Programmer Manual

Tektronix

AWG7000 Arbitrary Waveform Generator

077-0028-00

Adapted from Arbitrary Waveform Generator Programmer Online Help

www.tektronix.com

Contacting Tektronix

Tektronix, Inc.
14200 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.

Table of Contents

List of Figures

List of Tables

Getting Started

This online programmer guide provides you with the information you need to use commands for remotely controlling your instrument. With this information, you can 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. You can 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.
- Miscellaneous. 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 AWG7000 Series supports GPIB interface and LAN interface. To set the GPIB address, you can use the System Menu > GPIB/LAN Configuration menu.

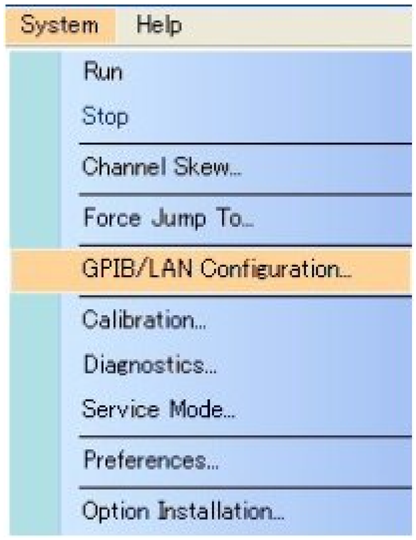


Figure 1: System menu

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, you can use the computer to remotely control your AWG7000 Series. With the AWG7000 Series, you can use the GPIB interface as a controller. See the GPIB Parameters section for information on GPIB parameters.

LAN Interface

The AWG7000 Series accepts two types of Ethernet LAN connections; one is simple ("Raw Socket") connection, and the other is VXI-11 protocol. See the LAN Parameters section for information on LAN parameters.

GPIB Parameters

To use the GPIB, the AWG7000 Series requires you to configure the GPIB mode and the GPIB address.

- Talk/Listen: Select this mode to remotely control your AWG7000 Series using an external computer as the controller.
- Off Bus: Select this mode to electronically disconnect the AWG7000 Series from the GPIB bus.
- Address: This address is a number that allows the software to identify each device connected to the GPIB bus. You must specify a unique number from 0 to 30 for each device.

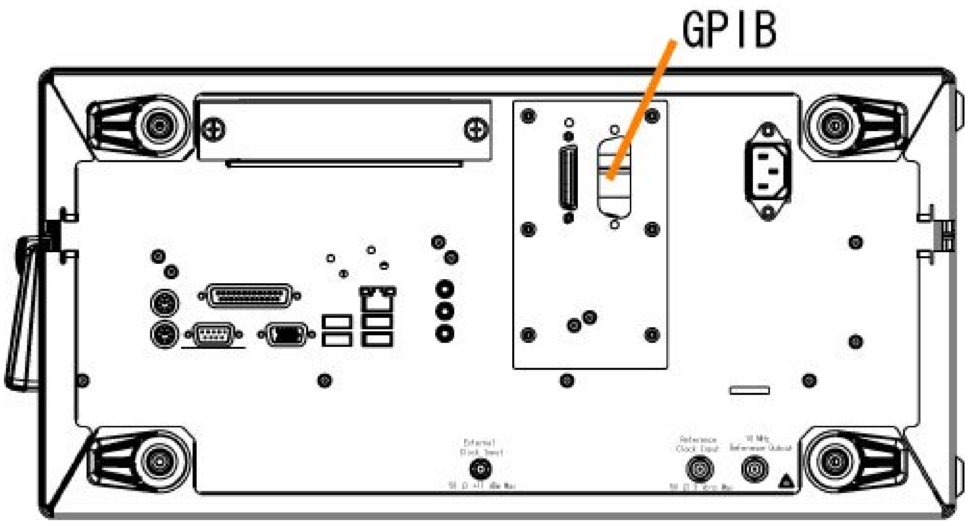


Figure 2: Rear panel of the instrument showing the GPIB port

LAN Parameters

In the AWG7000 Series, 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 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. You can 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 AWG7000 Series is specified to automatically acquire an IP address by DHCP. Refer to Windows documentation regarding network-related parameters. For TekVISA, refer to the TekVISA manual.

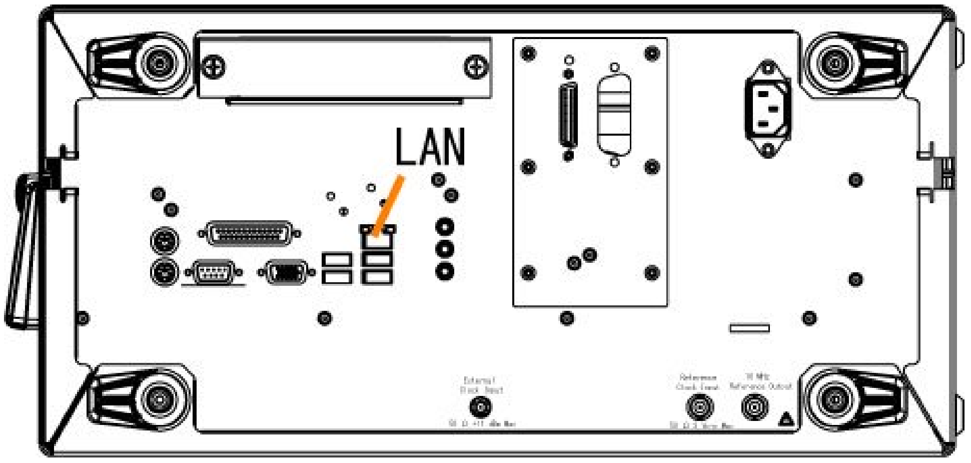


Figure 3: Rear panel of the instrument showing the LAN port

Connecting to the Instrument

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.

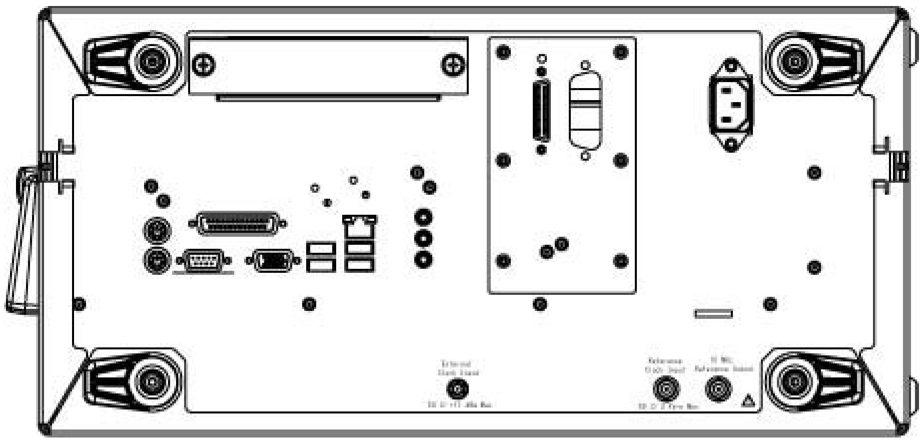


Figure 4: Rear panel of the instrument

Setting Up Remote Communications

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.

