Examples   MMEMORY:IMPORT:PARAMETER:RESAMPLING:FREQUENCY 1.2E+9 sets the resampling frequency to 1.2 GHz.

MMEMory:IMPort:PARameter:RESampling[:STATe]

This command sets or queries the resampling state for waveform import. This command is effective only when the following conditions are met:

■ Waveform data to be imported must have sampling rate information.

■ MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe] command must be set to True.

Use this command to set the resampling state on or off. If you set the resampling state on, resampling is automatically invoked when importing waveform data. The query form of this command returns the resampling state of the instrument.

Group   Mass memory

Syntax   MMEMory:IMPort:PARameter:RESampling[:STATe] <state>
MMEMory:IMPort:PARameter:RESampling[:STATe]?

Related Commands   MMEMory:IMPort, MMEMory:IMPort:PARameter:FREQuency[:UPDate][: STATe], MMEMory:IMPort:PARameter:RESampling:FREQuency

Arguments   <state>::=<Boolean>

0 indicates False

1 indicates True

At *RST, this returns 0.

Returns   <Boolean>

Examples   MMEMORY:IMPORT:PARAMETER:FREQUENCY:UPDATE:STATE 1 invokes resampling automatically.

MMEMory:MDIRectory (No Query Form)

This command creates a new directory in the current path on the mass storage system.

Group   Mass memory

Syntax   MMEMory:MDIRectory <directory_name>

Related Commands   MMEMory:CDIRectory, MMEMory:MSIS

Arguments   `<directory_name>`::=<string> specifies a new directory.

Examples   `MMEMORY:MDIRECTORY "WAVEFORM"` makes the directory "WAVEFORM".

MMEMory:MSIS

This command and query selects or returns a mass storage device used by all MMEMory commands. <msus> specifies a drive using a drive letter. The drive letter can represent hard disk drives, network drives, DVD/CD-RW drives, or USB memory.

Group   Mass memory

Syntax   `MMEMory:MSIS [<msus>]`
`MMEMory:MSIS?`

Related Commands   None

Arguments   `<msus>` (mass storage unit specifier)::=<string>

Returns   <msus> (mass storage unit specifier)::=<string>

At *RST, this values is "C:"

Examples   `MMEMORY:MSIS?` might return the following response: "X:"

*OPC

This command is used to ensure that the first command is complete before the second command is issued. Always returns one on this instrument.

Group   Synchronization

Syntax   `*OPC`
`*OPC?`

Related Commands   *WAI

Returns <NR1>

<NR1>=1 when all pending operations are finished.

Examples *OPC? might return 1 to indicate that all pending OPC operations are finished.

*OPT? (Query Only)

This command returns the implemented options for the arbitrary waveform generator.

Group System

Syntax *OPT?

Related Commands None

Returns <opt>[, <opt> [, <opt>]]]

<opt>::= {0|01|02|03|06|08}

Examples *OPT? might return 0 to indicate that no option is installed in the instrument.

OUTPut[n]:FILTer[:LPASs]:FREQuency

This command and query sets or returns the low-pass filter frequency of the filter. INFinity is same as Through (no filter). This command is not available on instruments with option 02 or option 06.

Group Output

Syntax OUTPut[n]:FILTer[:LPASs]:FREQuency {<NR3>|INFinity}
OUTPut[n]:FILTer[:LPASs]:FREQuency?

Related Commands AWGControl:DOUTput[n][:STATe]

Arguments <NR3>

At *RST, this value returns 9.9e37 (INFinity).

Returns   <NR3>

Examples   `OUTPUT1:FILTER:LPASS:FREQUENCY 200MHZ` sets the cutoff frequency of the low-pass filter for CH 1 to 200 MHz.

OUTPut[n][:STATe]

This command and query sets or returns the output state of the arbitrary waveform generator. Setting the output state of a channel to ON will switch on its analog output signal and marker.

Group   Output

Syntax   `OUTPut[n][:STATe] <output_state>`
`OUTPut[n][:STATe]?`

Related Commands   None

Arguments   `<output_state>::=<Boolean>`

0 sets the channel output to False (OFF)

1 sets the channel output to True (ON)

At *RST, this returns 0.

Returns   <NR1>

Examples   `OUTPUT1:STATE ON` turns the channel 1 output on.

*RST (No Query Form)

This command resets the arbitrary waveform generator to its default state.

Group   System

Syntax   `*RST`

Related Commands   None

Examples   `*RST` resets the instrument.

SEQuence:ELEMent[n]:GOTO:INDex

This command and query sets or returns the target index for the GOTO command of the sequencer.

After generating the waveform specified in a sequence element, the sequencer jumps to the element specified as GOTO target. This is an unconditional jump. If GOTO target is not specified, the sequencer simply moves on to the next element. If the Loop Count is Infinite, the GOTO target which is specified in the element is not used. For this command to work, the SEQuence:ELEMent[n]:GOTO:STATe must be ON and the sequence element must exist.

Note that the first element of a sequence is taken to be 1 not 0.

Group   Sequence

Syntax   `SEQuence:ELEMent[n]:GOTO:INDex <target>`
`SEQuence:ELEMent[n]:GOTO:INDex?`

Related Commands   SEQuence:ELEMent[n]:GOTO:STATe, SEQuence:LENGth

Arguments   `<target>`::=`<NR1>`

Returns   `<target>`

Examples   `SEQUENCE:ELEMENT1:GOTO:INDEX 6` causes the sequencer to jump to sixth element after executing the first element.

`SEQUENCE:ELEMENT1:GOTO:INDEX?` might return 6.

SEQuence:ELEMent[n]:GOTO:STATe

This command and query sets or returns the GOTO state of the sequencer. For the SEQuence:ELEMent[n]:GOTO:INDex command to take effect, the GOTO state must be set to ON.

Group   Sequence

Syntax   SEQuence:ELEMent[n]:GOTO:STATe <goto_state>
SEQuence:ELEMent[n]:GOTO:STATe?

Related Commands   SEQuence:ELEMent[n]:GOTO:INDex, SEQuence:LENGth

Arguments   <goto_state>::=<Boolean>

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

The value of <n> is an index number of sequence.

Returns   <NR1>

Examples   SEQUENCE:ELEMENT1:GOTO:STATE 1 sets the GOTO state to ON.

SEQuence:ELEMent[n]:JTARget:INDex

This command and query sets or returns the target index for the sequencer's event jump operation. Note that this will take effect only when SEQuence:ELEMent[n]:JTARget:TYPE is set to INDex.

Group   Sequence

Syntax   SEQuence:ELEMent[n]:JTARget:INDex <target>
SEQuence:ELEMent[n]:JTARget:INDex?

Related Commands   SEQuence:ELEMent[n]:JTARget:TYPE, SEQuence:LENGth

Arguments   <target>::=<NR1>

<n> is an index number of sequence.

Returns   <NR1>

Examples   SEQUENCE:ELEMENT1:JTARGET:INDEX 10 sets the jump target index to 10th element.

SEQuence:ELEMent[n]:JTARget:TYPE

This command and query sets or returns the event jump target type for the jump. Generate an event in one of the following ways:

- By connecting an external cable to instrument rear panel for external event.

- By pressing the Force Event button on the front panel.

- By sending the EVENt[:IMMediate] remote command.

Group   Sequence

Syntax   SEQuence:ELEMent[n]:JTARget:TYPE {INDex|NEXT|OFF}
SEQuence:ELEMent[n]:JTARget:TYPE?

Related Commands   SEQuence:ELEMent[n]:JTARget:INDex, SEQuence:LENGth

Arguments   INDex. This enables the sequencer to jump to an index set using SEQuence:ELEMent1:JTARget:INDex command.

NEXT. This enables the sequencer to jump to the next sequence element. SEQuence:ELEMent1:JTARget:INDex setting is ignored.

OFF. This enables the sequencer to turn off the event jump state. In this state, even if the event occurs, the sequencer ignores it.

AT *RST, this value is OFF.

The value of n is an index number of sequence.

Returns   IND|NEXT|OFF

Examples   SEQUENCE:ELEMENT1:JTARGET:TYPE INDEX sets the jump target to INDex.

SEQuence:ELEMent[n]:LOOP:COUNt

This command and query sets or returns the loop count. Loop count setting for an element is ignored if SEQuence:ELEMent[n]:LOOP:INFinite is set to ON.

Group   Sequence

Syntax   SEQuence:ELEMent[n]:LOOP:COUNt <NR1>
SEQuence:ELEMent[n]:LOOP:COUNt?

Related Commands SEQuence:ELEMent[n]:LOOP:INFinite, SEQuence:LENGth

Arguments `<NR1>`

The value ranges between 1 and 65,536.

At *RST, this returns 1.

The value of n is an index number of sequence.

Returns `<NR1>`

Examples `SEQUENCE:ELEMENT:LOOP:COUNT 100` sets the element loop count to 100.

SEQuence:ELEMent[n]:LOOP:INFinite

This command and query sets or returns the infinite looping state for a sequence element. When an infinite loop is set on an element, the sequencer continuously executes that element. To break the infinite loop, either issue the AWGControl:STOP[:IMMediate] command or change the run mode to Continuous by using AWGControl:RMODe command.

Group Sequence

Syntax `SEQuence:ELEMent[n]:LOOP:INFinite <loop_state>`
`SEQuence:ELEMent[n]:LOOP:INFinite?`

Related Commands SEQuence:ELEMent[n]:LOOP:COUNt, SEQuence:LENGth

Arguments `<loop_state>`::=`<Boolean>`

`0` indicates OFF

`1` indicates ON

At *RST, this returns 0.

The value of `<n>` is an index number of sequence.

Returns `<NR1>`

Examples `SEQUENCE:ELEMENT1:LOOP:INFINITE 1` sets the infinite flag to ON.

SEQuence:ELEMent[n]:SUBSequence

This command and query sets or returns the subsequence for a sequence element.

The AWG5012B, AWG5000C, and AWG7000C (option 09) series instruments support Subsequence.

The subsequence name can be a null string (""). When a waveform is assigned to this sequence, the command returns "".

Group	Subsequence
Syntax	SEQuence:ELEMent[n]:SUBSequence <subseq_name> SEQuence:ELEMent[n]:SUBSequence?
Related Commands	All Sequence and Subsequence group commands
Arguments	<subseq_name>::=<string>
Returns	<subseq_name>
Examples	SEQUENCE:ELEMENT10:SUBSEQUENCE "MYTEST" inserts the subsequence named "mytest" at array position 10 of the main sequence. SEQUENCE:ELEMENT10:SUBSEQUENCE? might return "mytest", which indicates the sequence name.

SEQuence:ELEMent[n]:TWAit

This command and query sets or returns the wait trigger state for an element. Send a trigger signal in one of the following ways:

- By using an external trigger signal.

- By pressing the "Force Trigger" button on the front panel.

- By sending the *TRG remote command.

Group	Sequence
Syntax	SEQuence:ELEMent[n]:TWAit <Boolean> SEQuence:ELEMent[n]:TWAit?

Related Commands   SEQuence:LENGth

Arguments   `<wait_trigger_state>`::=`<Boolean>`

0 indicates OFF

1 indicates ON

At *RST, this returns 0.

In the OFF state, the sequencer ignores trigger signals.

The value of <n> is an index number of sequence.

NOTE. The instrument without option 08 always sets Wait Trigger On. Trying to set the wait trigger state to off in an instrument without option 08 will cause an error.

Returns   `<NR1>`

Examples   `SEQUENCE:ELEMENT1:TWAIT 1` sets the wait trigger state to ON.

SEQuence:ELEMent[n]:WAVeform[m]

This command and query sets or returns the waveform for a sequence element.

NOTE. The value of n indicates index number of sequence.

The value of m = 1|2|3|4 is based on the model. If the suffix is omitted, 1 is assumed.

The value of m indicates the channel that will output the waveform when the sequence is run.

The length of all the waveforms specified for a sequence element must be equal.

Group   Sequence

Syntax   `SEQuence:ELEMent[n]:WAVeform[m] [1|2|3|4] <wfm_name>`
`SEQuence:ELEMent[n]:WAVeform[m] [1|2|3|4]?`

Related Commands   SEQuence:LENGth

Arguments <wfm_name>::=<string>

Returns <wfm_name>

Examples `SEQUENCE:ELEMENT1:WAVEFORM1 "*TRIANGLE1000"` sets the "*Triangle1000" waveform into the first element of the sequence.

`SEQUENCE:ELEMENT20:WAVEFORM1?` might return "*Sine1000" indicating that the waveform named *Sine1000 is assigned to index number 20 of the channel 1 sequence.

SEQuence:JUMP[:IMMediate] (No Query Form)

This command forces the sequencer to jump to the specified element index. This is called a Force jump. This command does not require an event for executing the jump. Also, the Jump target specified for event jump is not used here.

Group Sequence

Syntax `SEQuence:JUMP[:IMMediate] <target>`

Related Commands None

Arguments `<target>`::=<NR1>

Examples `SEQUENCE:JUMP:IMMEDIATE 10` forces the sequencer to jump to index number 10.

SEQuence:LENGth

This command and query sets or returns the sequence length. Use this command to create an uninitialized sequence. You can also use the command to clear all sequence elements in a single action by passing 0 as the parameter. However, this action cannot be undone so exercise necessary caution. Also note that passing a value less than the sequence's current length will cause some sequence elements to be deleted at the end of the sequence. For example if SEQuence:LENGth? returns 200 and you subsequently send SEQuence:LENGth 21, all sequence elements except the first 20 will be deleted.

Group Sequence

Syntax SEQuence:LENGth <NR1>
SEQuence:LENGth?

Related Commands None

Arguments <NR1>

At *RST, this returns 0.

Returns <NR1>

Examples SEQUENCE:LENGTH 10 creates a sequence of 10 elements initializing all sequence parameters to default values.

SEQUENCE:LENGTH? will now return 10.

SEQUENCE:LENGTH 12 will append two elements to the end of the above created sequence and initialize the new elements' parameters. However, it does not change the already existing elements.

SEQUENCE:LENGTH 0 will delete the sequence.

SLISt:NAME? (Query Only)

The query returns the name of the subsequence corresponding to the specified index in the subsequence list.

Group Subsequence

Syntax SLISt:NAME? <Index>

Related Commands SLISt:SIZE?

Arguments <Index>::=<NR1>

Returns <NR1>

Examples SLIST:NAME? 3 returns the name of the third subsequence in the subsequence list provided the subsequence list has three or more elements.

SLISt:SIZE? (Query Only)

This query returns the size of the subsequence list.

Group	Subsequence
Syntax	SLISt:SIZE?
Related Commands	SLISt:NAME?
Returns	<NR1>
Examples	SLIST:SIZE? returns the number of existing subsequences.

SLISt:SUBSequence:DELete (No Query Form)

This command deletes the subsequence from the currently loaded setup.

NOTE. The subsequence will be deleted even if it is a part of the sequence.

When a subsequence is deleted in the main sequence, the string "EMPTY" will replace the previous subseq_name.

When you specify ALL, all the subsequences in the subsequence list are deleted in a single action.

Group	Subsequence
Syntax	SLISt:SUBSequence:DELete {<subseq_name>|ALL}
Related Commands	SLISt:NAME?, SLISt:SUBSequence:NEW
Arguments	<subseq_name>::=<string>
Examples	SLIST:SUBSEQUENCE:DELETE ALL deletes all the subsequence from the currently loaded setup. The ALL parameter does not delete predefined waveforms.
	SLIST:SUBSEQUENCE:DELETE "MYTEST" deletes the subsequence named "mytest".

SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt

This command and query sets or returns the loop count for the specified subsequence element. The loop count is an integer.

Group   Subsequence

Syntax   SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt <subseq_name>,<NR1>
SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt? <subseq_name>

Arguments   <NR1>

At *RST, this returns 0.

Returns   <NR1>

Examples   SLIST:SUBSEQUENCE:ELEMENT5:LOOP:COUNT "MYTEST",11 sets the loop count for the index element five of the subsequence named "mytest" to 11.

SLIST:SUBSEQUENCE:ELEMENT5:LOOP:COUNT? "MYTEST" might return 11, which indicates that the loop count for the index element five of the sequence named "mytest" is set to 11.

SLISt:SUBSequence:ELEMent[n]:WAVeform[n]

This command and query sets or returns the waveform for an element of the subsequence.

The value of n = 1 | 2 | 3 | 4 based on the model. If suffix is not specified, the value of n is 1.

The value of 'n' indicates which channel will output the waveform when the sequence is run.

NOTE. The waveform name can be a null string (" "). However, a sequence with " " as one of its elements cannot be run.

Group   Subsequence

Syntax   SLISt:SUBSequence:ELEMent[n]:WAVeform[n]
<subseq_name>,<wfm_name>
SLISt:SUBSequence:ELEMent[n]:WAVeform[n]? <subseq_name>

Related Commands SLISt:NAME?, SLISt:SUBSequence:NEW

Arguments `<subseq_name>`::=`<string>`

`<wfm_name>`::=`<string>`

Returns <wfm_name>

Examples `SLIST:SUBSEQUENCE:ELEMENT5:WAVEFORM2 "MYTEST","*SINE360"` sets the waveform "*Sine360" to the fifth index element of the CH2 subsequence named "mytest".

`SLIST:SUBSEQUENCE:ELEMENT5:WAVEFORM2? "MYTEST"` might return "*Sine360", which indicates that the waveform named "*Sine360" is assigned to index element five of the CH2 subsequence named "mytest".

SLISt:SUBSequence:LENGth

This command and query sets or returns the size of the subsequence.

Group Subsequence

Syntax `SLISt:SUBSequence:LENGth <subseq_name>,<NR1>`
`SLISt:SUBSequence:LENGth? <subseq_name>`

Arguments `<subseq_name>`::=`<string>`

Returns <NR1>

Examples `SLIST:SUBSEQUENCE:LENGTH "MYTEST",101` changes the length of the subsequence named "mytest" to 101.

`SLIST:SUBSEQUENCE:LENGTH? "MYTEST"` might return 101, which indicates that the length of the subsequence named "mytest" is 101.

SLISt:SUBSequence:NEW (No Query Form)

This command creates a new subsequence.

Group Subsequence

Syntax   `SLISt:SUBSequence:NEW <subseq_name>,<length>`

Arguments   `<subseq_name>`::=`<string>`

Examples   `SLIST:SUBSEQUENCE:NEW "MYTEST",100` creates a subsequence named "`mtest`" of length 100.

SLISt:SUBSequence:TSTamp? (Query Only)

This query returns the time stamp of the subsequence.

Time stamp is updated whenever the subsequence is created or changed. It is not updated when it is renamed. It returns date as a string in the form "yyyy/mm/dd hh:mm:ss" (a white space between data and time).

Group   Subsequence

Syntax   `SLISt:SUBSequence:TSTamp?  <subseq_name>`
`SLISt:SUBSequence:TSTamp?`

Related Commands   SLISt:NAME?, SLISt:SUBSequence:NEW

Arguments   `<subseq_name>`::=`<string>`

Returns   `<string>`

Examples   `SLIST:SUBSEQUENCE:TSTAMP? "MYTEST"` returns the time stamp of the subsequence named "mytest".

[SOURce[1]]:FREQuency[:CW|:FIXed]

This command and query sets or returns the sampling frequency of the arbitrary waveform generator. Sampling frequency can be set when the internal clock source is selected and one of the following conditions is met:

■   Internal is selected as Reference Source.

■   External is selected as Reference Source and Fixed is selected as External Reference Type.

CW and FIXed are aliases and have the same effect.

Note that the frequency of the waveform output by the instrument is calculated as:

$$OutputFrequency = \frac{SamplingFrequency}{NumberOfPoints}$$

The minimum number of points in a waveform for the instrument is 1.

Group   Source

Syntax   [SOURce[1]]:FREQuency[:CW|:FIXed] <NR3>
[SOURce[1]]:FREQuency[:CW|:FIXed]?

Related Commands   [SOURce[n]]:WAVeform, AWGControl:INTerleave[:STATe]

Arguments   <NR3>. The value must be between 10 MHz to 10 GHz.

At *RST, this returns 1.2000000E+10.

Returns   <NR3>

Examples   SOURCE1:FREQUENCY 10MHZ sets the frequency to 10 MHz.

[SOURce[1]]:ROSCillator:FREQuency

This command selects the reference oscillator frequency. Valid values are 10 MHz, 20 MHz and 100 MHz. This command is used when the Clock Source is Internal and Reference Input is External and External Reference Type is Fixed.

Group   Source

Syntax   [SOURce[1]]:ROSCillator:FREQuency <NR3>
[SOURce[1]]:ROSCillator:FREQuency?

Related Commands   [SOURce[1]]:ROSCillator:SOURce, [SOURce[1]]:ROSCillator:TYPE

Arguments   <NR3>

At *RST, this returns 10 MHz.

Returns   <NR3>

Examples   SOURCE1:ROSCILLATOR:FREQUENCY 10MHZ sets the reference oscillator source frequency to 10 MHz.

SOURCE1:ROSCILLATOR:FREQUENCY? might return 1.00000000E+7.

[SOURce[1]]:ROSCillator:MULTiplier

This command and query sets or returns the ROSCillator multiplier rate. This parameter is valid only when Clock Source is Internal and Reference Source is External and External Reference Type is Variable.

Group   Source

Syntax   [SOURce[1]]:ROSCillator:MULTiplier <NR1>
[SOURce[1]]:ROSCillator:MULTiplier?

Related Commands   [SOURce[1]]:ROSCillator:SOURce, [SOURce[1]]:ROSCillator:TYPE

Arguments   <NR1>

At *RST, this returns 1.

Returns   <NR1>

Examples   SOURCE1:ROSCILLATOR:MULTIPLIER 10 sets the multiplier rate to 10.

SOURCE1:ROSCILLATOR:MULTIPLIER? might return 10.

[SOURce[1]]:ROSCillator:SOURce

This command selects the reference oscillator source. INTernal means that the reference frequency is derived from the internal precision oscillator. EXTernal means the reference frequency is derived from an external signal supplied through the Reference Clock Input connector.

Group   Source

Syntax   `[SOURce[1]]:ROSCillator:SOURce {INTernal|EXTernal}`
`[SOURce[1]]:ROSCillator:SOURce?`

Related Commands   [SOURce[1]]:ROSCillator:FREQuency, [SOURce[1]]:ROSCillator:TYPE

Arguments   `<INTernal|EXTernal>`

`INTernal` – The reference frequency is derived from the internal precision oscillator.

`EXTernal` – The reference frequency is derived from an external signal supplied through the reference clock input.

At *RST, this returns INTernal.

Returns   INT|EXT

Examples   `SOURCE1:ROSCILLATOR:SOURCE INTERNAL` selects the internal clock source.

[SOURce[1]]:ROSCillator:TYPE

This command selects the type of the reference oscillator. This parameter is valid only when Clock Source is Internal and Reference Source is External.

Group   Source

Syntax   `[SOURce[1]]:ROSCillator:TYPE {FIXed|VARiable}`
`[SOURce[1]]:ROSCillator:TYPE?`

Related Commands   [SOURce[1]]:ROSCillator:FREQuency, [SOURce[1]]:ROSCillator:SOURce

Arguments   `FIXed|VARiable`

`FIXed`: Selects a reference source whose frequency is fixed to 10MHz, 20MHz, or 100MHz. Select one of these frequencies using the [SOURce[1]]:ROSCillator:FREQuency command.

`VARiable`: Selects a reference source whose frequency is not fixed.

At *RST, this returns FIXed.

Returns   FIX|VAR

Examples SOURCE1:ROSCILLATOR:TYPE FIXED selects a fixed frequency external reference oscillator. The frequency is fixed to 10 MHz, 20 MHz, or 100 MHz.

[SOURce[n]]:COMBine:FEED (AWG5000 Series only)

This command adds the signal from an external input to the output of the channel.

> *NOTE. When the signal addition is enabled, the return value is "ESIG". It is always in uppercase.*
>
> *When setting the parameter either "ESIGnal" or "ESIG" can be specified.*
>
> *While setting the parameter, it is case insensitive.*

Group Source

Syntax [SOURce[n]]:COMBine:FEED {"ESIGnal"|""}
[SOURce[n]]:COMBine:FEED?

Related Commands None

Arguments "ESIGnal" – Adds the input from the external signal.

"" – Removes the signal feed.

At *RST, this returns "".

The value of n is 1,2,3 or 4 depending on the number of channels available. Specifying "" as input cancels the setting.

Returns "ESIG"|""

Examples SOURCE1:COMBINE:FEED "ESIGNAL" adds an external signal to the channel 1 output signal.

[SOURce[n]]:DAC:RESolution

This command and query sets or returns the DAC resolution.

NOTE. DAC supports 8-bit and 10-bit resolutions only for AWG7000 Series instruments.

When the resolution changes to 10-bit, marker data will not be available.

AWG5000 Series instruments support only 14-bit resolution. Therefore this command will have no effect for these instruments.

DAC resolution is independent for each channel in AWG7000 Series instruments.

Group   Source

Syntax   [SOURce[n]]:DAC:RESolution <NR1>
[SOURce[n]]:DAC:RESolution?

Related Commands   [SOURce[n]]:MARKer[1|2]:DELay, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude], [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet

Arguments   <NR1>

8 sets the DAC resolution to 8 bits.

10 sets the DAC resolution to 10 bits.

The value of n indicates the channel number.

At *RST, this returns 8.

Returns   <NR1>

Examples   SOURCE1:DAC:RESOLUTION 10 sets the channel 1 resolution to 10 bits.

[SOURce[n]]:DELay[:ADJust]

This command and query sets or returns the delay (in seconds) of the analog output.

NOTE. The effect of this command can be seen only in non-sequence mode.

This command does not change the waveform display on the user interface.

Group Source

Syntax [SOURce[n]]:DELay[:ADJust] <NR3>
[SOURce[n]]:DELay[:ADJust]?

Related Commands [SOURce[n]]:DELay:POINts, [SOURce[n]]:PDELay:HOLD, [SOURce[n]]:PHASe[:ADJust]

Arguments <NR3>

At *RST, this returns 0 s.

Returns <NR3>

Examples SOURCE1:DELAY:ADJUST 20PS sets the analog output delay for channel 1 to 20 picoseconds.

[SOURce[n]]:DELay:POINts

This command and query sets or returns the delay (in points) of the analog output.

NOTE. The effect of this command can be seen only in non-sequence mode.

This command does not change the waveform display on the user interface.

Group Source

Syntax [SOURce[n]]:DELay:POINts <NR3>
[SOURce[n]]:DELay:POINts?

Related Commands [SOURce[n]]:DELay[:ADJust], [SOURce[n]]:PDELay:HOLD, [SOURce[n]]:WAVeform

Arguments <NR3>

At *RST, this returns 0 points.

Returns <NR3>

Examples SOURCE1:DELAY:POINTS 20 sets the analog output delay for channel 1 to 20 points.

[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude] (AWG5000 Series only)

This command and query sets or returns the amplitude of digital output. This command is available only for AWG500B and AWG5000C instruments with option 03.

Group Source

Syntax [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude]
<NR3>
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude]?

Related Commands [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH, [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW, [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 1 V_{pp}.

Returns <NR3>

Examples SOURCE1:DIGITAL:VOLTAGE:LEVEL:IMMEDIATE:AMPLITUDE 1.4 sets the digital output amplitude level to 1.4 volts.

[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH (AWG5000 Series only)

This command and query sets or returns the high digital output. This command is available only for AWG5000B and AWG5000C instruments with option 03.

Group Source

Syntax [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH <NR3>
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH?

Related Commands [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW,
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude],
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 1 V.

Returns <NR3>

Examples SOURCE1:DIGITAL:VOLTAGE:LEVEL:IMMEDIATE:HIGH 1.4 sets the digital
output high level to 1.4 volts.

[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW (AWG5000 Series only)

This command and query sets or returns the low digital output. This command is
available only for AWG5000B and AWG5000C instruments with option 03.

Group Source

Syntax [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW <NR3>
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW?

Related Commands [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH,
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude],
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 0 V.

Returns <NR3>

Examples SOURCE1:DIGITAL:VOLTAGE:LEVEL:IMMEDIATE:LOW 0.5 sets the digital
output high level to 0.5 volts.

[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet (AWG5000 Series only)

This command and query sets or returns the offset of digital output. This command is available only for AWG5000B and AWG5000C instruments with option 03.

Group Source

Syntax [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet <NR3>
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet?

Related Commands [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH, [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW, [SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude]

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 0.5 V.

Returns <NR3>

Examples SOURCE1:DIGITAL:VOLTAGE:LEVEL:IMMEDIATE:OFFSET 1.0 sets the digital output amplitude level to 1.0 volts.

[SOURce[n]]:FUNCtion:USER

This command and query sets or returns the waveform to waveform memory.