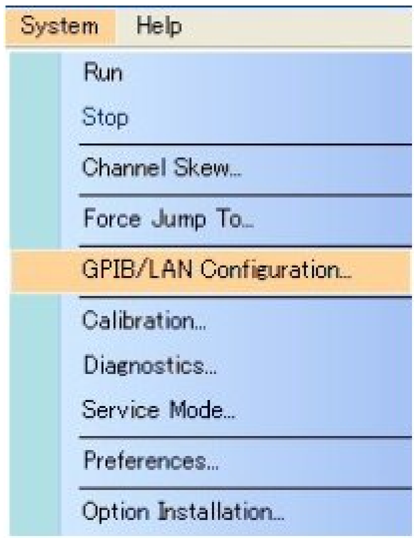
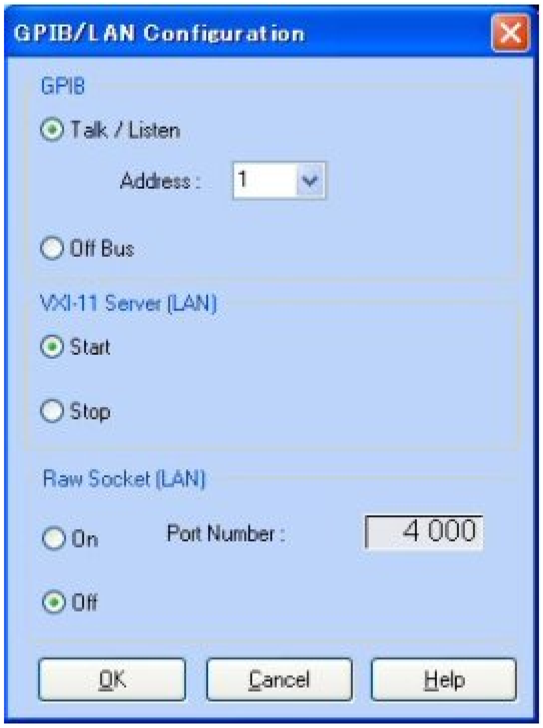


Figure 5: System menu

2. The GPIB/LAN Configuration dialog box is displayed.



GPIB: [✓ Talk / Listen] Off Bus
Address : 1
VXI-11 Server (LAN): [✓ Start] Stop
Raw Socket (LAN): On [✓ Off]
Port Number : 4 000

Figure 6: GPIB/LAN configuration screen

3. Change the GPIB Address to a unique address.

4. Click the OK button.

Documentation

In addition to this AWG7000 Series Programmer Online Guide, the following documentation is included with this instrument:

- AWG7000 Series Arbitrary Waveform Generators Quick Start User Manual. The Quick Start User Manual has information about installing and operating your instrument.
- AWG7000 Series Arbitrary Waveform Generators User Online Help. The User Online Help system is integrated with the User Interface application that ships with this product. The online help provides in-depth operation and user interface help.
- AWG7000 Series Arbitrary Waveform Generators Specifications and Performance Verification Technical Reference Manual (071-1853-xx). The technical reference manual is a PDF only manual; it includes both the specifications and the performance verification procedure.
- Optional AWG7000 Series Arbitrary Waveform Generators Service Manual (071-1854-xx). A service manual is available as an optional accessory; it includes procedures to service the instrument to the module level.

Command Syntax

Syntax Overview

You can 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 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: 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

You can clear the Output Queue and reset the instrument to accept a new command or query by using the Device Clear (DCL) or Selected Device Clear (SDC) GPIB functions. 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, you can enter the command MMEMory:CATalog simply as MMEM:CAT.

Concatenating

You can concatenate any combination of set commands and queries using a semicolon (;). The instrument executes concatenated commands in the order received. When concatenating commands and queries, you must 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, you can abbreviate the second command and eliminate the beginning colon. For example, you can 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 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 AWG7000 series command set and some are defined by SCPI.

Table 4: Parameter types, their descriptions, and examples

Parameter type	Description	Example
Arbitrary block	A block of data bytes	#21012234567890
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, NR2n NR3 examples in this table
String	Alphanumeric characters (must be within quotation marks)	"Testing 1, 2, 3"

About MIN, MAX

You can use MINimum and MAXimum keywords in addition to Numeric in the commands with the “Numeric” parameter. You can set the minimum value or the maximum value by using these keywords. You can query the minimum value or the maximum value at that time.

Block

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

Table 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)
<Block>	A block of data bytes defined as: <Block> ::={#<NZDig><Dig>[<Dig>...][<DChar>...] |#0[<DChar>...]<terminator>}

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

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 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, you can 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, you can 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.

You can omit the unit when you describe commands, but you must include the SI unit prefix. You can 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 (UI_{rms}).
- SPP for Peak-to-Peak, Unit is second (s_{p-p}).
- SRMS for RMS, Unit is second (s_{rms}).
- V/NS for SLEW's unit (V/ns).

In the case of angles, you can 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 7: SI prefixes and their indexes

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

* 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.

** 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.

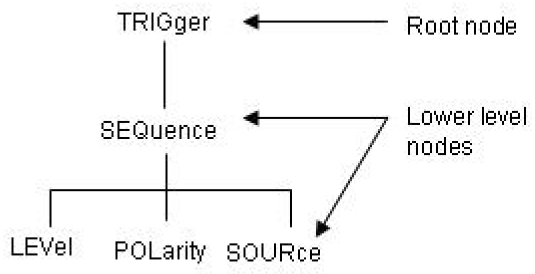
correct: 10MHz, 10E+6Hz, 10E+6

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

You can use the following commands to control operating modes:

Table 8: Control group commands and their descriptions

Command	Description
AWGControl:CLOCk:DRATe(?)	Sets or returns the divider rate for the external oscillator
AWGControl:CLOCk:SOURce(?)	Sets or returns the clock source
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 offset
AWGControl:DOUTput[n][:STATe](?)	Outputs the raw waveform in the DAC of the specified channel
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
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

You can use the following calibration commands to calibrate the arbitrary waveform generator:

Table 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

You can use the following diagnostic commands to control self-test diagnostic routines:

Table 10: Diagnostic group commands and their descriptions

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

Display Group Commands

You can use the following display commands to set the display state of waveform and sequence windows on the AWG7000 series instrument:

Table 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

You can use the following event commands to configure external event input and generate an event:

Table 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

You can use the following instrument commands to set or return the coupled state of instrument models:

Table 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

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

Table 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 the waveform file 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:NORMalize(?)	Sets or queries whether waveform data are to be normalized
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

You can use the following output commands to set or return the characteristics of the output port of the arbitrary waveform generator:

Table 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

You can use the following sequence commands to define and edit a sequence:

Table 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[n](?)	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 in Table 17 provides ways to create and edit the waveform sequences in AWG7000 Series 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. You can 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 Series Arbitrary Waveform Generators User Manual for a detailed 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 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[n](?)
CH 2 Waveform	“”	SEQuence:ELEMent[n]:WAVeform[n](?)
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

You can use the following source commands to set and query the waveform or marker output parameter:

Table 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
[SOURce[n]]:DAC:RESolution(?)	Sets or returns the DAC resolution
[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	Sets or returns the amplitude of digital output
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH(?)	Sets or returns the high digital output
[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW(?)	Sets or returns the low digital output
[SOURce[n]]: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]]:SKEW(?)	Sets or returns the skew for the waveform associated with a channel
[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 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)
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

Synchronization Group Commands

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

Table 20: 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

You can use the following system commands to control miscellaneous instrument functions:

Table 21: 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

You can use the following trigger commands synchronize the arbitrary waveform generator actions with events:

Table 22: Trigger group commands and their descriptions

Command	Description
*TRG	Generates a trigger event
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]: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
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

You can use the following waveform commands to create and transfer waveforms between the instrument and the external controller:

Table 23: 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:PREDefined?	True or false based on whether the waveform is predefined
WLISt:WAVeform:TSTamp?	Returns the time stamp of the waveform
WLISt:WAVeform:TYPE?	Returns the type of the waveform

Waveform Data Format

The AWG7000 Series supports 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 figure. 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.

	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				

Figure 7: Integer data format

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.

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 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 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[n](?) commands to understand the waveform sizes that are allowed in each case.

Syntax and Commands

*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?
Returns	<NR1> 0 indicates no error.
Related Commands	CALibration[:ALL]?
Examples	*CAL? 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.

*ESE(?)

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

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

*IDN? (Query only)

This command returns identification information for the arbitrary waveform generator.

Group	System
Syntax	*IDN?
Returns	<Manufacturer>, <model>, <serial number>, <Firmware version> <Manufacturer>:: = TEKTRONIX <Model>:: = AWG7101, AWG7102, AWG7051, AWG7052 <Serial number>:: = XXXXXXX (indicates an actual serial number) <Firmware level>:: = SCPI: 99.0 FW:x.x.x (x.x.x is system software version)
Examples	*IDN? Might return the following response: TEKTRONIX,AWG7102,B010123,SCPI:99.0 FW:1.0

*TST? (Query only)

This query executes a self test and returns the results.

Group	Diagnostic
Syntax	*TST?
Returns	<NR1> 0 indicates no error.
Related Commands	DIAGnostic[:IMMediate] DIAGnostic:DATA? DIAGnostic[:SELect]?
Examples	*TST? Might return -330 indicating that the self test failed.

*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?
Returns	<NR1> <NR1>=1 when all pending operations are finished.
Related Commands	*WAI
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?
Returns	<opt> [, <opt> [, <opt>]]] <opt>::= { 0 | 01 | 02 | 03 | 05 | 06 }
Related Commands	None
Examples	*OPT? Might return 0 to indicate that no option is installed in the instrument.

*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.

*SRE(?)

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

Group	Status
Syntax	*SRE <NR1> *SRE?
Arguments	<NR1>
Returns	<NR1>
Related Commands	*CLS *ESE *ESR? *STB?
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.

*STB? (Query only)

This query returns the contents of Status Byte Register.

Group	Status
Syntax	*STB?
Returns	<NR1>
Related Commands	*CLS *ESE *ESR? *SRE
Examples	*STB? Might return 96, which indicates that the SBR contains the binary number 0110 0000.

*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.

*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.

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:CLOCk:DRATe(?)

This command 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.

Group	Control
Syntax	AWGControl:CLOCk:DRATe <divider_rate>
	AWGControl:CLOCk:DRATe?
Arguments	<divider_rate>::=<NR1>
	At *RST, this returns the minimum value.
Returns	<NR1>
Related Commands	AWGControl:CLOCk:SOURce?
Examples	AWGControl:DRATe 8
	Sets the divider rate to 8.
	AWGControl:DRATe ?
	Returns 8.

AWGControl:CLOCk:SOURce(?)

This command 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?

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

Related Commands

AWGControl:CLOCk:DRATe?

Examples

AWGControl:CLOCk:SOURce EXTernal
Sets the clock source to EXTernal.

AWGControl:CLOCk:SOURce?
Returns EXT.

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?
Returns	<NR1> Returns 1 or 2 depending on the model.
Related Commands	None
Examples	AWGControl:CONFigure:CNUMber? Might return 2.

AWGControl:DC[n][:STATe](?)

This commands sets or returns the output state of the 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?
Arguments	<state>::= <Boolean> 0 indicates False 1 indicates True At *RST, this returns False.
Returns	<state>
Related Commands	AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet?
Examples	AWGControl:DC1:STATe 1 Sets the DC1 output to On.

AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)

This command 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?

Arguments

<offset>::=<NR3>
The value will be between –3.0 V to +5.0 V.
At *RST, this returns 0 V.

Returns

<NR3>

Related Commands

AWGControl:DC[n][:STATe](?)

Examples

AWGControl:DC1:VOLTage:OFFSet 1.0V
Sets the DC1 level to 1.0 V.

AWGControl:DC1:VOLTage:OFFSet?
Will return 1.0 V.

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]?
Arguments	<state>::= <Boolean> 0 indicates False 1 indicates True At *RST, this returns False.
Returns	<state>
Related Commands	SOURce[n]:VOLTage:LEVel:IMMEdiate:OFFSet OUTPut[n]:FILTer[:LPASs]:FREQuency
Examples	AWGControl:DOUTput1:STATe 1 Causes the instrument to output raw DAC waveform from Channel 1.

AWGControl:INTerleave[:STATe](?)

This command enables or disables the interleave state for channels. This is available only on the AWG7102 with Option 06. The query form of this command returns the interleave state of the instrument.

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 that:

- 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]?
Arguments	<state>::=<Boolean> 0 indicates False 1 indicates True At *RST, this returns False.
Returns	<state>
Related Commands	AWGControl:INTerleave:ZERoing?
Examples	AWGControl:INTerleave:STATe 1 Sets the instrument to interleave mode.

AWGControl:INTerleave:ZERoing(?)

This command turns the zeroing on and off for the interleave mode.

Note that:

- This command is available only on the AWG7102 with Option 06.
- 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?
Arguments	<state>::=<Boolean> 0 indicates False 1 indicates True At *RST, this returns False.
Returns	<state>
Related Commands	AWGControl:INTerleave[:STATe]?
Examples	AWGControl:INTerleave:ZERoing Turns on the zeroing function..

AWGControl:RMODe(?)

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

Group

Control

Syntax

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

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.
At *RST, this value is CONTinuous.

Returns

CONT | TRIG | GAT | SEQ

Related Commands

AWGControl:RUN
AWGControl:STOP
*TRG
SOURce[n]:FUNction:USER

Examples

AWGControl:RMODe?
Returns CONT if the instrument is in continuous mode.

AWGControl:RMODe TRIGgered
Sets the instrument Run mode to Triggered.

The following table lists the run modes and their descriptions:

Table 24: Mode options 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 sets or returns the repetition rate of the arbitrary waveform generator.

Group	Control
Syntax	AWGControl:RRATe <repetition rate> AWGControl:RRATe?
Arguments	<repetition rate>::=<NR3>
Returns	<NR3>
Related Commands	SOURce[n]:FREQuency
Examples	AWGControl:RRATe 1000000 Sets the repetition rate to 1MHz. AWGControl:RRATe? Returns 1E+6.

AWGControl:RRATe:HOLD(?)

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

Group	Control
Syntax	AWGControl:RRATe:HOLD <hold_state> AWGControl:RRATe:HOLD?
Arguments	<hold_state>::=<Boolean> 0 indicates False 1 indicates True At *RST, this returns 0 (False).
Returns	<NR1>
Related Commands	AWGControl RRATe?
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?
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
Related Commands	AWGControl:RMODe AWGControl:RUN
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/Delete/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?
Returns	<NR1> At *RST, this value is –1 (according to the ISDB definition 3.6.6).
Related Commands	AWGControl:SEQuencer:TYPE?
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?
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.
Related Commands	None
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?
Returns	<file_name>,<msus> <file_name>::=<string> <msus> (mass storage unit specifier)::=<string> At *RST, this values is “”,“C:”
Related Commands	AWGControl:SSAVe AWGControl:SREStore
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>]
Arguments	<file_name>::=<string> <msus> (mass storage unit specifier)::=<string>
Related Commands	AWGControl:SNAMe AWGControl:SSAVe
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>]
Arguments	<file_name>::=<string> <msus> (mass storage unit specifier)::=<string>
Related Commands	AWGConrol:SREStore AWGControl:SNAMe
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
Examples	AWGControl:STOP[:IMMediate] Stops the output of a waveform.

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]?
Returns	<calibration error code> ::= <NR1> 0 indicates no error –340: error
Related Commands	*CAL
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.

DIAGnositc:DATA? (Query only)

This command returns the results of a self test.

Group
Diagnostic

Syntax
DIAGnostic:DATA?

Returns
<NR1>
0 indicates no error.
–330 indicates that the self test failed.

Related Commands
DIAGnostic:IMMediate?
DIAGnostic:SELect?

Examples
DIAGnostic:DATA?
Might return 0, which indicates that the diagnostics completed without any errors.

DIAGnositc[:IMMediate](?)

The DIAGnostic[:IMMediate] command executes the selected self-test routines. The DIAGnostic[:IMMediate]? command executes the selected self-test routines and returns the results

Group	Diagnostic
Syntax	DIAGnostic[:IMMediate] DIAGnostic[:IMMediate]?
Returns	<NR1> 0 indicates no error. –330 indicates that the self test failed.
Related Commands	DIAGnostic:DATA? DIAGnostic:SELect?
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
- AREGister – AWG register read back
- A1Memory – CH1 waveform memory test
- A2Memory – CH2 waveform memory test
- CREGister – Clock register read back
- CPLock – PLL Lock/unlock
- O1Register – Output1 register read back
- O1ALevel – Output1 analog level
- O1MLevel – Output1 Marker level
- O2Register – Output2 register read back
- O2ALevel – Output2 analog level
- O2MLevel – Output2 marker level

Group	Diagnostic
Syntax	DIAGnostic:SELect { ALL | FPANel | AREGister | A1Memory | A2Memory | CREGister | CPLock | O1Register | O1ALevel | O1MLevel | O2Register | O2ALevel | O2MLevel } DIAGnosctic:SELect?
Returns	ALL | FPAN | AREG | A1M | A2M | CREG | CPL | O1R | O1AL | O1ML | O2R | O2AL | O2ML
Related Commands	DIAGnostic[:IMMediate]?
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]?
Arguments	<display_state>::={ON | OFF|<NR1>} OFF or <NR1>=0 minimizes the window display. ON or <NR1>≠0 restores the window display. At *RST, this value is 1.
Returns	<NR1>
Examples	DISPlay:WINDow1:STATe 0 Minimizes the sequence window.