- Use this command to directly load an AWG400/500/600/700 series waveform (WFM), pattern file (PAT), or sequence (SEQ) file from mass memory to a specified channel.

- However, when loading a sequence file, the SOURce's suffix is ignored.

- The waveform is internally converted to the AWG5000/AWG7000 series format and inserted into the current waveform list. To successfully load a waveform, the waveform name should conform to AWG5000/AWG7000 series waveform naming conventions.

- If you specify the SEQ file, Run Mode is changed to Sequence.

- If you specify a WFM or PAT file while the Run Mode is Sequence, Run Mode is changed to Continuous.

- When you import a sequence file (*.SEQ) for the AWG400/500/600/700 series using this command, all the user-defined waveforms are deleted before the import operation.

Group   Source

Syntax   [SOURce[n]]:FUNCtion:USER <Waveform file_name>[,<msus>]
[SOURce[n]]:FUNCtion:USER?

Related Commands   WLISt:NAME?

Arguments   <Waveform file_name>:: = <string>

Value of n indicates the channel number.

At *RST, this returns " ","C:".

Returns   <file_name>

Examples   SOURCE1:FUNCTION:USER "SAMPLE1.WFM" loads sample1.wfm into waveform list and also sets it as the output waveform of channel1.

[SOURce[n]]:MARKer[1|2]:DELay

This command and query sets or returns the marker delay. Marker delay is independent for each channel.

In the AWG7000 Series when DAC resolution is changed to 10 bits, marker output is not available. However, marker related parameters can be modified using SCPI commands.

Group   Source

Syntax   `[SOURce[n]]:MARKer[1|2]:DELay <NR3>`
`[SOURce[n]]:MARKer[1|2]:DELay?`

Related Commands   [SOURce[n]]:DAC:RESolution

Arguments   `<NR3>`

The value of n indicates the channel number.

At *RST, this returns 0.

Returns   <NR3>

Examples   `SOURCE1:MARKER1:DELAY 20PS` sets the marker1 delay of channel1 to 20 picoseconds.

`SOURCE1:MARKER1:DELAY?` might return 2.00000000E -011 indicating 20 ps.

[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude]

This command sets the marker amplitude.

In the AWG7000 Series, when the DAC resolution is changed to 10 bits, marker output is not available. However, marker related parameters can be modified using SCPI commands.

Group   Source

Syntax   `[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:`
`AMPLitude] <NR3>`
`[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:`
`AMPLitude]?`

Related Commands   [SOURce[n]]:DAC:RESolution

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 1 Vpp.

Returns <NR3>

Examples SOURCE1:MARKER1:VOLTAGE:AMPLITUDE 0.5V sets the channel1 marker1amplitude to 0.5 volts.

SOURCE1:MARKER1:VOLTAGE:AMPLITUDE? might return 0.5 volts.

[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH

This command sets the marker high level.

In the AWG7000 Series, when DAC resolution is changed to 10 bits, marker output is not available. However, marker related parameters can be modified using SCPI commands.

Refer to the User Online Help for the setting range of marker high and marker low.

Group Source

Syntax [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH <NR3>
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH?

Related Commands [SOURce[n]]:DAC:RESolution, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns 1 V.

Returns <NR3>

Examples SOURCE1:MARKER1:VOLTAGE:HIGH 0.75 sets the marker1 high to 0.75 volts.

[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW

This command sets the marker low level.

In the AWG7000 Series, when the DAC resolution is changed to 10 bits, marker output is not available. However, marker related parameters can be modified using SCPI commands.

Refer to the User Online Help for the setting range of marker high and marker low.

Group   Source

Syntax   [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW
               <NR3>
               [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW?

Related Commands   [SOURce[n]]:DAC:RESolution, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:
                         IMMediate]:HIGH

Arguments <NR3>

               The value of n indicates the channel number.

               At *RST, this returns 0 V.

Returns   <NR3>

Examples   SOURCE1:MARKER1:VOLTAGE:LOW 0.5 sets the marker1 low to 0.5 volts.

[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet

This command sets the marker offset.

In the AWG7000 Series, when the DAC resolution is changed to 10 bits, marker output is not available. However, marker related parameters can be modified using SCPI commands.

Group   Source

Syntax   [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet
               <NR3>
               [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet?

Related Commands [SOURce[n]]:DAC:RESolution, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH, [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW

Arguments `<NR3>`

The value of n indicates the channel number.

At *RST, this returns 0.5 V.

Returns `<NR3>`

Examples `SOURCE1:MARKER1:VOLTAGE:OFFSET 1.0` sets the offset channel1 marker1 to 1 V.

[SOURce[n]]:PDELay:HOLD

This command and query sets or returns the parameter that is retained when sampling rate or waveform length is changed.

NOTE. The effect of this command can be seen only in non-sequence mode.

Group Source

Syntax `[SOURce[n]]:PDELay:HOLD {PHASe|DELay|POINt}`
`[SOURce[n]]:PDELay:HOLD?`

Related Commands [SOURce[n]]:DELay[:ADJust], [SOURce[n]]:DELay:POINts, [SOURce[n]]:PHASe[:ADJust]

Arguments `PHASe|DELay|POINt`

At *RST, this returns PHASe.

Returns PHAS|DEL|POIN

Examples `SOURCE1:PDELAY:HOLD PHASE` will retain the channel 1 phase value when the instrument sampling rate or waveform length is changed.

[SOURce[n]]:PHASe[:ADJust] (AWG5000 Series only)

This command and query sets or returns the phase of the analog output.

NOTE. The effect of this command can be seen only in non-sequence mode.

This command does not change the waveform display on the user interface.

Group Source

Syntax [SOURce[n]]:PHASe[:ADJust] <NR3>
[SOURce[n]]:PHASe[:ADJust]?

Related Commands [SOURce[n]]:DELay[:ADJust], [SOURce[n]]:DELay:POINts, [SOURce[n]]:PDELay:HOLD

Arguments <NR3>

At *RST, this returns 0 degree.

Returns <NR3>

Examples SOURCE1:PHASE:ADJUST 180 sets the analog output phase for channel 1 to 180 degrees.

[SOURce[n]]:SKEW

This command and query sets or returns the skew for the waveform associated with a channel.

Group Source

Syntax [SOURce[n]]:SKEW <NR3>
[SOURce[n]]:SKEW?

Related Commands None

Arguments   <NR3>

–100 ps to +100 ps. It can be changed by a minimum of 1 ps at a time.

The value of n indicates the channel number.

At *RST, this returns 0 s.

Returns   <NR3>

Examples   SOURCE2:SKEW 75PS sets the skew for channel2 to 75 ps.

SOURCE2:SKEW? might return 7.50000000E-011, indicating that the skew is 75 ps.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]

This command and query sets or returns the amplitude for the waveform associated with a channel.

Group   Source

Syntax   [SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] <NR3>
[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]?

Related Commands   None

Arguments   <NR3> in the range 50 mV to 2V pk-pk.

The value of n indicates the channel number.

At *RST, this returns 1 Vpp.

Returns   <NR3>

Examples   SOURCE1:VOLTAGE:AMPLITUDE 1.5 sets the amplitude of channel1 to 1.5 volts.

SOURCE1:VOLTAGE:AMPLITUDE? might return 1.5.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH

This command and query sets or returns the high voltage level for the waveform associated with a channel. The command is not available on instruments with the Option 02 or Option 06 installed.

Group   Source

Syntax   [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH <NR3>
              [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH?

Related Commands   [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW

Arguments   <NR3>

The value of n indicates the channel number.

At *RST, this returns 0.5 V.

Returns   <NR3>

Examples   SOURCE1:VOLTAGE:HIGH 0.75 sets the channel1's high to 0.75 volts.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW

This command and query sets or returns the low voltage level for the waveform associated with a channel. The command is not available on instruments with Option 02 or Option 06 installed.

Group   Source

Syntax   [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW <NR3>
              [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW?

Related Commands   None

Arguments   <NR3>

The value of n indicates the channel number.

At *RST, this returns –0.5 V.

Returns <NR3>

Examples SOURCE1:VOLTAGE:LOW 0.25 sets the channel1 low to 0.25 volts.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet

This command and query sets or returns the offset for the waveform associated with a channel. The command is not available on instruments with Option 02 or Option 06 installed.

Group Source

Syntax [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet <NR3>
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet?

Related Commands [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH, [SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW, AWGControl:DOUTput[n][:STATe]

Arguments <NR3>

The value of n indicates the channel number.

At *RST, this returns –0.5 V.

Returns <NR3>

Examples SOURCE1:VOLTAGE:LEVEL:IMMEDIATE:OFFSET 0.5 sets the channel 1 offset to 0.5 V.

[SOURce[n]]:WAVeform

This command and query sets or returns the output waveform from the current waveform list for each channel when Run Mode is not Sequence. However, this command cannot be used to load a waveform stored in an AWG400/500/600/700 waveform or pattern file. To load a waveform stored in an AWG400/500/600/700 waveform or pattern file, use the [SOURce[n]]:FUNCtion:USER command.

Group Source

Syntax   [SOURce[n]]:WAVeform <wfm_name>
[SOURce[n]]:WAVeform?

Related Commands   [SOURce[n]]:FUNCtion:USER

Arguments   ʼ.
<wfm_name>::=<string>

The value of n indicates the channel number.

Returns   <wfm_name>

Examples   SOURCE1:WAVEFORM "*SINE100" loads a predefined waveform called
"*Sine100" into channel1 memory.

SOURCE1:WAVEFORM? might return "*Sine100".

*SRE

This command sets or queries the bits in Service Request Enable register.

Group   Status

Syntax   *SRE <NR1>
*SRE?

Related Commands   *CLS, *ESE, *ESR?, *STB?

Arguments   <NR1>

Returns   <NR1>

Examples   *SRE 48 sets the bits in the SRER to the binary value 00110000.

*SRE? might return a value of 32, showing that the bits in the SRER have the
binary value 00100000.

STATus:OPERation:CONDition? (Query Only)

This query returns the contents of the Operation Condition Register.

Note that the OCR is not used in the arbitrary waveform generator.

Group Status

Syntax `STATus:OPERation:CONDition?`

Related Commands STATus:OPERation:ENABle, STATus:OPERation[:EVENt]?

Returns <NR1>

STATus:OPERation:ENABle

This command and query sets or returns the mask for the Operation Enable Register.

Note that the OENR is not used in the arbitrary waveform generator.

Group Status

Syntax `STATus:OPERation:ENABle <NR1>`
`STATus:OPERation:ENABle?`

Related Commands STATus:OPERation:CONDition?, STATus:OPERation[:EVENt]?

Arguments <NR1>

Returns <NR1>

STATus:OPERation[:EVENt]? (Query Only)

This query returns the contents of Operation Event Register.

Note that the OEVR is not used in the arbitrary waveform generator.

Group Status

Syntax   `STATus:OPERation[:EVENt]?`

Related Commands   STATus:OPERation:CONDition?, STATus:OPERation:ENABle

Returns   <NR1>

STATus:PRESet (No Query Form)

This command sets the OENR and QENR registers.

Group   Status

Syntax   `STATus:PRESet`

Related Commands   None

Examples   `STATUS:PRESET` resets the SCPI enable registers.

STATus:QUEStionable:CONDition? (Query Only)

This query returns the status of the Questionable Condition Register.

Note that the QCR is not used in the arbitrary waveform generator.

Group   Status

Syntax   `STATus:QUEStionable:CONDition?`

Related Commands   STATus:QUEStionable:ENABle, STATus:QUEStionable[:EVENt]?

Returns   <NR1>

STATus:QUEStionable:ENABle

This command and query sets or returns the mask for Questionable Enable Register.

Note that the QENR is not used in the arbitrary waveform generator.

Group Status

Syntax STATus:QUEStionable:ENABle <NR1>
STATus:QUEStionable:ENABle?

Related Commands STATus:QUEStionable:CONDition?, STATus:QUEStionable[:EVENt]?

Returns <NR1>

STATus:QUEStionable[:EVENt]? (Query Only)

This query returns the status of the QEVR register and clears it.

Group Status

Syntax STATus:QUEStionable[:EVENt]?

Related Commands STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABle

Returns <NR1>

Examples STATUS:QUESTIONABLE:EVENT? might return 32, which indicates that the
QEVR contains the binary number 00000000 00100000.

*STB? (Query Only)

This query returns the contents of Status Byte Register.

Group Status

Syntax *STB?

Related Commands *CLS, *ESE, *ESR?, *SRE

Returns   <NR1>

Examples   `*STB?` might return 96, which indicates that the SBR contains the binary number 0110 0000.

SYSTem:DATE

This command and query sets or returns the system date. When the values are nonintegers, they are rounded off to nearest integral values.

Group   System

Syntax   `SYSTem:DATE <year>,<month>,<day>`
                 `SYSTem:DATE?`

Related Commands   None

Arguments   `<year>`::=<NRf> (Four digit number)

                     `<month>`::=<NRf> from 1 to 12

                     `<day>`::=<NRf> from 1 to 31

Returns   <year>,<month>,<day>

Examples   `SYSTEM:DATE 2008,6,20` sets the date to June 20, 2008.

SYSTem:ERRor[:NEXT]? (Query Only)

This command retrieves and returns data from the error and event queues.

Group   System

Syntax   `SYSTem:ERRor[:NEXT]?`

Related Commands   None

Returns   <Error / event number>, <error / event description [;device dependant info]>

0 – No Error

Error / event number <NR1>

error / event description <string>

Examples   SYSTEM:ERROR:NEXT? might return the following response:

−102,"Syntax error;possible invalid suffix - :SOUR:FREQ 2V"

This response indicates that the unit is invalid.

SYSTem:KLOCk

This command locks or unlocks the keyboard and front panel of the arbitrary waveform generator.

Group   System

Syntax   SYSTem:KLOCk <state>
SYSTem:KLOCk?

Related Commands   None

Arguments   <state>::=<Boolean>

0 indicates False. The front panel and keyboard are unlocked.

1 indicates True. The front panel and keyboard are locked.

At *RST, this returns 0.

*NOTE. When you manually set the Touch Screen button to OFF on the front panel, the *RST will not reset to 0.*

Returns   <NR1>

Examples   SYSTEM:KLOCK ON locks the front panel and keyboard.

SYSTEM:KLOCK might return 1, which indicates that the front panel and keyboard are locked.

SYSTem:TIME

This command and query sets or returns the system time. When the values are nonintegers, they are rounded off to nearest integral values.