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]:JTYPe(?)).

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 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?
Arguments	<ohms>::=<NR3> Valid values are 50 ohm or 1 kohm. At *RST, this value is 1e3 ohm.
Returns	<NR3>
Related Commands	EVENt[:IMMediate] EVENt:JTIMing? EVENt:LEVel? EVENt:POLarity?
Examples	EVENt:IMPedance 50 Sets the impedance to 50 ohms.

EVENt:JTIMing(?)

This command 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?
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
Related Commands	EVENt[:IMMediate] EVENt:IMPedance? EVENt:LEVel? EVENt:POLarity?
Examples	EVENT:JTIMing ASYNchrounous Sets the jump to asynchronous type.

EVENt:LEVel(?)

This command sets or returns the event level.

Group	Event
Syntax	EVENt:LEVel <level> EVENt:LEVel?
Arguments	<level>::=<NR3> Range is between 5 V and –5 V. At *RST, this returns 1.4 V
Returns	<NR3>
Related Commands	EVENt[:IMMediate] EVENt:IMPedance? EVENt:JTIMing? EVENt:POLarity?
Examples	EVENt:LEVel 1.0V Sets the level to 1 volt.

EVENt:POLarity(?)

This command 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?
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
Related Commands	EVENt[:IMMediate] EVENt:IMPedance? EVENt:JTIMing? EVENt:LEVel?
Examples	EVENt:POLarity NEGative Sets the event polarity to negative.

INSTrument:COUPle:SOURce(?)

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

Note that:

- When coupling is done, CH1 parameters are copied to CH2 parameters. This cannot be changed.
- When ALL is used, all other channels get the parameters of CH1.
- On two channel models, ALL is equivalent to PAIR.
- On one channel models, only NONE is available.
- 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?

Arguments

<state>::={NONE | PAIR | ALL}
NONE
PAIR – CH1 to CH2
ALL – CH1 to CH2

Returns <state>

Related Commands None

Examples

INSTrument:COUPle:SOURce ALL
Copies the channel 1 parameter values to channel 2.

MMEMory:CATalog? (Query only)

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

Group	Mass memory
Syntax	MMEMory:CATalog? [<msus>]
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.
Related Commands	MEMory:CDIRectory MMEMory:MSIS
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 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?
Arguments	<directory_name>::=<string>
Returns	<directory_name>
Related Commands	None
Examples	MMEMory:CDIRectory "/AWG/WORK0" Changes the current directory to /AWG/WORK0.

MMEMory:DATA(?)

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

Note that:

- 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.

Group	Mass memory
Syntax	MMEMory:DATA <file_name>,<block_data> MMEMory:DATA? <file_name>
Arguments	<file_name>,<block_data>
Returns	Block_data – IEEE 488.2 data block file_name – string having filename and path.
Related Commands	MMEMory:CDIRectory MMEMory:MSIS
Examples	MMEMory:DATA "FILE1",#41024xxxxx... Loads data into the file FILE1.

MMEMory:DELete (No query form)

This command deletes the waveform file 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>]
Arguments	<file_name>::=<string> <msus> (mass storage unit specifier)::=<string>
Related Commands	MMEMory:CDIRectory MMEMory:MSIS
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:IMPort (No query form)

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

Note that:

- 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:

- TDS – TDS5000/TDS6000/TDS7000 Series waveform
- TXT – Text file with analog data
- TXT8 – Text file with 8-bit DAC resolution
- TXT10 – Text file with 10-bit DAC resolution
- WFM – AWG400/AWG500/AWG600/AWG700 Series waveform
- PAT – AWG400/AWG500/AWG600/AWG700 Series pattern file

Group

Mass memory

Syntax

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

Arguments

<wfm_name>, <filename>, <type>
<wfm_name>:: =<string>
<filename>:: =<string>
<type> = {TDS | TXT | TXT8 | TXT10 | WFM | PAT}

Related Commands

MMEMory:IMPort:PARameter:FREQuency[:UPDate][:STATe]?
MMEMory:IMPort:PARameter:LEVel[:UPDate]:CHANnel?
MMEMory:IMPort:PARameter:LEVel[:UPDate][:STATe]?
MEMory:IMPort:PARameter:NORMalize?

Examples

MMEMory:IMPort "sine1024","sine1024.txt",txt
Imports a waveform file named "sine1024", whose file format is text with normalized analog value.

MMEMory:IMPort: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?

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

Related Commands

MMEMory:IMPort

Examples

MMEMory:IMPort:PARameter:NORMalize NONE
Will not normalize the imported data.

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]?
Arguments	<state>::=<Boolean> 0 indicates False 1 indicates True At *RST, this returns True.
Returns	<state>
Related Commands	MMEMory:IMPort
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 that:

- 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?

Arguments

<NR1>
At *RST, the value is 1.

Returns

1 | 2

Related Commands

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

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]?

Arguments

<state>::=<Boolean>
0 indicates False
1 indicates True
At *RST, this returns True.

Returns

<Boolean>

Related Commands

MMEMory:IMPort

Examples

MMEMory:IMPort:PARameter:LEVel:UPDate:STATe 1
The instrument will automatically modify the amplitude and offset when importing the waveform data.

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>
Arguments	<directory_name>::=<string> specifies a new directory.
Related Commands	MMEMory:CDIRectory MMEMory:MSIS
Examples	MMEMory:MDIRectory "WAVEFORM" Makes the directory “WAVEFORM” .

MMEMory:MSIS(?)

This command selects 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?
Arguments	<msus> (mass storage unit specifier)::=<string>
Returns	<msus> (mass storage unit specifier)::=<string> At *RST, this values is “C:”
Related Commands	None
Examples	MMEMory:MSIS? Might return the following response: “X:”

OUTPut[n]:FILTer[:LPASs]:FREQuency(?)

This command sets or returns the low-pass filter frequency of the filter. INFinity is same as Through (no filter).

Group	Output
Syntax	OUTPut[n]:FILTer[:LPASs]:FREQuency {<NR3> | INFinity } OUTPut[n]:FILTer[:LPASs]:FREQuency?
Arguments	<NR3> At *RST, this value returns 9.9e37 (INFinity).
Returns	<NR3>
Related Commands	AWGControl:DOUTput[n][:STATe](?)
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 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] {ON|OFF|<NR1>} OUTPut[n][:STATe]?
Arguments	0 sets the channel output to False (OFF) 1 sets the channel output to True (ON) At *RST, this returns False.
Returns	<NR1>
Related Commands	None
Examples	OUTPut1:STATe ON Turns the channel 1 output on.

SEQuence:ELEMent[n]:GOTO:INDex(?)

This command sets or retrieves 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 true.

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?
Arguments	<target>::=<NR1>
Returns	<target>
Related Commands	SEQuence:ELEMent[n]:GOTO:STATe SEQuenc:ELEMent[n]:GOTO:TYPE(?)
Examples	SEQuence:ELEMent1:GOTO:INDex 6 Will cause the sequencer to jump to sixth element after executing the first element. SEQuence:ELEMent1:GOTO:INDex? Will return 6.

SEQuence:ELEMent[n]:GOTO:STATe(?)

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

Group	Sequence
Syntax	SEQuence:ELEMent[n]:GOTO:STATe {ON|OFF|<NR1>} SEQuence:ELEMent[n]:GOTO:STATe?
Arguments	0 indicates False 1 indicates True At *RST, this returns False. <n> is an index number of sequence.
Returns	<NR1>
Related Commands	SEQuenc:ELEMents[n]:GOTO:TYPE(?) SEQuence:ELEMent[n]:GOTO:INDex(?)
Examples	SEQuence:ELEMent1:GOTO:STATe 1 Sets the GOTO state to ON

SEQuence:ELEMent[n]:JTARget:INDex(?)

This command sets or retrieves 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?
Arguments	<target>::=<NR1> <n> is an index number of sequence.
Returns	<NR1>
Related Commands	SEQuence:ELEMent[n]:JTARget:TYPE(?)
Examples	SEQuence:ELEMent1:JTARget:INDex 10 Sets the jump target index to 10th element.

SEQuence:ELEMent[n]:JTARget:TYPE(?)

This command sets or queries the event jump target type for the jump. You can generate an event in three 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?
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
Related Commands	SEQuence:ELEMent1:JTARget:INDex
Examples	SEQuence:ELEMent1:JTARget:TYPE INDex Sets the jump target to INDex.

SEQuence:ELEMent[n]:LOOP:COUNt(?)

This command will set or query the loop count. Loop count setting for an element is ignored if SEQence:ELEMent[n]:LOOP:INFinite is set to TRUE.

Group	Sequence
Syntax	SEQuence:ELEMent[n]:LOOP:COUNt <NR1> SEQuence:ELEMent[n]:LOOP:COUNt?
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>
Related Commands	SEQuence:ELEMent[n]:LOOP:INFinite(?)
Examples	SEQuence:ELEMent:LOOPCOUNt 100 Sets the element loop count to 100.

SEQuence:ELEMent[n]:LOOP:INFinite(?)

This command 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 command or change the run mode to Continuous by using AWGControl:RMODe CONTinuous command.

Group	Sequence
Syntax	SEQuence:ELEMent[n]:LOOP:INFinite {ON|OFF|<NR1>} SEQuence:ELEMent[n]:LOOP:INFinite?
Arguments	0 indicates False (OFF) 1 indicates True (ON) At *RST, this returns False. The value of n is an index number of sequence.
Returns	<NR1>
Related Commands	SEQuence:ELEMent[n]:LOOP:COUNt(?)
Examples	SEQuence:ELEMent1:LOOP:INFinite 1 Sets the infinite flag to True (ON).

SEQuence:ELEMent[n]:WAVeform[m](?)

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

Note that:

- The value of n indicates index number of sequence.
- The value of m = 1 | 2 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 {1 | 2} <wfm_name>
SEQuence:ELEMent[n]:WAVeform {1 | 2}?

Arguments

<wfm_name>::=<string>

Returns

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

Related Commands

None

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:ELEMent[n]:TWAit(?)

This command sets or returns the wait trigger state for an element. You can send a trigger signal in three 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 {ON | OFF|<NR1>} SEQuence:ELEMent[n]:TWAit?
Arguments	0 indicates False (OFF) 1 indicates True (ON) At *RST, this returns False. In the OFF state, the sequencer ignores trigger signals. The value of n is an index number of sequence.
Returns	<NR1>
Related Commands	None
Examples	SEQuence:ELEMent1:TWAit 1 Sets the wait trigger state to ON.

SEQuence:JUMP[:IMMediate] (No query form)

This command executes the sequencer jump to the specified element index. This is called Force jump. This jump 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>
Arguments	<target>::=<NR1>
Related Commands	None
Examples	SEQuence:JUMP:IMMediate 10 Forces the sequencer to jump to index number 10.

SEQuence:LENGth(?)

This command 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?
Arguments	<NR1> At *RST, this returns 0.
Returns	<NR1>
Related Commands	None
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.

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?
Returns	<NR1>
Related Commands	STATus:OPERation:ENABle STATus:OPERation[:EVENt]?

STATus:OPERation:ENABle(?)

This command 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?
Arguments	<NR1>
Returns	<NR1>
Related Commands	STATus:OPERation:CONDition? STATus:OPERation[:EVENt]?

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]?
Returns	<NR1>
Related Commands	STATus:OPERation:CONDition? STATus:OPERation:ENABle

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?
Returns	<NR1>
Related Commands	STATus:QUEStionable:ENABle STATus:QUEStionable[:EVENt]?

STATus:QUEStionable:ENABle(?)

This command 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?
Returns	<NR1>
Related Commands	STATus:QUEStionable:CONDition? STATus:QUEStionable[:EVENt]?

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

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

Group	Status
Syntax	STAtus:QUEStionable[:EVENt]?
Returns	<NR1>
Related Commands	STATus:QUEStionable:CONDition? STATus:QUEStionable:ENABle
Examples	STATus:QUEStionable:EVENt? Might return 32, which indicates that the QEVR contains the binary number 00000000 00100000.

[SOURce[1]]:FREQuency[:CW| :FIXed] (?)

This command 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 AWG7000 Series is 1.

Group	Source
Syntax	[SOURce[1]]:FREQuency[:CW | :FIXed] <NR3> [SOURce[1]]:FREQuency?
Arguments	<NR3>.The value must be between 10 MHz to 10 GHz. At *RST, this returns 1.0000000E+10
Returns	<NR3>
Related Commands	[SOURce[n]]:WAVeform AWGControl:INTerleave[:STATe]
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?
Arguments	<NR3> At *RST, this returns 10 MHz.
Returns	<NR3>
Related Commands	[SOURce[1]]:ROSCillator:SOURce(?) [SOURce[1]]:ROSCillator:TYPE(?)
Examples	SOURce1:ROSCillator:FREQuency 10MHz Sets the reference oscillator source frequency to 10 MHz. SOURce1:ROSCillator:FREQuency? Will return 1.00000000E+7

[SOURce[1]]:ROSCillator:MULTiplier(?)

This command 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?
Arguments	<NR1> At *RST, this returns 1.
Returns	<NR1>
Related Commands	[SOURce[1]]:ROSCillator:SOURce(?) [SOURce[1]]:ROSCillator:TYPE(?)
Examples	SOURce1:ROSCillator:MULTiplier 10 Sets the multiplier rate to 10. SOURce1:ROSCillator:MULTiplier? Will 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?
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
Related Commands	[SOURce[1]]:ROSCillator:FREQuency(?) [SOURce[1]]:ROSCillator:TYPE(?)
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?
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
Related Commands	[SOURce[1]]:ROSCillator:FREQuency(?) [SOURce[1]]:ROSCillator:SOURce(?)
Examples	SOURce1:ROSCillator:SOURce EXTernal;TYPE FIXed;FREQuency 20MHz This selects a fixed frequency external reference oscillator whose frequency is 20 MHz.

[SOURce[n]]:DAC:RESolution(?)

This command sets or returns the DAC resolution.

Group	Source
Syntax	[SOURce[n]:DAC:RESolution <NR1> [SOURce[n]]:DAC:RESolution ?
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>
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(?)
Examples	[SOURce[n]:DAC:RESolution 10 Sets the DAC resolution to 10 bits.

[SOURce[n]]:FUNCtion:USER(?)

This command sets or returns the waveform to waveform memory.

- Use this command to directly load a waveform or pattern file for the AWG400/500/600/700 Series from mass memory to a specified channel.
- The waveform is internally converted to the AWG7000 Series format and inserted into the current waveform list. To successfully load a waveform, the waveform name should conform to AWG7000 Series waveform naming conventions.

Group	Source
Syntax	[SOURce[n]]:FUNCtion:USER <file_name> [SOURce[n]]:FUNCtion:USER?
Arguments	<file_name>:: = <string> Value of n indicates the channel number.
Returns	<file_name>
Related Commands	WLISt: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 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?
Arguments	<NR3> in the range of 0 to 300 ps. The value of n indicates the channel number. At *RST, this returns 0.
Returns	<NR3>
Related Commands	[SOURce[n][:DAC:RESolution(?)
Examples	SOURce1:MARKer1:DELay 20ps Sets the marker1 delay of channel1 to 20 picoseconds. SOURce1:MARKer1:DELay? Will return 2.00000000E -011 indicating 20 ps.

[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?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns 1 V.
Returns	<NR3>
Related Commands	[SOURce[n[]:DAC:RESolution(?) [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW(?)
Examples	SOUR1:MARK1:VOLT:HIGH 0.75 Sets the marker1 high to 0.75 volts.

[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]?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns 1 V (Vpp).
Returns	<NR3>
Related Commands	[SOURce[n[]:DAC:RESolution(?)
Examples	SOURce1:MARKer1:VOLTage:AMPLitude 0.5V Sets the channel1 marker1amplitude to 0.5 volts. SOURce1:MARKer1:VOLTage:AMPLitude? Will return 0.5 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?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns 0 V.
Returns	<NR3>
Related Commands	[SOURce[n[]:DAC:RESolution(?) [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH(?)
Examples	SOUR1:MARK1:VOLT: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?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns 0.5 V.
Returns	<NR3>
Related Commands	[SOURce[n[]:DAC:RESolution(?) [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH(?) [SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW(?)
Examples	SOURce1:MARKer1:VOLTage:OFFSet 1.0 Sets the offset channel1 marker1 to 1 V.