Group System

Syntax `SYSTem:TIME <hour>,<minute>,<second>`
`SYSTem:TIME?`

Related Commands None

Arguments `<hour>,<minute>,<second>`

`<hour>`::=<NRf> from 0 to 23

`<minute>`::=<NRf> from 0 to 59

`<second>`::=<NRf> from 0 to 59

Returns <hour>,<minute>,<second>

Examples `SYSTEM:TIME 11,23,58` sets the time.

SYSTem:VERSion? (Query Only)

This command returns the SCPI version number to which the command conforms.

Group System

Syntax `SYSTem:VERSion?`

Related Commands None

Returns <NR2>::=YYYY.V

where YYYY represents the year version and V represents an approved revision number for that year.

Examples `SYSTEM:VERSION?` might return 1999.0.

*TRG (No Query Form)

This command generates a trigger event. This is equivalent to pressing Trig button on front panel.

Group	Trigger
Syntax	`*TRG`
Related Commands	TRIGger[:SEQuence][:IMMediate]
Examples	`*TRG` generates a trigger event.

TRIGger[:SEQuence][:IMMediate] (No Query Form)

This command generates a trigger event. This is equivalent to *TRG.

Group	Trigger
Syntax	`TRIGger[:SEQuence][:IMMediate]`
Related Commands	*TRG
Examples	`TRIGGER:SEQUENCE:IMMEDIATE` generates the trigger event.

TRIGger[:SEQuence]:IMPedance

This command and query sets or returns the trigger impedance. It applies only to the external trigger.

Group	Trigger
Syntax	`TRIGger[:SEQuence]:IMPedance <impedance>` `TRIGger[:SEQuence]:IMPedance?`
Related Commands	None

Arguments <impedance>::=<NR3> the value will be 50 and 1000.

At *RST, this returns 1000 Ω.

Returns <NR3>

Examples TRIGGER:SEQUENCE:IMPEDANCE 50 selects 50 Ω impedance for the external trigger input.

TRIGger[:SEQuence]:LEVel

This command and query sets or returns the trigger input level (threshold).

Group Trigger

Syntax TRIGger[:SEQuence]:LEVel <NR3>
TRIGger[:SEQuence]:LEVel?

Related Commands TRIGger[:SEQuence]:SOURce

Arguments <NR3>

At *RST, this returns 1.4 V.

Returns <NR3>

Examples TRIGGER:SEQUENCE:LEVEL 200MV sets the trigger level to 200 mV.

TRIGger[:SEQuence]:MODE (AWG7000B and AWG7000C Series only)

This command and query sets or returns the trigger timing. It is used in the Triggered or Sequence mode. Trigger timing can be set when the external trigger source is selected.

Group Trigger

Syntax TRIGger[:SEQuence]:MODE <trigger_type>
TRIGger[:SEQuence]:MODE?

Related Commands TRIGger[:SEQuence]:SOURce

Arguments <trigger_type>::={SYNChronous|ASYNchronous}

SYNChronous does not lower trigger jitter.

ASYNchronous lowers trigger jitter.

At *RST, this returns ASYNchronous.

Returns SYNC|ASYN

Examples TRIGGER:SEQUENCE:MODE ASYNCHRONOUS sets the trigger timing to asynchronous type.

TRIGger[:SEQuence]:POLarity

This command and query sets or returns the trigger input polarity. It is used to set polarity in gated mode.

Group Trigger

Syntax TRIGger[:SEQuence]:POLarity {POSitive|NEGative}
TRIGger[:SEQuence]:POLarity?

Related Commands AWGControl:RMODe, TRIGger[:SEQuence]:LEVel

Arguments POSitive means the gate signal is activated when the external trigger signal is greater (more Positive) than the trigger level.

NEGative means the gate signal is activated when the external trigger signal is less (more Negative) than the trigger level.

At *RST, this returns POSitive.

Returns POS|NEG

Examples TRIGGER:SEQUENCE:POLARITY NEGATIVE selects the Negative polarity.

TRIGger[:SEQuence]:SLOPe

This command and query sets or returns the trigger slope. It is used to set polarity in modes other than gated mode.

Group	Trigger

Syntax
```
TRIGger[:SEQuence]:SLOPe {POSitive|NEGative}
TRIGger[:SEQuence]:SLOPe?
```

Related Commands TRIGger[:SEQuence]:SOURce

Arguments POSitive means that the event occurs on the rising edge of the external trigger signal.

NEGative means that the event occurs on the falling edge of the external trigger signal.

At *RST, this returns POSitive.

Returns POS|NEG

Examples TRIGGER:SEQUENCE:SLOPE NEGATIVE selects the Negative slope.

TRIGger[:SEQuence]:SOURce

This command and query sets or returns the trigger source.

Group	Trigger

Syntax
```
TRIGger[:SEQuence]:SOURce {INTernal|EXTernal}
TRIGger[:SEQuence]:SOURce?
```

Related Commands TRIGger[:SEQuence]:LEVel, TRIGger[:SEQuence]:POLarity, TRIGger[:SEQuence]:SLOPe, TRIGger[:SEQuence]:TIMer

Arguments INTernal selects internal clock as the trigger source.

EXTernal selects external clock as the trigger source.

At *RST, this returns EXTernal.

Returns   INT|EXT

Examples   TRIGGER:SEQUENCE:SOURCE INTERNAL selects the internal clock as the trigger source.

TRIGger[:SEQuence]:TIMer

This command and query sets or returns the internal trigger rate (trigger interval).

Group   Trigger

Syntax   TRIGger[:SEQuence]:TIMer <NR3>
TRIGger[:SEQuence]:TIMer?

Related Commands   TRIGger[:SEQuence]:SOURce

Arguments   <NR3>

At *RST, this returns 100 ms.

Returns   <NR3>

Examples   TRIGGER:SEQUENCE:TIMER 5MS sets the internal trigger rate to 5 ms.

TRIGger[:SEQuence]:WVALue

This command and query sets or returns the output data position of a waveform while the instrument is in the waiting-for-trigger state. This is valid only when Run Mode is Triggered or Gated.

Group   Trigger

Syntax   TRIGger[:SEQuence]:WVALue {FIRSt|LAST}
TRIGger[:SEQuence]:WVALue?

Related Commands   TRIGger[:SEQuence]:SOURce

Arguments FIRSt specifies the first value of the waveform as the output level.

LAST specifies the last value of the waveform as the output level.

At *RST, this returns FIRSt.

Returns FIRS|LAST

Examples TRIGGER:SEQUENCE:WVALUE LAST selects the last value as the output level.

*TST? (Query Only)

This query executes a self test and returns the results.

Group Diagnostic

Syntax *TST?

Related Commands DIAGnostic[:IMMediate], DIAGnostic:DATA?, DIAGnostic:SELect

Returns <NR1>

0 indicates no error.

Examples *TST? might return –330 indicating that the self test failed.

*WAI (No Query Form)

This command prevents the arbitrary waveform generator from executing further commands until all pending commands are executed.

Group Synchronization

Syntax *WAI

Related Commands *OPC

Examples   `*WAI` prevents the execution of any commands or queries until all pending operations complete.

WLISt:NAME? (Query Only)

This query returns the waveform name of an element in the waveform list. This query can be used to query the waveform name in the waveform list.

Group   Waveform

Syntax   `WLISt:NAME? <Index>`

Related Commands   None

Arguments   `<Index>`::=<NR1>

Returns   <string>::=<wfm_name> is the waveform name specified by <index>.

Examples   `WLIST:NAME? 21` might return "untitled21".

WLISt:SIZE? (Query Only)

This query returns the size (number of waveforms) of the waveform list. Names of both predefined and user-created waveforms are stored in a single list. The maximum size depends on the length of each waveform.

Group   Waveform

Syntax   `WLISt:SIZE?`

Related Commands   WLISt:WAVeform:NEW

Returns   <NR1>

At *RST, this returns the number of predefined waveforms.

Examples   `WLIST:SIZE?` might return 25 when user-defined waveform list is empty.

WLISt:WAVeform:DATA

This command transfers waveform data from the external controller into the waveform list or from the waveform list to the external control program.

NOTE. Before transferring data to the instrument, a waveform must be created using the WLISt:WAVeform:NEW command.

Use this command to set both analog and marker data. To set only the marker data, use the WLISt:WAVeform:MARKer:DATA command.

Using StartIndex and Size, part of a waveform can be transferred at a time. Very large waveforms can be transferred in chunks.

Waveform data is always transferred in the LSB first format.

The format of the transferred data depends on the waveform type.

If Size is omitted, the length of waveform is assumed to be the value of the Size parameter.

Transferring large waveforms in chunks allows external programs to cancel the operation before it is completed.

The instrument supports two types of waveform data: integer format and floating point format. The integer format occupies two bytes per waveform data point. Floating point waveform data points occupy five bytes. So the total bytes will be five times the size of the waveform. The first four bytes of each data point represent the floating point representation of the sample value and the fifth byte stores the marker data. The marker data occupy the two most significant bits of the fifth byte.

The minimum size of the waveform must be 1 and the maximum size depends on the instrument model and configuration.

This command has a limit of 650,000,000 bytes of data. If this limit is insufficient, consider the following alternatives:

- Use a more efficient file encoding (WFM or PAT) when sending data.

- Use instrument commands for direct control (WLISt:WAVeform:DATA, FREQ, VOLT, and so on).

- Use Ethernet (ftp, http, or file sharing) to transfer the file.

Refer to the User Online Help, **AWG Reference > Waveform General Information** section for the detailed format specification.

Group   Waveform

Syntax	`WLISt:WAVeform:DATA`
	`<wfm_name>[,<StartIndex>[,<Size>]],<block_data>`
	`WLISt:WAVeform:DATA? <wfm_name>[,<StartIndex>[,<Size>]]`

Related Commands   WLISt:WAVeform:NEW, WLISt:WAVeform:MARKer:DATA

Arguments   `StartIndex, Size,<block_data>`

`<wfm_name>`::=<string>

`<StartIndex>`::=<NR1>

`<Size>`::=<NR1>

`<block_data>`::=<IEEE 488.2 block>

Returns   <block_data>

Examples   `WLIST:WAVEFORM:DATA` "TestWfm",0,1024,#42048xxxx... this transfers waveform data to a waveform called "TestWfm" created earlier using the `WLISt:WAVeform:NEW` command. The data size is 1024 points (2048 bytes) and the start index is 1 (the first data point). Note that the IEEE 488.2 block header depends on the type of the data being transferred. If it is integer type, the total bytes will be twice the size of the waveform and if it is a real waveform, the total bytes will be five times the size of the waveform.

WLISt:WAVeform:DELete (No Query Form)

This command deletes the waveform from the currently loaded setup.

NOTE. The waveform will be deleted even if it is a part of the sequence

The sequence element corresponding to the deleted waveform will have WFMID_EMPTY

When ALL is specified, all user-defined waveforms in the list are deleted in a single action. Note that there is no "UNDO" action once the waveforms are deleted. Use caution before issuing this command.

If the deleted waveform is currently loaded into waveform memory, it is unloaded. If the RUN state of the instrument is ON, the state is turned OFF. If the channel is on, it will be switched off.

Group   Waveform

Syntax   WLISt:WAVeform:DELete {<wfm_name>|ALL}

Related Commands   WLISt:SIZE?

Arguments   <wfm_name>::=<string>

Examples   WLIST:WAVEFORM:DELETE ALL deletes all user-defined waveforms from the currently loaded setup. The ALL parameter does not delete predefined waveforms.

WLIST:WAVEFORM:DELETE "Test1" deletes a waveform called "Test1".

WLISt:WAVeform:LENGth? (Query Only)

This query returns the size of the waveform. The returned value represents data points (not bytes).

Group   Waveform

Syntax   WLISt:WAVeform:LENGth? <wfm_name>

Related Commands   WLISt:WAVeform:NEW

Arguments   <wfm_name>::=<string>

Returns   <NR1>

Examples   WLIST:WAVEFORM:LENGTH? "*Sine 360" might return 360.

WLISt:WAVeform:MARKer:DATA

This command sets or queries the waveform marker data.

NOTE. This command returns or sends only marker data for the waveform.

Each marker data occupies one bit. Two most significant bits of each byte are used for marker1 and marker2 (bit 6 for marker1 and bit 7 for marker2). For more information about the waveform data format, refer to the AWG Reference > Waveform General Information section of the User Online Help.

You will have to use bit masks to obtain the actual value.

When used on a waveform with n data points, you get only n bytes, each byte having values for both markers.

Block data can be sent in batches using "Size" and "StartIndex" parameters.

This command has a limit of 650,000,000 bytes of data. If this limit is insufficient, consider the following alternatives:

- Use a more efficient file encoding (WFM or PAT) when sending data.

- Use instrument commands for direct control (WLISt:WAVeform:DATA, FREQ, VOLT, and so on).

- Use Ethernet (ftp, http, or file sharing) to transfer the file.

Group   Waveform

Syntax   WLISt:WAVeform:MARKer:DATA
<wfm_name>[,<StartIndex>[,<Size>]],<block_data>
WLISt:WAVeform:MARKer:DATA?
<wfm_name>[,<StartIndex>[,<Size>]]

Related Commands   None

Arguments   <wfm_name>::=<string>

<StartIndex>::=<NR1>

<Size>::=<NR1>

<block_data>::=<IEEE 488.2 block>

Returns   <block_data>

Examples   WLIST:WAVEFORM:MARKER:DATA "myWaveform",0,1000,#41000….

WLIST:WAVEFORM:MARKER:DATA? "myWaveform",0,1000

WLISt:WAVeform:NEW (No Query Form)

This command creates a new empty waveform in the waveform list of current setup.

Group   Waveform

Syntax   WLISt:WAVeform:NEW <wfm_name>,<Size>,<Type>

Related Commands   WLISt:WAVeform:DATA

Arguments   <wfm_name>::=<string>

<Size>::=<NR1>

<Type>::={REAL|INTeger}

Examples   WLIST:WAVEFORM:NEW "Test1", 1024, INTEGER creates a new integer waveform called Test1 with 1024 points.

WLISt:WAVeform:NORMalize (No Query Form)

This command normalizes a waveform that exists in the waveform list of the current setup.

Group   Waveform

Syntax   WLISt:WAVeform:NORMalize <wfm_name>,<Type>

Related Commands   None

Arguments   <wfm_name>::=<string>

<Type>::={FSCale|ZREFerence}

Examples WLIST:WAVEFORM:NORMALIZE "Untitled25", FSC normalizes the waveform titled "Untitled25", if it exists, using full scale.

WLISt:WAVeform:PREDefined? (Query Only)

This query returns true or false based on whether the waveform is predefined.