[SOURce[n]]:SKEW(?)

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

Group	Source
Syntax	[SOURce[n]]:SKEW <NR3> [SOURce[n]]:SKEW?
Arguments	<NR3> –100 ps to +100 ps. It can be changed by a minimum of 1ps at a time. The value of n indicates the channel number. At *RST, this returns 0 s.
Returns	<NR3>
Related Commands	None
Examples	SOURce2:SKEW 75ps Sets the skew for channel2 to 75 ps. SOURce2:SKEW? Will return 7.50000000E-011, indicating that the skew is 75 ps.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)

This command 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]?

Arguments

<NR3> in the range 50 mV to 2V pk-pk.
The value of n indicates the channel number.
At *RST, this returns 1 V (Vpp).

Returns

<NR3>

Related Commands

None

Examples

SOURce1:VOLTage:AMPLitude 1.5
Sets the amplitude of channel1 to1.5 volts.

SOURce1:VOLTage:AMPLitude?
Will return 1.5.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH(?)

This command 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?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns 0.5 V.
Returns	<NR3>
Related Commands	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW?
Examples	SOURce1:VOLTage:HIGH 0.75 Sets the channel1’s high to 0.75 volts.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW(?)

This command 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?
Arguments	<NR3> The value of n indicates the channel number. At *RST, this returns –0.5 V.
Returns	<NR3>
Related Commands	None
Examples	SOURce1:VOLTage:LOW 0.25 Sets the channel1 low to 0.25 volts.

[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)

This command 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?
Arguments	<NR3> in the range -0.5 V to +0.5V
	The value of n indicates the channel number.
	At *RST, this returns 0 V.
Returns	<NR3>
Related Commands	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:HIGH?
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate]:LOW?
	AWGControl:DOUTput[n][:STATe](?)
Examples	SOURce1:VOLTage:LEVel:IMMediate:OFFSet 0.5
	Sets the channel 1 offset to 0.5 V.

[SOURce[n]]:WAVeform(?)

This command 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?
Arguments	<wfm_name>::=<string> The value of n indicates the channel number.
Returns	<wfm_name>
Related Commands	[SOURce[n]]:FUNCtion:USER(?)
Examples	SOURce1:WAVeform “*Sine100” Loads a predefined waveform called "*Sine100” into channel1 memory. SOURce1:WAVeform? Will return “*Sine100”.

SYSTem:DATE(?)

This command 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?
Arguments	<year>::=<NRf> (Four digit number) <month>::=<NRf> from 1 to 12 <day>::=<NRf> from 1 to 31
Returns	<year>,<month>,<day>
Related Commands	None
Examples	SYSTem:DATE 2006,6,20 Sets the date to June 20, 2006.

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

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

Group	System
Syntax	SYSTem:ERRor[:NEXT]?
Returns	<Error / event number>, <error / event description [;device dependant info]> 0 – No Error Error / event number <NR1> error / event description <string>
Related Commands	None
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 {ON | OFF|<NR1>} SYSTem:KLOCk?
Arguments	0 or OFF indicates the front panel and keyboard are unlocked. 1 or ON indicates the front panel and keyboard are locked.
Returns	<NR1>
Related Commands	None
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 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?
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>
Related Commands	None
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?
Returns	<NR2>::=YYYY.V where YYYY represents the year version and V represents an approved revision number for that year.
Related Commands	None
Examples	SYSTem:VERSion? Might return 1999.0.

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 sets or returns the trigger impedance. It applies only to the external trigger.

Group	Trigger
Syntax	TRIGger[:SEQuence]:IMPedance <impedance> TRIGger[:SEQuence]:IMPedance?
Arguments	<impedance>::=<NR3> At *RST, this returns 1 kΩ
Returns	<NR3>
Related Commands	None
Examples	TRIGger:SEQuence:IMPedance 50 Selects 50 Ω impedance for the external trigger input.

TRIGger[:SEQuence]:LEVel(?)

This command sets or returns the trigger input level (threshold).

Group	Trigger
Syntax	TRIGger[:SEQuence]:LEVel <NR3>
	TRIGger[:SEQuence]:LEVel?
Arguments	<NR3>
	At *RST, this returns 1.4 V.
Returns	<NR3>
Related Commands	TRIGger[:SEQuence]:SOURce
Examples	TRIGger:SEQuence:LEVel 200mV
	Sets the trigger level to 200 mV.

TRIGger[:SEQuence]:POLarity(?)

This command 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?
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
Related Commands	AWGControl:RMODe TRIGger[:SEQuence]:LEVel
Examples	TRIGger[:SEQuence]:POLarity NEGative Selects the Negative polarity.

TRIGger[:SEQuence]:SLOPe(?)

This command 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?
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
Related Commands	TRIGger[:SEQuence]:SOURce
Examples	TRIGger:SEQuence:SLOPe NEGative Selects the Negative slope.

TRIGger[:SEQuence]:SOURce(?)

This command sets or returns the trigger source.

Group	Trigger
Syntax	TRIGger[:SEQuence]:SOURce {INTernal | EXTernal} TRIGger[:SEQuence]:SOURce?
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
Related Commands	TRIGger[:SEQuence]:LEVel TRIGger[:SEQuence]:POLarity TRIGger[:SEQuence]:SLOPe TRIGger[:SEQuence]:TIMer
Examples	TRIGger:SEQuence:SOURce INTernal Selects the internal clock as the trigger source.

TRIGger[:SEQuence]:TIMer(?)

This command sets or returns the internal trigger rate (trigger interval).

Group	Trigger
Syntax	TRIGger[:SEQuence]:TIMer<NR3> TRIGger[:SEQuence]:TIMer?
Arguments	<NR3> At *RST, this returns 100 ms.
Returns	<NR3>
Related Commands	TRIGger[:SEQuence]:SOURce
Examples	TRIGger:SEQuence:TIMer 5ms Sets the internal trigger rate to 5 ms.

TRIGger[:SEQuence]:WVALue (?)

This command 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?
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
Related Commands	TRIGger[:SEQuence]:SOURce
Examples	TRIGger:SEQuence:WVALue LAST Selects the last value as the output level.

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>
Arguments	<Index> <Index>::= <NR1>
Returns	<string>::=<wfm_name> is the waveform name specified by <index>.
Related Commands	None
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?
Returns	<NR1>
Related Commands	WLISt:WAVeform:NEW
Examples	WLISt:SIZE? Might return 20 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 that:

- Before transferring data to the instrument, a waveform must be created using the WLISt:WAVeform:NEW command.
- Using StartIndex and Size, part of a waveform can be transferred at a time. Very large waveforms can be transferred in chunks.
- 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 four bytes.
- The minimum size of the waveform must be 1 and the maximum size depends on the instrument model and configuration.

Group	Waveform
Syntax	WLISt:WAVeform:DATA <wfm_name>[,<StartIndex>[,<Size>]],<Arbitrary block> WLISt:WAVeform:DATA? <wfm_name>[,<StartIndex>[,<Size>]]
Arguments	StartIndex, Size,<block_data> <wfm_name>:: =<string> <StartIndex>::=<NR1> <Size>::=<NR1> <Block_data>::=IEEE 488.2 block
Returns	<block_data>
Related Commands	WAVeform:NEW WLISt:WAVeform:MARKer: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 four times the size of the waveform.

WLISt:WAVEeform:DELete (No query form)

This command deletes the waveform from the currently loaded setup.

Note that:

- 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:WAVEeform:DELete {<wfm_name> | ALL }

Arguments
<wfm_name>::=<string>

Related Commands
WLISt:SIZe

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>
Arguments	<wfm_name>::=<string>
Returns	<NR1>
Related Commands	WLISt:WAVeform:NEW
Examples	WLISt:WAVeform:LENGth? “*Sine 360” Will return 360.

WLISt:WAVeform:MARKer:DATA(?)

This command sets or queries the waveform marker data.

Note that:

- This command returns or sends only marker data for the waveform.
- Each marker data occupies one bit. Two most significant bits of each byte is used for marker1 and marker2.
- 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.