NOTE. Predefined waveforms have fixed length and name. Therefore, renaming or deleting them is not possible.

Creating a new waveform with the same name as the predefined waveform is not possible.

Data of a predefined waveform can be transferred to an external controller using WLISt:WAVeform:DATA command.

Group Waveform

Syntax WLISt:WAVeform:PREDefined? <wfm_name>

Related Commands None

Arguments <wfm_name>::=<string>

Returns <state>::=<Boolean>

Examples WLIST:WAVEFORM:PREDEFINED? "*Sine3600" might return 1 indicating that it is a predefined waveform.

WLISt:WAVeform:RESAmple (No Query Form)

This command resamples a waveform that exists in the waveform list of the current setup.

Group Waveform

Syntax WLISt:WAVeform:RESAmple <wfm_name>,<Size>

Related Commands None

Arguments <wfm_name::=<string>

<Size>::=<NR1>

Examples WLIST:WAVEFORM:RESAMPLE "Untitled25", 1024 resamples the waveform titled "Untitled25", if it exists, to 1024 points.

WLISt:WAVeform:TSTamp? (Query Only)

This query returns the time stamp of the waveform.

NOTE. Time stamp is updated whenever the waveform is created or changed. It is not updated when it is renamed.

The command returns date as a string in the form yyyy/mm/dd hh:mm:ss (a white space between date and time).

Time stamp for predefined waveforms is null string ("").

Group Waveform

Syntax WLISt:WAVeform:TSTamp? <wfm_name>

Related Commands None

Arguments <wfm_name>::=<string>

Returns "yyyy/mm/dd hh:mm:ss" is the waveform time stamp.

Where

yyyy refers to a four-digit year number mm refers to two-digit month number from 01 to 12

dd refers to two-digit day number in the month

hh refers to two-digit hour number mm refers to two-digit minute number

ss refers to two-digit second number

Examples   WLIST:WAVEFORM:TSTAMP? "Sine" might return the date and time the "Sine" waveform was created or last modified.

WLIST:WAVEFORM:TSTAMP? "*DC" might return "" because "*DC" is a predefined waveform.

WLISt:WAVeform:TYPE? (Query Only)

This query returns the type of the waveform.

Group   Waveform

Syntax   WLISt:WAVeform:TYPE? <wfm_name>

Related Commands   WLISt:WAVeform:NEW

Arguments   <wfm_name>::=<string>

Returns   INT|REAL

Examples   WLIST:WAVEFORM:TYPE? "*Ramp1000" might return REAL.

Status and Events

Status and Event Reporting

Status Reporting Structure

The arbitrary waveform generator status reporting functions conform to IEEE-488.2 and SCPI standards. Use the status reporting function to check for instrument errors and to identify the types of events that have occurred on the instrument.

The status reporting function is separated into three functional blocks:

- Standard/Event Status

- Operation Status

- Questionable Status

The operations processed in these three blocks are summarized in status bytes, which provide the error and event data. The following figure is a diagram of the instrument's status reporting function.
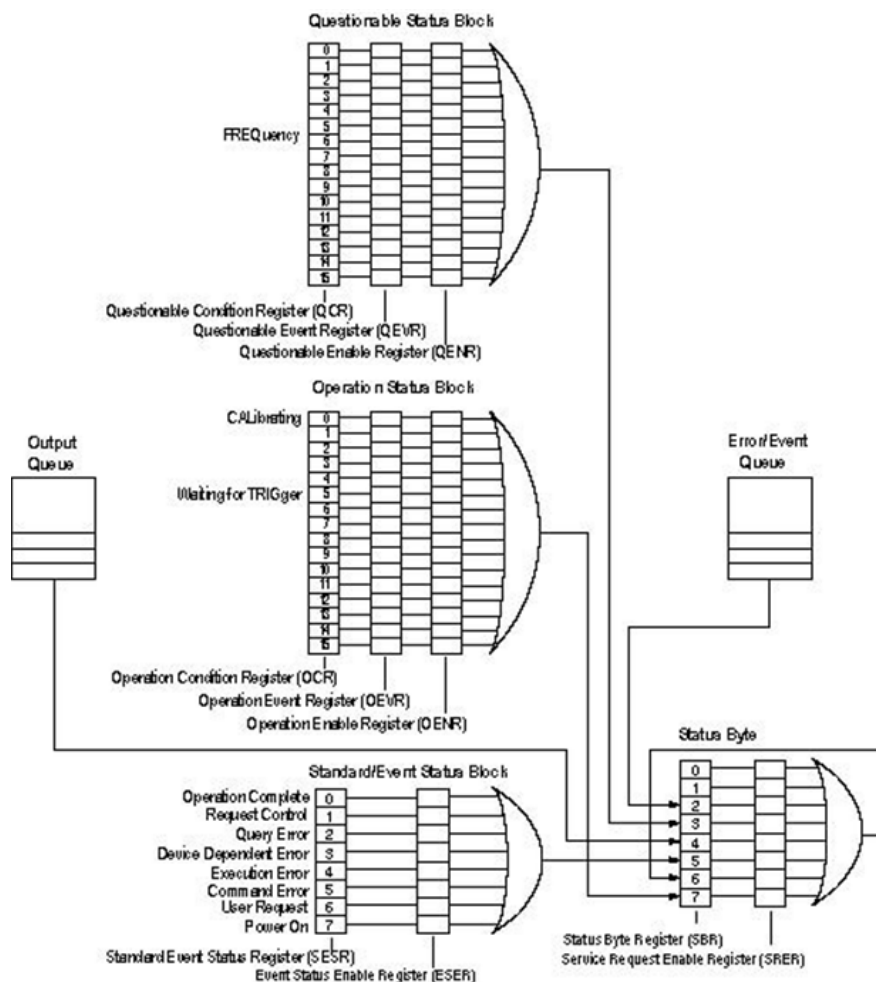
Registers

There are two main types of registers:

- Status Registers: store data relating to instrument status. These registers are set by the arbitrary waveform generator.

- Enable Registers: determine whether to set events that occur in the instrument to the appropriate bits in the status registers and event queues. You can set this register.

Status Registers

There are six types of status registers:

- Status Byte Register (SBR)

- Standard Event Status Register (SESR)

- Operation Condition Register (OCR)

- Operation Event Register (OEVR)

- Questionable Condition Register (QCR)

Status Byte Register (SBR)

The Status Byte Register (SBR) is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1987 (see the following figure and table). These bits are used to monitor the output queue, SESR, and service requests. The contents of this register are returned when the *STB? query is used.

The following figure shows the bit values of the SBR.
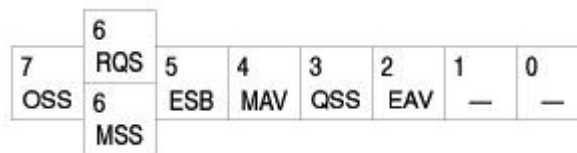
The following table lists the SBR bit functions.

Table 3-1: SBR bit functions

Bit	Function
7	Operation Summary Status (OSS).
6	RQS (Request Service)/MSS (Master Summary Status). When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (that the GPIB bus SRQ is LOW). The RQS bit is cleared when the serial poll ends.
	When the instrument is accessed using the *STB? query, this bit is called the Master Summary Status (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query.
5	Event Status Bit (ESB). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed.
4	Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved.
3	Questionable Summary Status (QSS)

Table 3-1: SBR bit functions (cont.)

Bit	Function
2	Event Queue Available (EAV)
1-0	Not used

Standard Event Status Register (SESR)

The Standard Event Status Register (SESR) is made up of 8 bits. Each bit records the occurrence of a different type of event, shown in following figure. The contents of this register are returned when the *ESR? query is used.

The following figure shows the bit values of the SESR.

The following table lists the SESR bit functions.

Table 3-2: SESR bit functions

Bit	Function
7	Power On (PON). Indicates that the instrument was powered on.
6	Not used.
5	Command Error (CME). Indicates that a command error has occurred while parsing was in progress.
4	Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons: ■ A value designated in the argument is outside the allowable range of the instrument, or is in conflict with the instrument's capabilities. ■ The conditions for execution differed from those essentially required.
3	Device Specific Error (DDE). An instrument error has been detected.
2	Query Error (QYE). Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons: ■ An attempt was made to retrieve messages from the output queue though the output queue is empty or in pending status. ■ The output queue messages have been cleared though they have not been retrieved.
1	Not used
0	Operation Complete (OPC). This bit is set with the results of the execution of the *OPC command. It indicates that all pending operations have been completed.

Operation Enable Register (OENR)

None of the bits in the Operation Enable Register are used.

Operation Condition Register (OCR)

None of the bits in the Operation Condition Register are used.

Operation Event Register (OEVR)

None of the bits in the Operation Event Register are used.

Questionable Condition Register (QCR)

None of the bits in the Questionable Condition Register are used.

Enable Registers

There are four types of enable registers:

- Event Status Enable Register (ESER)

- Service Request Enable Register (SRER)

- Operation Enable Register (OENR)

- Questionable Enable Register (QENR)

Each bit in the enable registers corresponds to a bit in the controlling status register. By setting and resetting the bits in the enable register, you can determine whether or not events that occur will be registered to the status register and queue.
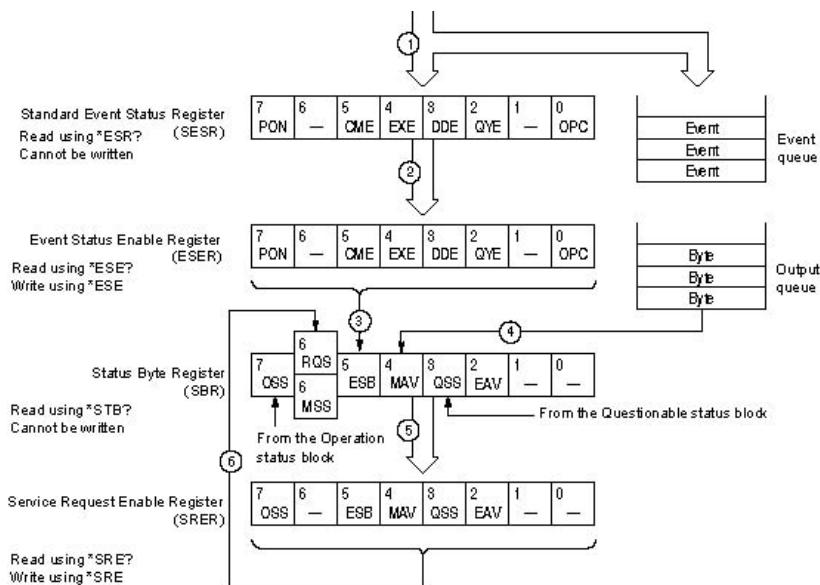
Event Status Enable Register (ESER)

The ESER is made up of bits defined exactly the same as bits 0 through 7 in the SESR. Use this register to designate whether or not the SBR ESB bit should be set when an event has occurred, and to determine if the corresponding SESR bit is set.

To set the SBR ESB bit (when the SESR bit has been set), set the ESER bit corresponding to that event. To prevent the ESB bit from being set, reset the ESER bit corresponding to that event.

Use the `*ESE` command to set the bits of the ESER. Use the `*ESE`? query to read the contents of the ESER.

7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR. Use this register to define which events will generate service requests.

The SRER bit 6 cannot be set. Also, the RQS is not maskable.

The generation of a service request with the GPIB interface involves changing the SRQ line to LOW, and making a service request to the controller. The result is that a status byte for which an RQS has been set is returned in response to serial polling by the controller.

Use the `*SRE` command to set the bits of the SRER. Use the `*SRE?` query to read the contents of the SRER. Bit 6 must be set to 0.

7	6	5	4	3	2	1	0
OSS	—	ESB	MAV	QSS	EAV	—	—

Questionable Enable Register (QENR)

None of the bits in the Questionable Enable Register are used.

Queues

There are two types of queues in the status reporting system: output queues and error/event queues.

Output Queue The output queue is a FIFO (first-in, first-out) queue that holds response messages to queries awaiting retrieval. When there are messages in the queue, the SBR MAV bit is set.

The output queue is emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error occurs and the output queue is emptied; however, the operation proceeds even if an error occurs.

Error/Event Queue The event queue is a FIFO queue, which stores events as they occur in the instrument. If more than 100 events are stored, the 100th event is replaced with event code –350 ("Queue Overflow").

The oldest error code and text are retrieved by using one of the following queries:

`SYSTem:ERROr[:NEXT]?`

First, issue the `*ESR?` query to read the contents of the SESR. The contents of the SESR are cleared after they are read. If an SESR bit is set, events are stacked in the Error/Event Queue. Retrieve the event code with the following command sequence:
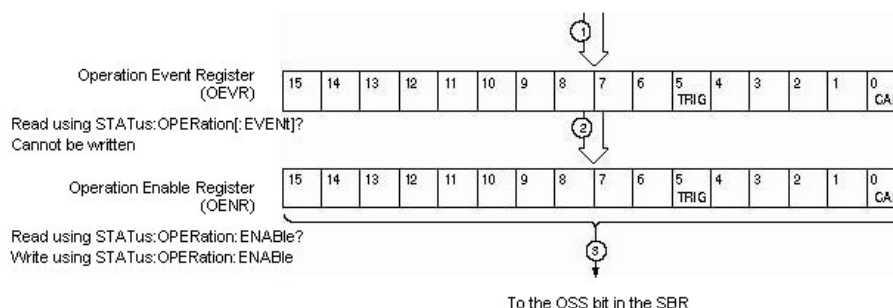
`*ESR?`

`SYSTem:ERROr[:NEXT]?`

If you omit the `*ESR?` query, the SESR bit will remain set, even if the event disappears from the Error/Event Queue.

Operation Status Block

This block is used to report on the status of several operations being executed by the arbitrary waveform generator. The block is made up of three registers: the Operation Condition Register (OCR), the Operation Event Register (OEVR) and the Operation Enable Register (OENR). Refer to the Operation Status Block shown in the figure in section Status Reporting Structure.

When the instrument achieves a certain status, the corresponding bit is set to the OCR. You cannot write to this register. OCR bits that have changed from false (reset) to true (set) status are set in the OEVR. The function of the OENR is to mask the OEVR. You can set this mask and take AND with the OEVR to determine whether or not the OSS bit in the Status Byte Register (SBR) should be set.

As shown in the following figure, a signal is sent to the OEVR (1) when an event occurs. If the corresponding bit in the OENR is also enabled (2), the OSS bit in the SBR is set to one (3).

Operation Event Register
(OEVR)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 TRIG | 4 | 3 | 2 | 1 | 0 CAL |

Read using STATus:OPERation[:EVENt]?
Cannot be written

Operation Enable Register
(OENR)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 TRIG | 4 | 3 | 2 | 1 | 0 CAL |

Read using STATus:OPERation:ENABle?
Write using STATus:OPERation:ENABle

To the OSS bit in the SBR

Questionable Status Block

This block reports on the status of signals and data, such as the accuracy of entered data and signals generated by the instrument. The register configuration and process flow are the same as for the Questionable Status Block.

As shown in the following figure, when an event occurs, a signal is sent to the QEVR (1). If the corresponding bit in the QENR is also enabled (2), the QSS bit in the SBR is set to one (3).

Operation Event Register
(OEVR)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Read using STATus:OPERation[:EVENt]?
Cannot be written

Operation Enable Register
(OENR)

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Read using STATus:OPERation:ENABle?
Write using STATus:OPERation:ENABle

To the OSS bit in the SBR

Standard/Event Status Block

This block is used to report power on/off, command error, and command execution status. The block has two registers: the Standard Event Status Register (SESR) and the Event Status Enable Register (ESER). Refer to the Standard/Event Status Block shown in the figure in section Status Reporting Structure.

The SESR is an eight-bit status register. When an error or other type of event occurs on the instrument, the corresponding bit is set. You cannot write to this register. The ESER is an eight-bit enable register that masks the SESR. You can set this mask, and take AND with the SESR to determine whether or not the ESB bit in the Status Byte Register (SBR) should be set.
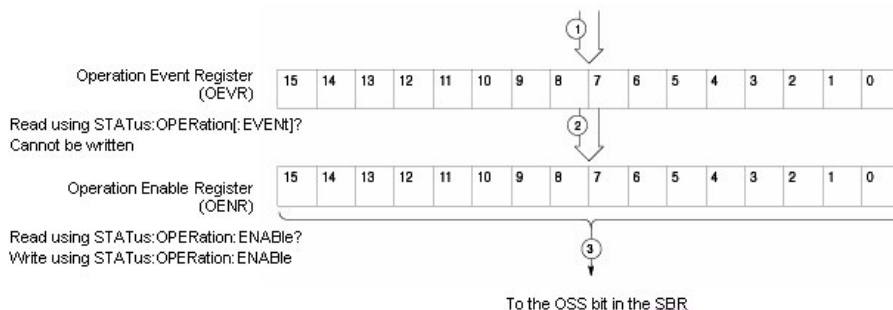
As shown in the following figure, when an event occurs, a signal is sent to the SESR and the event is recorded in the Event Queue (1). If the corresponding bit in the ESER is also enabled (2), the ESB bit in the SBR is set to one (3).

When output is sent to the Output Queue, the MAV bit in the SBR is set to one (4).

When a bit in the SBR is set to one and the corresponding bit in the SRER is enabled (5), the MSS bit in the SBR is set to one and a service request is generated (6).
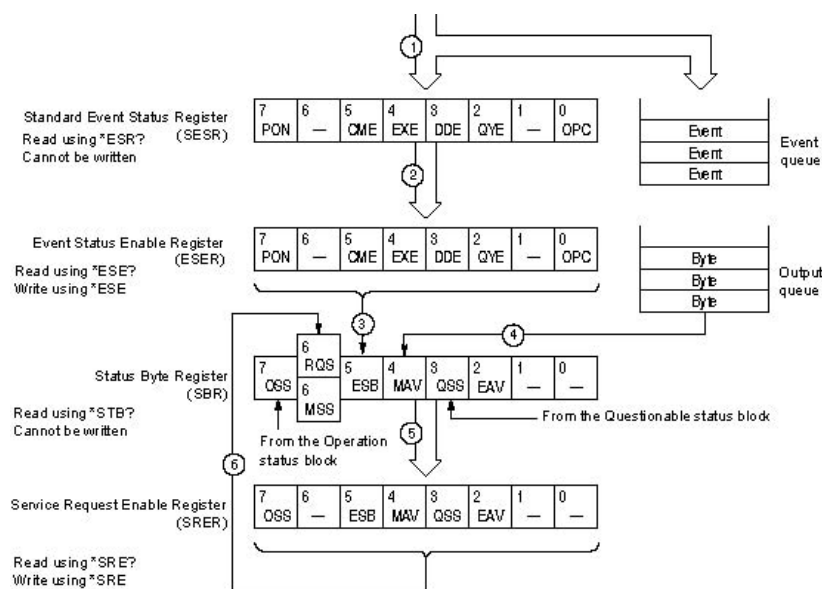
Synchronizing Execution

All commands used in the arbitrary waveform generator are designed to be executed in the order in which they are sent from the external controller. The following synchronization commands are included to ensure compliance with the SCPI standard.

- ■ `*WAI`
- ■ `*OPC`
- ■ `*OPC?`

Messages and Codes

Messages and Codes

Error and event codes with negative values are SCPI standard codes. Error and event codes with positive values are unique to the arbitrary waveform generator series number.

The following table lists event code definitions. When an error occurs, find its error class by checking for the code range in tables that are organized by event class.

Table 3-3: Definition of event codes

Event class	Code range	Description
No error	0	No event or status
Command errors	-100 to -199	Command syntax errors
Execution errors	-200 to -299	Command execution errors
Device-specific errors	-300 to -399	Internal device errors
Query errors	-400 to -499	System event and query errors
Power-on events	-500 to -599	Power-on events
User request events	-600 to -699	User request events
Request control events	-700 to -799	Request control events
Operation complete events	-800 to -899	Operation complete events
Extended device-specific errors	1 to 32767	Device dependent device errors
Reserved	other than those listed above	Not used

Other error messages include:

Table 3-4: Other error codes and messages

Error code range	Operation
3000	Sequence editing
4000	Waveform editing
5000	Sequence/Waveform loading
6000	Other
8000	Hardware

Command Errors

Command errors are returned when there is a syntax error in the command.

Table 3-5: Command errors

Error code	Error message
-100	Command error
-101	Invalid character
-102	Syntax error
-103	Invalid separator
-104	Data type error
-105	GET not allowed
-108	Parameter not allowed
-109	Missing parameter
-110	Command header error
-111	Header separator error
-112	Program mnemonic too long
-113	Undefined error
-114	Header suffix out of range
-115	Unexpected number of parameters
-120	Numeric data error
-121	Invalid character in number
-123	Exponent too large
-124	Too many digits
-128	Numeric data not allowed
-130	Suffix error
-131	Invalid suffix
-134	Suffix too long
-138	Suffix not allowed
-140	Character data error
-141	Invalid character data
-144	Character data too long
-148	Character data not allowed
-150	String data error
-151	Invalid string data
-158	String data not allowed
-160	Block data error
-161	Invalid block data
-168	Block data not allowed
-170	Expression error

Table 3-5: Command errors (cont.)

Error code	Error message
-171	Invalid expression
-178	Expression data not allowed
-180	Macro error
-181	Invalid outside macro definition
-183	Invalid inside macro definition
-184	Macro parameter error

Execution errors

These error codes are returned when an error is detected during command execution.

Table 3-6: Execution errors

Error code	Error message
-200	Execution error
-201	Invalid while in local
-202	Settings lost due to rtl
-203	Command protected
-210	Trigger error
-211	Trigger ignored
-212	Arm ignored
-213	Init ignored
-214	Trigger deadlock
-215	Arm deadlock
-220	Parameter error
-221	Settings conflict
-222	Data out of range
-223	Too much data
-224	Illegal parameter value
-225	Out of memory
-226	Lists not same length
-230	Data corrupt or stale
-231	Data questionable
-232	Invalid format
-233	Invalid version
-240	Hardware error
-241	Hardware missing

Table 3-6: Execution errors (cont.)

Error code	Error message
-250	Mass storage error
-251	Missing mass storage
-252	Missing media
-253	Corrupt media
-254	Media full
-255	Directory full
-256	File name not found
-257	File name error
-258	Media protected
-260	Expression error
-261	Math error in expression
-270	Macro error
-271	Macro syntax error
-272	Macro execution error
-273	Illegal macro label
-274	Macro parameter error
-275	Macro definition too long
-276	Macro recursion error
-277	Macro redefinition not allowed
-278	Macro header not found
-280	Program error
-281	Cannot create program
-282	Illegal program name
-283	Illegal variable name
-284	Program currently running
-285	Program syntax error
-286	Program runtime error
-290	Memory use error
-291	Out of memory
-292	Referenced name does not exist
-293	Referenced name already exists
-294	Incompatible type

Device-specific Errors

These error codes are returned when an internal instrument error is detected. This type of error can indicate a hardware problem.

Table 3-7: Device-specific errors

Error code	Error message
-300	Device-specific error
-310	System error
-311	Memory error
-312	PUD memory lost
-313	Calibration memory lost
-314	Save/recall memory lost
-315	Configuration memory lost
-320	Storage fault
-321	Out of memory
-330	Self-test failed
-340	Calibration failed
-350	Queue overflow
-360	Communication error
-361	Parity error in program message
-362	Framing error in program message
-363	Input buffer overrun
-365	Time out error

Query Errors

Table 3-8: Query errors

Error code	Error message
-400	Query error
-410	Query INTERRUPTED
-420	Query UNTERMINATED
-430	Query DEADLOCKED
-440	Query UNTERMINATED after indefinite response

Power On Event

Table 3-9: Power-On event

Error code	Error message
-500	Power on

User request Event

Table 3-10: User request event

Error code	Error message
-600	User request

Request Control Event

Table 3-11: Request control event

Error code	Error message
-700	Request control

Operation Complete Event

Table 3-12: Operation complete event

Error code	Error message
-800	Operation complete

Appendices

Appendix A: Character Charts

B4 B3 B2 B1	CONTROL (0 0 0)	CONTROL (0 0 1)	NUMBERS SYMBOLS (0 1 0)	NUMBERS SYMBOLS (0 1 1)	UPPER CASE (1 0 0)	UPPER CASE (1 0 1)	LOWER CASE (1 1 0)	LOWER CASE (1 1 1)
0 0 0 0	0 NUL 0 · 0	20 DLE 10 · 16	40 LA0 SP 20 · 32	60 LA16 0 30 · 48	100 TA0 @ 40 · 64	120 TA16 P 50 · 80	140 SA0 ` 60 · 96	160 SA16 p 70 · 112
0 0 0 1	1 GTL SOH 1 · 1	21 LL0 DC1 11 · 17	41 LA1 ! 21 · 33	61 LA17 1 31 · 49	101 TA1 A 41 · 65	121 TA17 Q 51 · 81	141 SA1 a 61 · 97	161 SA17 q 71 · 113
0 0 1 0	2 STX 2 · 2	22 DC2 12 · 18	42 LA2 " 22 · 34	62 LA18 2 32 · 50	102 TA2 B 42 · 66	122 TA18 R 52 · 82	142 SA2 b 62 · 98	162 SA18 r 72 · 114
0 0 1 1	3 ETX 3 · 3	23 DC3 13 · 19	43 LA3 # 23 · 35	63 LA19 3 33 · 51	103 TA3 C 43 · 67	123 TA19 S 53 · 83	143 SA3 c 63 · 99	163 SA19 s 73 · 115
0 1 0 0	4 SDC EOT 4 · 4	24 DCL DC4 14 · 20	44 LA4 $ 24 · 36	64 LA20 4 34 · 52	104 TA4 D 44 · 68	124 TA20 T 54 · 84	144 SA4 d 64 · 100	164 SA20 t 74 · 116
0 1 0 1	5 PPC ENQ 5 · 5	25 PPU NAK 15 · 21	45 LA5 % 25 · 37	65 LA21 5 35 · 53	105 TA5 E 45 · 69	125 TA21 U 55 · 85	145 SA5 e 65 · 101	165 SA21 u 75 · 117
0 1 1 0	6 ACK 6 · 6	26 SYN 16 · 22	46 LA6 & 26 · 38	66 LA22 6 36 · 54	106 TA6 F 46 · 70	126 TA22 V 56 · 86	146 SA6 f 66 · 102	166 SA22 v 76 · 118
0 1 1 1	7 BEL 7 · 7	27 ETB 17 · 23	47 LA7 ' 27 · 39	67 LA23 7 37 · 55	107 TA7 G 47 · 71	127 TA23 W 57 · 87	147 SA7 g 67 · 103	167 SA23 w 77 · 119
1 0 0 0	10 GET BS 8 · 8	30 SPE CAN 18 · 24	50 LA8 (28 · 40	70 LA24 8 38 · 56	110 TA8 H 48 · 72	130 TA24 X 58 · 88	150 SA8 h 68 · 104	170 SA24 x 78 · 120
1 0 0 1	11 TCT HT 9 · 9	31 SPD EM 19 · 25	51 LA9) 29 · 41	71 LA25 9 39 · 57	111 TA9 I 49 · 73	131 TA25 Y 59 · 89	151 SA9 i 69 · 105	171 SA25 y 79 · 121
1 0 1 0	12 LF A · 10	32 SUB 1A · 26	52 LA10 * 2A · 42	72 LA26 : 3A · 58	112 TA10 J 4A · 74	132 TA26 Z 5A · 90	152 SA10 j 6A · 106	172 SA26 z 7A · 122
1 0 1 1	13 VT B · 11	33 ESC 1B · 27	53 LA11 + 2B · 43	73 LA27 ; 3B · 59	113 TA11 K 4B · 75	133 TA27 [5B · 91	153 SA11 k 6B · 107	173 SA27 { 7B · 123
1 1 0 0	14 FF C · 12	34 FS 1C · 28	54 LA12 , 2C · 44	74 LA28 < 3C · 60	114 TA12 L 4C · 76	134 TA28 \ 5C · 92	154 SA12 l 6C · 108	174 SA28 ¦ 7C · 124
1 1 0 1	15 CR D · 13	35 GS 1D · 29	55 LA13 – 2D · 45	75 LA29 = 3D · 61	115 TA13 M 4D · 77	135 TA29] 5D · 93	155 SA13 m 6D · 109	175 SA29 } 7D · 125
1 1 1 0	16 SO E · 14	36 RS 1E · 30	56 LA14 . 2E · 46	76 LA30 > 3E · 62	116 TA14 N 4E · 78	136 TA30 ^ 5E · 94	156 SA14 n 6E · 110	176 SA30 ~ 7E · 126
1 1 1 1	17 SI F · 15	37 US 1F · 31	57 LA15 / 2F · 47	77 UNL ? 3F · 63	117 TA15 O 4F · 79	137 UNT _ 5F · 95	157 SA15 o 6F · 111	177 RUBOUT (DEL) 7F · 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

B7 B6 B5 — BITS B4 B3 B2 B1

KEY

octal → 5
hex → 5
ENQ
PPC ← GPIB code (with ATN asserted)
ENQ ← ASCII character
5 ← decimal

Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: GPIB Interface Specifications

GPIB Interface Specifications

This appendix lists and describes the GPIB functions and messages that the arbitrary waveform generator implements.