Group Waveform

Syntax

```
WLISt:WAVeform:MARKer:DATA
<wfm_name>[,<StartIndex>[,<Size>]],<block_data>
WLISt:WAVeform;MARKer:DATA?
<wfm_name>[,<StartIndex>[,<Size>]]
```

Arguments

```
<wfm_name>::=<string>
<StartIndex>::=<NR1>
<Size>::=<NR1>
<Block_data>::=IEEE 488.2 block
```

Returns <block_data>

Related Commands None

Examples

```
WLISt:WAVeform:MAKRer: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.

The newly created waveform will not have any data. The data for the new waveform must be set separately using the WLISt:WAVeform:DATA command.

Group	Waveform
Syntax	WLISt:WAVeform:NEW <wfm_name>,<Size>,<Type>
Arguments	<wfm_name>::=<string> <Size>:: =<NR1> <Type>::={REAL | INTeger}
Related Commands	WLISt:WAVeform:DATA
Examples	WLISt:WAVeform:NEW “Test1”, 1024, INTeger Creates a new integer waveform called Test1 with 1024 points.

WLISt:WAVeform:PREDefined? (Query only)

This query returns true or false based on whether the waveform is predefined.

Note that:

- 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. Predefined waveforms are not counted when you use WLISt:SIZE? to get the length of the waveform list.
- 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>
Arguments	<wfm_name>:: =<string>
Returns	<state>::=<Boolean>
Related Commands	None
Examples	WLISt:WAVeform:PREDefined? “*Sine3600” Will return 1 indicating that it is a predefined waveform.

WLISt:WAVeform:TSTamp? (Query only)

This query returns the time stamp of the waveform.

Note that:

- 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>

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

Related Commands

None

Examples

WLISt:WAVeform:TSTamp? “Sine”
Will return the date and time the “Sine” waveform was created or last modified.

WLISt:WAVeform:TSTamp? “*DC”
Will 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>
Arguments	<wfm_name>::=<string>
Returns	INT | REAL
Related Commands	WLISt:WAVeform:NEW
Examples	WLISt:WAVeform:TYPE? "*Ramp1000" Will return REAL.

Status and Events

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.

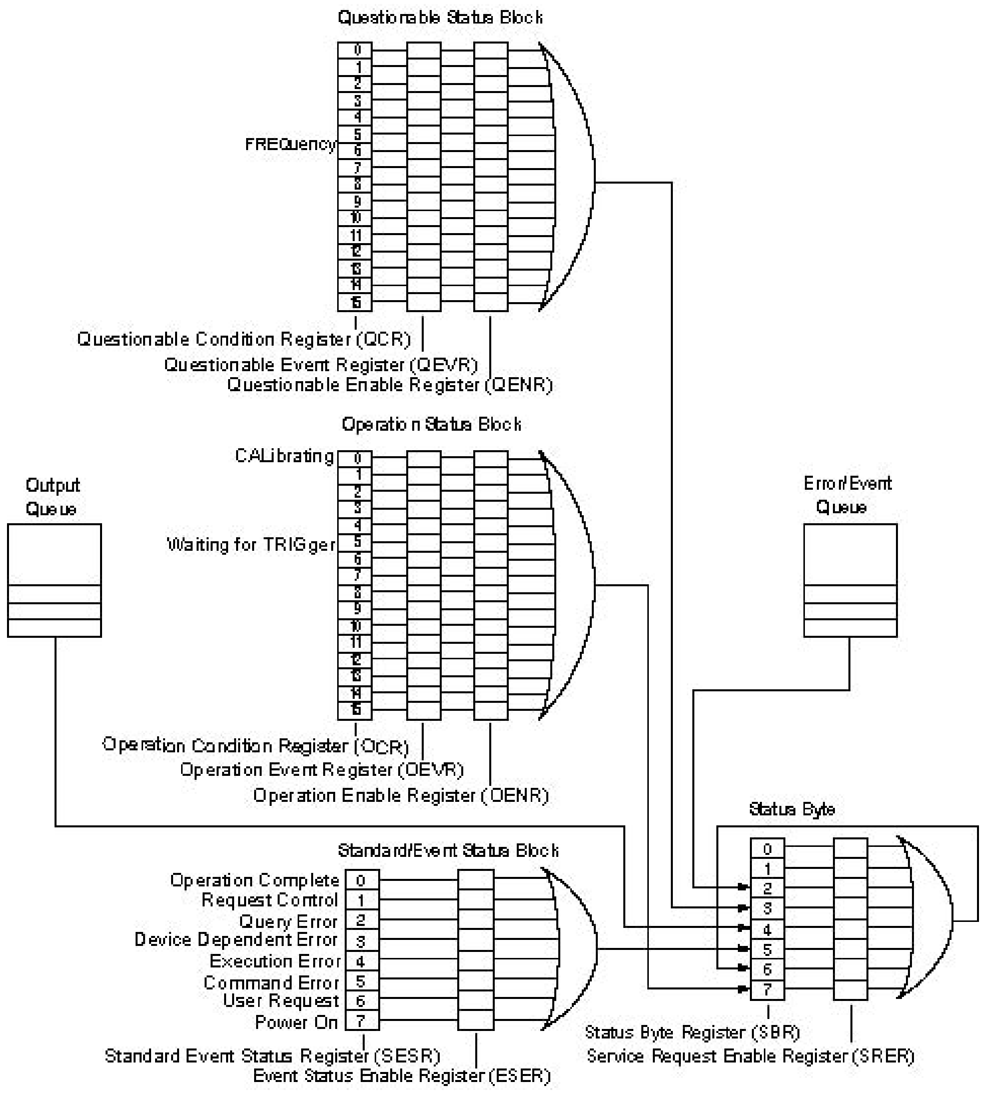

Questionable Status Block
FREQuency
Questionable Condition Register (QCR)
Questionable Event Register (QEVR)
Questionable Enable Register (QENR)
Operation Status Block
CALibrating
Waiting for TRIGger
Operation Condition Register (OCR)
Operation Event Register (OEVR)
Operation Enable Register (OENR)
Output Queue
Error/Event Queue
Status Byte
Standard/Event Status Block
Operation Complete
Request Control
Query Error
Device Dependent Error
Execution Error
Command Error
User Request
Power On
Standard Event Status Register (SESR)
Event Status Enable Register (ESER)
Status Byte Register (SBR)
Service Request Enable Register (SRER)

Figure 8 : Error and Event handling process overview

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.

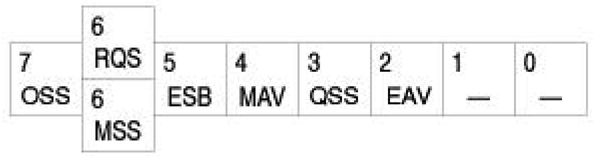


Figure 9: The Status Byte Register (SBR)

The following table lists the SBR bit functions.

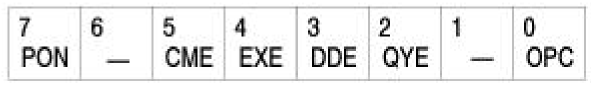
Table 25: SBR bit functions

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)
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.



7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

Figure 10: The Standard Event Status Register (SESR)

The following table lists the SESR bit functions.

Table 26: SESR bit functions

Bit	Function
7	Power On (PON). Indicates that the power to the instrument is 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.

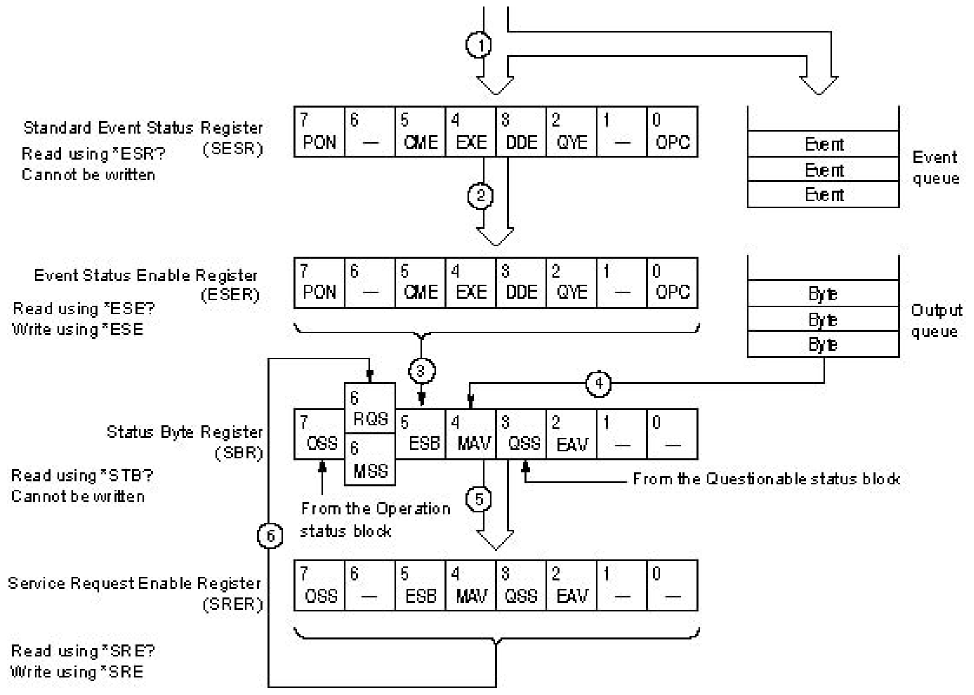


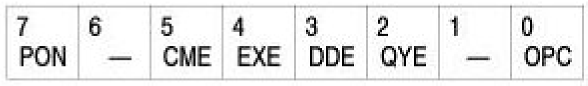
Figure 11: Status and Event processing sequence –Standard/Event status block

Event Status Enable Register (ESER)

The ESER is made up of bits defined exactly the same as bits 0 through 7 in the SESR. You can 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

Figure 12: The Event Status Enable Register (ESER)

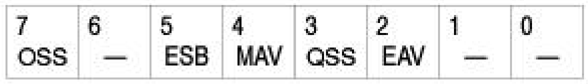
Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR. You can 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	—	—

Figure 13: The Event Status Enable Register (ESER)

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]?
```