Interface Functions

Interface Messages

Interface Functions

The following table lists the GPIB interface functions this instrument implements. Each function is briefly described.

Table B-1: GPIB interface function implementation

Interface function	Implemented subset	Capability	Description
Acceptor Handshake (AH)	AH1	Complete	Enables a listening device to coordinate data reception. The AH function delays data transfer initiation or termination until the listening device is ready to receive the next data byte.
Source Handshake (SH)	SH1	Complete	Enables a talking device to support the coordination of data transfer. The SH function controls the initiation and termination of data byte transfers.
Talker (T)	T6	Basic Talker, Serial Poll Unaddress if my-listen-address (MLA) No Talk Only mode	Enables a device to send device-dependent data over the interface. This capability is available only when the device is addressed to talk, and uses a one-byte address.

Table B-1: GPIB interface function implementation (cont.)

Interface function	Implemented subset	Capability	Description
Listener (L)	L4	Basic Listener Unaddress if my-talk-address (MTA) No Listen Only mode	Enables a device to receive device-dependent data over the interface. This capability is available only when the device is addressed to listen, and uses a one-byte address.
Service Request (SR)	SR1	Complete	Enables a device to request service from the controller.
Remote/Local (RL)	RL1	Complete	Enables a device to select between one of two sources for arbitrary waveform generator control. It determines whether input information is controlled from the front panel (local control) or by GPIB commands (remote control).
Parallel Poll (PL)	PP0	None	-
Device Clear (DC)	DC1	Complete	Enables a device to be cleared or initialized, either individually, or as part of a group of devices.
Device Trigger (DT)	DT1	Complete	-

Table B-1: GPIB interface function implementation (cont.)

Interface function	Implemented subset	Capability	Description
Controller (C)	C0	None	Enables a device that has this capability to send its address, universal commands, and addressed commands to other devices over the interface.
Electrical Interface	E2	Three-state driver	Identifies the electrical interface driver type. The notation E1 means the electrical interface uses open collector drivers, E2 means the electrical interface uses three-state drivers.

Interface Messages

The following table lists the standard interface messages the arbitrary waveform generator supports. Each function is briefly described.

Table B-2: AWG standard interface messages

Message	GPIB	Description
DCL	Yes	Device Clear (DCL). Will clear (initialize) all devices on the bus that have a device clear function, whether or not the controller has addressed them.
GET	Yes	Group Execute Trigger (GET).Triggers all applicable devices and causes them to initiate their programmed actions.
GTL	Yes	Go To Local (GTL). Causes the listen-addressed device to switch from remote to local (front-panel) control.
LLO	Yes	Local Lockout (LLO). Disables the return to local function.

Table B-2: AWG standard interface messages (cont.)

Message	GPIB	Description
PPC	No	Parallel Poll Configure (PPC). Causes the listen-addressed device to respond to the secondary commands Parallel Poll Enable (PPE) and Parallel Poll Disable (PPD), which are placed on the bus following the PPC command. PPE enables a device with parallel poll capability to respond on a particular data line. PPD disables the device from responding to the parallel poll.
PPD	No	
PPE	No	
PPU	No	
SDC	Yes	Select Device Clear (SDC). Clears or initializes all listen-addressed devices.
SPD	Yes	
SPE	Yes	Serial Poll Enable (SPE). Puts all bus devices that have a service request function into the serial poll enabled state. In this state, each device sends the controller its status byte, instead of its normal output, after the device receives its talk address on the data lines. This function may be used to determine which device sent a service request.
TCT	No	Take Control (TCT). Allows the controller in charge to pass control of the bus to another controller on the bus.
UNL	Yes	-
UNT	Yes	-
Listen Addresses	Yes	-
Talk Addresses	Yes	-

Appendix C: SCPI Conformance Information

All commands in the arbitrary waveform generator are based on SCPI Version 1999.0. The following tables list the SCPI commands this arbitrary waveform generator supports.

Group	Command	Defined in SCPI	Not defined SCPI 1999.0
AWGControl			
	AWGControl:APPLication:RUN		√
	AWGControl:APPLication:STATe?		√
	AWGControl:CLOCk:DRATe(?)		√
	AWGControl:CLOCk:PHASe[:ADJust]		√
	AWGControl:CLOCk:SOURce(?)		√
	AWGControl:COMPile		√
	AWGControl:CONFigure:CNUMber?		√
	AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)		√
	AWGControl:DC[n][:STATe](?)		√
	AWGControl:DOUTput[n][:STATe](?)		√
	AWGControl:ENHanced:SEQuence:JMODe		√
	AWGControl:EVENt:DJUMp:DEFine		√
	AWGControl:EVENt:JMODe		√
	AWGControl:EVENt:SOFTware[:IMMediate]		√
	AWGControl:EVENt:TABLe[:IMMediate]		√
	AWGControl:INTerleave:ADJustment:AMPLitude		√
	AWGControl:INTerleave:ADJustment:PHASe		√
	AWGControl:INTerleave:ZERoing(?)		√
	AWGControl:INTerleave[:STATe](?)		√
	AWGControl:RMODe(?)		√
	AWGControl:RRATe(?)		√
	AWGControl:RRATe:HOLD(?)		√
	AWGControl:RSTate?		√
	AWGControl:RUN[:IMMediate]		√
	AWGControl:SEQuencer:POSition?		√
	AWGControl:SEQuencer:TYPE?		√
	AWGControl:SNAMe?		√
	AWGControl:SREStore		√
	AWGControl:SSAVe		√
	AWGControl:STOP[:IMMediate]		√
Calibration			
	*CAL?	√	
	CALibration[:ALL](?)	√	
Diagnostic			
	*TST?	√	
	DIAGnostic:DATA?		√
	DIAGnostic[:IMMediate](?)		√
	DIAGnostic:SELect(?)		√
Display			
	DISPlay[:WINDow[1|2]][:STATe] <state> (?)	√	
Event			
	EVENt[:IMMediate]		√
	EVENt:IMPedance(?)		√
	EVENt:JTIMing (?)		√
	EVENt:LEVel(?)		√
	EVENt:POLarity(?)		√
Instrument			
	INSTrument:COUPle:SOURce(?)	√	
Mass Memory			
	MMEMory:CATalog?	√	
	MMEMory:CDIRectory(?)	√	
	MMEMory:DATA(?)	√	
	MMEMory:DELete	√	
	MMEMory:IMPort		√
	MMEMory:IMPort: PARameter:FREQuency[:UPDate][:STATe](?)		√
	MMEMory:IMPort: PARameter:LEVel [:UPDate]:CHANnel(?)		√
	MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe] (?)		√
	MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE		√
	MMEMory:IMPort:PARameter:NORMalize(?)		√
	MMEMory:IMPort:PARameter:RESampling:FREQuency		√
	MMEMory:IMPort:PARameter:RESampling[:STATe]		√
	MMEMory:MDIRectory		√
	MMEMory:MSIS(?)	√	

Group	Command	Defined in SCPI	Not defined SCPI 1999.0
Output			
	OUTPut[n]:FILTer[:LPASs]:FREQuency(?)	√	
	OUTPut[n][:STATe](?)	√	
Sequence			
	SEQuence:ELEMent[n]:GOTO:INDex(?)		√
	SEQuence:ELEMent[n]:GOTO:STATe(?)		√
	SEQuence:ELEMent[n]:JTARget:INDex(?)		√
	SEQuence:ELEMent[n]:JTARget:TYPE(?)		√
	SEQuence:ELEMent[n]:LOOP:COUNt(?)		√
	SEQuence:ELEMent[n]:LOOP:INFinite(?)		√
	SEQuence:ELEMent[n]:TWAit(?)		√
	SEQuence:ELEMent[n]:WAVeform[n](?)		√
	SEQuence:JUMP[:IMMediate] <target>		√
	SEQuence:LENGth(?)		√
Source			
	[SOURce[1]]:FREQuency[:CW|:FIXed](?)	√	
	[SOURce[1]]:ROSCillator:FREQuency(?)		√
	[SOURce[1]]:ROSCillator:MULTiplier(?)		√
	[SOURce[1]]:ROSCillator:SOURce(?)	√	
	[SOURce[1]]:ROSCillator:TYPE(?)		√
	[SOURce[n]]:DAC:RESolution(?)		√
	[SOURce[n]]:FUNCtion:USER(?)		√
	[SOURce[n]]:COMBine:FEED(?)	√	
	[SOURce[n]]:MARKer[1|2]:DELay(?)		√
	[SOURce[n]]:DELay[:ADJust](?)		√
	[SOURce[n]]:DELay:POINts(?)		√
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)		√
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH(?)		√
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW(?)		√
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet(?)		√
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)		√
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH(?)		√
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW(?)		√
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)		√
	[SOURce[n]]:PDELay:HOLD(?)		√
	[SOURce[n]]:PHASe[:ADJust](?)	√	
	[SOURce[n]]:SKEW(?)		√
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)		√
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	√	
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW(?)	√	
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	√	
	[SOURce[n]]:WAVeform(?)		
Subsequence			
	SEQuence:ELEMent[n]:SUBSequence		√
	SLISt:NAME?		√
	SLISt:SIZE?		√
	SLISt:SUBSequence:DELete		√
	SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt		√
	SLISt:SUBSequence:ELEMent[n]:WAVeform[n]		√
	SLISt:SUBSequence:NEW		√
	SLISt:SUBSequence:LENGth		√
	SLISt:SUBSequence:TSTamp?		√

Group	Command	Defined in SCPI	Not defined SCPI 1999.0
Status			
	*CLS	√	
	*ESE(?)	√	
	*ESR?	√	
	*SRE(?)	√	
	*STB?	√	
	STATus:OPERation:CONDition?	√	
	STATus:OPERation:ENABle(?)	√	
	STATus:OPERation[:EVENt]?	√	
	STATus:PRESet	√	
	STATus:QUEStionable:CONDition?	√	
	STATus:QUEStionable:ENABle(?)	√	
	STATus:QUEStionable[:EVENt]?	√	
Synchronization			
	*OPC(?)	√	
	*WAI	√	
System			
	*IDN?	√	
	*OPT?	√	
	*RST	√	
	SYSTem:DATE(?)	√	
	SYSTem:ERRor[:NEXT]?	√	
	SYSTem:KLOCk(?)	√	
	SYSTem:TIME(?)	√	
	SYSTem:VERSion?	√	
Trigger			
	*TRG	√	
	ABORt	√	
	TRIGger[:SEQuence][:IMMediate]	√	
	TRIGger[:SEQuence]:IMPedance(?)		√
	TRIGger[:SEQuence]:LEVel(?)	√	
	TRIGger[:SEQuence]:MODE		√
	TRIGger[:SEQuence]:POLarity(?)		√
	TRIGger[:SEQuence]:SLOPe(?)	√	
	TRIGger[:SEQuence]:SOURce(?)	√	
	TRIGger[:SEQuence]:TIMer(?)	√	
	TRIGger[:SEQuence]:WVALue (?)		√
Waveform			
	WLISt:NAME?		√
	WLISt:SIZE?		√
	WLISt:WAVeform:DATA(?)		√
	WLISt:WAVeform:DELete		√
	WLISt:WAVeform:LENGth?		√
	WLISt:WAVeform:MARKer:DATA(?)		√
	WLISt:WAVeform:NEW		√
	WLISt:WAVeform:PREDefined?		√
	WLISt:WAVeform:TSTamp?		√
	WLISt:WAVeform:TYPE?		√

Appendix D: Raw Socket Specification

TCP/IP is used as the network protocol, and the port number is variable. Commands can be sent from the application program through the TCP/IP socket interface, and queries can be received through the interface. The following lists the differences between the GPIB interface and the Raw Socket interface.

- The Line Feed (LF) code is needed as a terminator at the end of a message.

- The IEEE 488.1 standard (for example, Device Clear or Service Request) is not supported.

- The Message Exchange Control Protocol in the IEEE 488.2 is not supported. However, common commands such as *ESE and the event handling features are supported.

- The Indefinite format (the block start at #0) in the <ARBITRARY BLOCK PROGRAM DATA> of the IEEE 488.2 is not supported.

AWG5000 and AWG7000 Series Programmer Manual

Appendix E: Factory Initialization Settings

The following tables list the default settings for the each command.

Group	Command	Default setting
AWGControl		
	AWGControl:APPLication:RUN	NA
	AWGControl:APPLication:STATe?	NA
	AWGControl:CLOCk:DRATe(?)	1(AWG7121B/AWG7122B and AWG5012B/AWG5014B) 2 (AWG7061B/AWG7062B and AWG5002B/AWG5004B)
	AWGControl:CLOCk:PHASe[:ADJust]	0 degree
	AWGControl:CLOCk:SOURce(?)	INTernal
	AWGControl:COMPile	NA
	AWGControl:CONFigure:CNUMber?	1,2 or 4
	AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	0 V
	AWGControl:DC[n][:STATe](?)	0
	AWGControl:DOUTput[n][:STATe](?)	0
	AWGControl:ENHanced:SEQuence:JMODe	LOGic
	AWGControl:EVENt:DJUMp:DEFine	
	AWGControl:EVENt:JMODe	EJUMp
	AWGControl:EVENt:SOFTware[:IMMediate]	NA
	AWGControl:EVENt:TABLe[:IMMediate]	NA
	AWGControl:INTerleave:ADJustment:AMPLitude	0 Vpp
	AWGControl:INTerleave:ADJustment:PHASe	0 degree
	AWGControl:INTerleave:ZERoing(?)	0
	AWGControl:INTerleave[:STATe](?)	0
	AWGControl:RMODe(?)	CONTinuous
	AWGControl:RRATe(?)	10 MHz
	AWGControl:RRATe:HOLD(?)	0
	AWGControl:RSTate?	0
	AWGControl:RUN[:IMMediate]	NA
	AWGControl:SEQuencer:POSition?	1
	AWGControl:SEQuencer:TYPE?	HARDware
	AWGControl:SNAMe?	"","C:"
	AWGControl:SREStore	NA
	AWGControl:SSAVe	NA
	AWGControl:STOP[:IMMediate]	NA
Calibration		
	*CAL?	NA
	CALibration[:ALL] (?)	NA
Diagnostic		
	*TST?	NA
	DIAGnositc:DATA?	NA
	DIAGnositc[:IMMediate] (?)	NA
	DIAGnostic:SELect(?)	ALL
Display		
	DISPlay[:WINDow[1|2]][:STATe](?)	0
Event		
	EVENt[:IMMediate]	NA
	EVENt:IMPedance(?)	1e3 ohm
	EVENt:JTIMing (?)	ASYNchronous
	EVENt:LEVel(?)	1.4 V
	EVENt:POLarity(?)	POSitive
Instrument		
	INSTrument:COUPle:SOURce(?)	NONE
Mass Memory		
	MMEMory:CATalog?	NA
	MMEMory:CDIRectory(?)	NA
	MMEMory:DATA(?)	NA
	MMEMory:DELete	NA
	MMEMory:IMPort	NA
	MMEMory:IMPort: PARameter:FREQuency[:UPDate][:STATe](?)	1
	MMEMory:IMPort: PARameter:LEVel [:UPDate]:CHANnel(?)	1
	MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe] (?)	1
	MEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE	IDATa
	MMEMory:IMPort:PARameter:NORMalize(?)	NONE
	MMEMory:IMPort:PARameter:RESampling:FREQuency	Maximum sampling rate for non-interleaved mode
	MMEMory:IMPort:PARameter:RESampling[:STATe]	0
	MMEMory:MDIRectory	NA
	MMEMory:MSIS(?)	"C:"

Group	Command	Default setting
Output		
	OUTPut[n]:FILTer[:LPASs]:FREQuency(?)	9.9e37 (INFinity)
	OUTPut[n][:STATe](?)	0
Sequence		
	SEQuence:ELEMent[n]:GOTO:INDex(?)	1
	SEQuence:ELEMent[n]:GOTO:STATe(?)	0
	SEQuence:ELEMent[n]:JTARget:INDex(?)	1
	SEQuence:ELEMent[n]:JTARget:TYPE(?)	OFF
	SEQuence:ELEMent[n]:LOOP:COUNt(?)	1
	SEQuence:ELEMent[n]:LOOP:INFinite(?)	0
	SEQuence:ELEMent[n]:TWAit(?)	0
	SEQuence:ELEMent[n]:WAVeform[n](?)	""
	SEQuence:JUMP[:IMMediate]	NA
	SEQuence:LENGth(?)	0
Source		
	[SOURce[1]]:FREQuency[:CW | :FIXed](?)	10 GHz
	[SOURce[1]]:ROSCillator:FREQuency(?)	10 MHz
	[SOURce[1]]:ROSCillator:MULTiplier(?)	1
	[SOURce[1]]:ROSCillator:SOURce(?)	INTernal
	[SOURce[1]]:ROSCillator:TYPE(?)	FIXed
	[SOURce[n]]:COMBine:FEED(?)	""
	[SOURce[n]]:DAC:RESolution(?)	8
	[SOURce[n]]:DELay[:ADJust](?)	0 s
	[SOURce[n]]:DELay:POINts(?)	0 points
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	1V (Vpp)
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH(?)	1V
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW(?)	0 V
	[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	0.5 V
	[SOURce[n]]:FUNCtion:USER(?)	"", "C:"
	[SOURce[n]]:MARKer[1|2]:DELay(?)	0
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	1V (Vpp)
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	1.0 V
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW(?)	0 V
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	0.5 V
	[SOURce[n]]:PDELay:HOLD(?)	PHASe
	[SOURce[n]]:PHASe[:ADJust](?)	0 degree
	[SOURce[n]]:SKEW(?)	0 s
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	1V (Vpp)
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	0.5 V
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW(?)	- 0.5 V
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	0 V
	[SOURce[n]]:WAVeform(?)	""
Subsequence		
	SEQuence:ELEMent[n]:SUBSequence	NA
	SLISt:NAME?	NA
	SLISt:SIZE?	NA
	SLISt:SUBSequence:DELete	NA
	SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt	NA
	SLISt:SUBSequence:ELEMent[n]:WAVeform[n]	NA
	SLISt:SUBSequence:NEW	NA
	SLISt:SUBSequence:LENGth	NA
	SLISt:SUBSequence:TSTamp?	NA

Group	Command	Default setting
Synchronization		
	*OPC(?)	NA
	*WAI	NA
Trigger		
	*TRG	NA
	ABORt	NA
	TRIGger[:SEQuence][:IMMediate]	NA
	TRIGger[:SEQuence]:IMPedance(?)	1KOhm
	TRIGger[:SEQuence]:LEVel(?)	1.4 V
	TRIGger[:SEQuence]:MODE	ASYNchronous
	TRIGger[:SEQuence]:POLarity(?)	POSitive
	TRIGger[:SEQuence]:SLOPe(?)	POSitive
	TRIGger[:SEQuence]:SOURce(?)	EXTernal
	TRIGger[:SEQuence]:TIMer(?)	100 ms
	TRIGger[:SEQuence]:WVALue(?)	FIRSt
Waveform		
	WLISt:NAME?	NA
	WLISt:SIZE?	20
	WLISt:WAVeform:DATA(?)	NA
	WLISt:WAVeform:DELete	NA
	WLISt:WAVeform:LENGth?	NA
	WLISt:WAVeform:MARKer:DATA(?)	NA
	WLISt:WAVeform:NEW	NA
	WLISt:WAVeform:PREDefined?	NA
	WLISt:WAVeform:TSTamp?	NA
	WLISt:WAVeform:TYPE?	NA

Appendix F: Compatibility with Other Instruments

The following tables list the compatibility of the commands with other Tektronix arbitrary waveform generators like the AWG400, AWG500, AWG600, and AWG700 Series.

AWG5000/AWG7000 Series Command Group	AWG400	AWG500	AWG600/700	Note
AWGControl				
AWGControl:APPLication:RUN				
AWGControl:APPLication:STATe?				
AWGControl:CLOCk:DRATe(?)				
AWGControl:CLOCk:PHASe[:ADJust]				
AWGControl:CLOCk:SOURce(?)	✓	✓		AWG400 / AWG500 only
AWGControl:COMPile				
AWGControl:CONFigure:CNUMber?				
AWGControl:DC[n][:STATe](?)				
AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)				
AWGControl:DOUTput[n][:STATe](?)	✓	✓	✓	
AWGControl:ENHanced:SEQuence:JMODe	✓	✓	✓	
AWGControl:EVENt:DJUMp:DEFine				
AWGControl:EVENt:JMODe				
AWGControl:EVENt:SOFTware[:IMMediate]	✓	✓	✓	
AWGControl:EVENt:TABLe[:IMMediate]	✓	✓	✓	
AWGControl:INTerleave:ADJustment:AMPLitude				
AWGControl:INTerleave:ADJustment:PHASe				
AWGControl:INTerleave[:STATe](?)				
AWGControl:INTerleave:ZERoing(?)				
AWGControl:RMODe(?)		See Note		SEQuence, instead of ENHanced
AWGControl:RRATe(?)				
AWGControl:RRATe:HOLD(?)				
AWGControl:RSTate?	✓	✓	✓	
AWGControl:RUN[:IMMediate]	✓	✓	✓	
AWGControl:SEQuencer:POSition?				
AWGControl:SEQuencer:TYPE?				
AWGControl:SNAMe?				
AWGControl:SREStore	✓	✓	✓	
AWGControl:SSAVe	✓	✓	✓	
AWGControl:STOP[:IMMediate]	✓	✓	✓	
Calibration				
*CAL?	✓	✓	✓	
CALibration[:ALL] (?)	✓	✓	✓	
Diagnostic				
*TST?	✓	✓	✓	
DIAGnositc:DATA?	✓	✓	✓	
DIAGnositc[:IMMediate] (?)	✓	✓	✓	
DIAGnostic:SELect(?)		See Note		Arguments are different, except for ALL
Display				
DISPlay[:WINDow[1|2]][:STATe](?)				
Event				
EVENt[:IMMediate]				
EVENt:IMPedance(?)				
EVENt:JTIMing (?)				
EVENt:LEVel(?)				
EVENt:POLarity(?)				
Instrument				
INSTrument:COUPle:SOURce(?)				
Mass Memory				
MMEMory:CATalog?	✓	✓	✓	
MMEMory:CDIRectory(?)	✓	✓	✓	
MMEMory:DATA(?)	✓	✓	✓	
MMEMory:DELete	✓	✓	✓	
MMEMory:IMPort				
MMEMory:IMPort: PARameter:FREQuency[:UPDate][:STATe](?)				
MMEMory:IMPort: PARameter:LEVel [:UPDate]:CHANnel(?)				
MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe] (?)				
MMEMory:IMPort:PARameter:LEVel[:UPDate]:TYPE				
MMEMory:IMPort:PARameter:NORMalize(?)				
MMEMory:IMPort:PARameter:RESampling:FREQuency				
MMEMory:IMPort:PARameter:RESampling[:STATe]				
MMEMory:MDIRectory	✓	✓	✓	
MMEMory:MSIS(?)		See Note		Drive letter, instead of "MAIN" or "FLOPpy"

AWG5000/AWG7000 Series Command Group	AWG400	AWG500	AWG600/700	Note
Output				
OUTPut[n]:FILTer[:LPASs]:FREQuency(?)		See Note		Valid values depending on models
OUTPut[n][:STATe](?)	✓	✓	✓	
Sequence				
SEQuence:ELEMent[n]:GOTO:INDex(?)				
SEQuence:ELEMent[n]:GOTO:STATe(?)				
SEQuence:ELEMent[n]:JTARget:INDex(?)				
SEQuence:ELEMent[n]:JTARget:TYPE(?)				
SEQuence:ELEMent[n]:LOOP:COUNt(?)				
SEQuence:ELEMent[n]:LOOP:INFinite(?)				
SEQuence:ELEMent[n]:TWAit(?)				
SEQuence:ELEMent[n]:WAVeform[n](?)				
SEQuence:JUMP[:IMMediate] <target>				
SEQuence:LENGth(?)				
Source				
[SOURce[n]]:COMBine:FEED(?)	✓	✓		Parameter varies depending on models
[SOURce[1]]:FREQuency[:CW|:FIXed](?)	✓	✓	✓	
[SOURce[1]]:ROSCillator:FREQuency(?)				
[SOURce[1]]:ROSCillator:MULTiplier(?)				
[SOURce[1]]:ROSCillator:SOURce(?)	✓	✓	✓	
[SOURce[1]]:ROSCillator:TYPE(?)				
[SOURce[n]]:DAC:RESolution(?)				
[SOURce[n]]:DELay[:ADJust](?)				
[SOURce[n]]:DELay:POINts(?)				
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:HIGH(?)				
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:LOW(?)				
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)				
[SOURce[n]]:DIGital:VOLTage[:LEVel][:IMMediate]:OFFSet(?)				
[SOURce[n]]:ENABle(?)				
[SOURce[n]]:FUNCtion:USER(?)	✓	✓	✓	
[SOURce[n]]:MARKer[1|2]:DELay(?)	✓			AWG400 only
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)				
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	✓			AWG400 only
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW(?)	✓			AWG400 only
[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)				
[SOURce[n]]:PDELay:HOLD(?)				
[SOURce[n]]:PHASe[:ADJust](?)				
[SOURce[n]]:SKEW(?)	✓	✓	✓	
[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	✓	✓	✓	
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	✓	✓	✓	
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW(?)	✓	✓	✓	
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)	✓	✓	✓	
[SOURce[n]]:WAVeform(?)				
Subsequence				
SEQuence:ELEMent[n]:SUBSequence				
SLISt:NAME?				
SLISt:SIZE?				
SLISt:SUBSequence:DELete				
SLISt:SUBSequence:ELEMent[n]:LOOP:COUNt				
SLISt:SUBSequence:ELEMent[n]:WAVeform[n]				
SLISt:SUBSequence:NEW				
SLISt:SUBSequence:LENGth				
SLISt:SUBSequence:TSTamp?				

AWG5000/AWG7000 Series Command Group	AWG400	AWG500	AWG600/700	Note
Status				
*CLS	✓	✓	✓	
*ESE(?)	✓	✓	✓	
*ESR?	✓	✓	✓	
*SRE(?)	✓	✓	✓	
*STB?	✓	✓	✓	
STATus:OPERation:CONDition?	✓	✓	✓	
STATus:OPERation:ENABle(?)	✓	✓	✓	
STATus:OPERation[:EVENt]?	✓	✓	✓	
STATus:PRESet	✓	✓	✓	
STATus:QUEStionable:CONDition?	✓	✓	✓	
STATus:QUEStionable:ENABle(?)	✓	✓	✓	
STATus:QUEStionable[:EVENt]?	✓	✓	✓	
Synchronization				
*OPC(?)	✓	✓	✓	
*WAI	✓	✓	✓	
System				
*IDN?	✓	✓	✓	
*OPT?	✓	✓	✓	
*RST	✓	✓	✓	
SYSTem:DATE(?)	✓	✓	✓	
SYSTem:ERRor[:NEXT]?	✓	✓	✓	
SYSTem:KLOCk(?)	✓	✓	✓	
SYSTem:TIME(?)	✓	✓	✓	
SYSTem:VERSion?	✓	✓	✓	
Trigger				
*TRG	✓	✓	✓	
ABORt	✓	✓	✓	
TRIGger[:SEQuence][:IMMediate]	✓	✓	✓	
TRIGger[:SEQuence]:IMPedance(?)	✓	✓	✓	
TRIGger[:SEQuence]:LEVel(?)	✓	✓	✓	
TRIGger[:SEQuence]:MODE				
TRIGger[:SEQuence]:POLarity(?)	✓	✓	✓	
TRIGger[:SEQuence]:SLOPe(?)	✓	✓	✓	
TRIGger[:SEQuence]:SOURce(?)	✓	✓	✓	
TRIGger[:SEQuence]:TIMer(?)	✓	✓	✓	
TRIGger[:SEQuence]:WVALue (?)				
Waveform				
WLISt:NAME?				
WLISt:SIZE?				
WLISt:WAVeform:DATA(?)				
WLISt:WAVeform:DELete				
WLISt:WAVeform:LENGth?				
WLISt:WAVeform:MARKer:DATA(?)				
WLISt:WAVeform:NEW				
WLISt:WAVeform:PREDefined?				
WLISt:WAVeform:TSTamp?				
WLISt:WAVeform:TYPE?				

Index