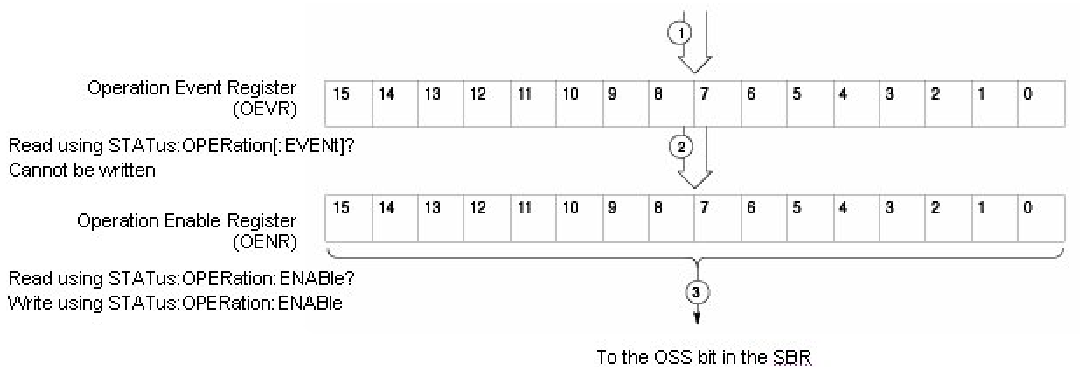
If you omit the *ESR? query, the SESR bit will remain set, even if the event disappears from the Error/Event Queue.

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).



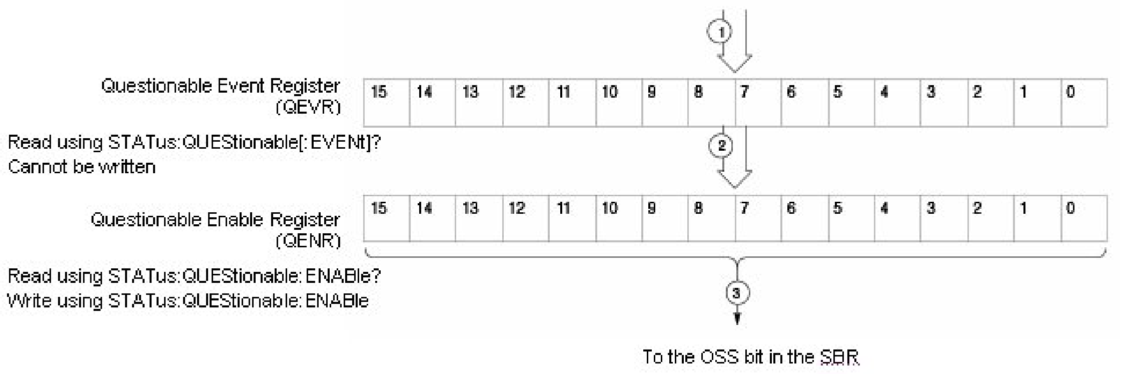
Operation Event Register (OEVR)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Read using STATus:OPERation[:EVENt]?
Cannot be written
Operation Enable Register (OENR)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Read using STATus:OPERation:ENABle?
Write using STATus:OPERation:ENABle
1
2
3
To the OSS bit in the SBR

Figure 14: Status and Event processing sequence – Operation status block

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).



Questionable Event Register (QEVR)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Read using STATus:QUEStionable[:EVENt]?
Cannot be written
Questionable Enable Register (QENR)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Read using STATus:QUEStionable:ENABle?
Write using STATus:QUEStionable:ENABle
1
2
3
To the OSS bit in the SBR

Figure 15: Status and Event processing sequence – Questionable status block

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).

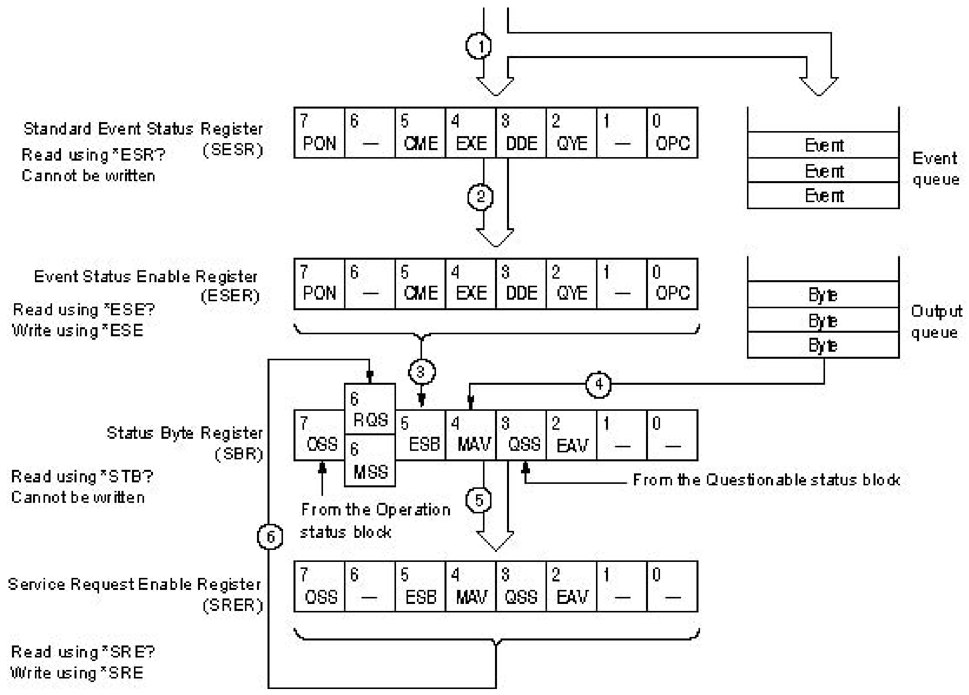


Figure 16: Status and Event processing sequence –Standard/Event status block

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

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, you can find its error class by checking for the code range in tables that are organized by event class.

Table 27: 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 28: 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 29: 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
-171	Invalid expression
-178	Expression data not allowed

Table 29: Command errors (cont.)

Error code	Error message
-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 30: 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
-250	Mass storage error
-251	Missing mass storage
-252	Missing media

Table 30: Execution errors (cont.)

Error code	Error message
-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 31: 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 32: 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 33: Power-On event

Error code	Error message
-500	Power on

User request Event

Table 34: User request event

Error code	Error message
-600	User request

Request Control Event

Table 35: Request control event

Error code	Error message
-700	Request control

Operation Complete Event

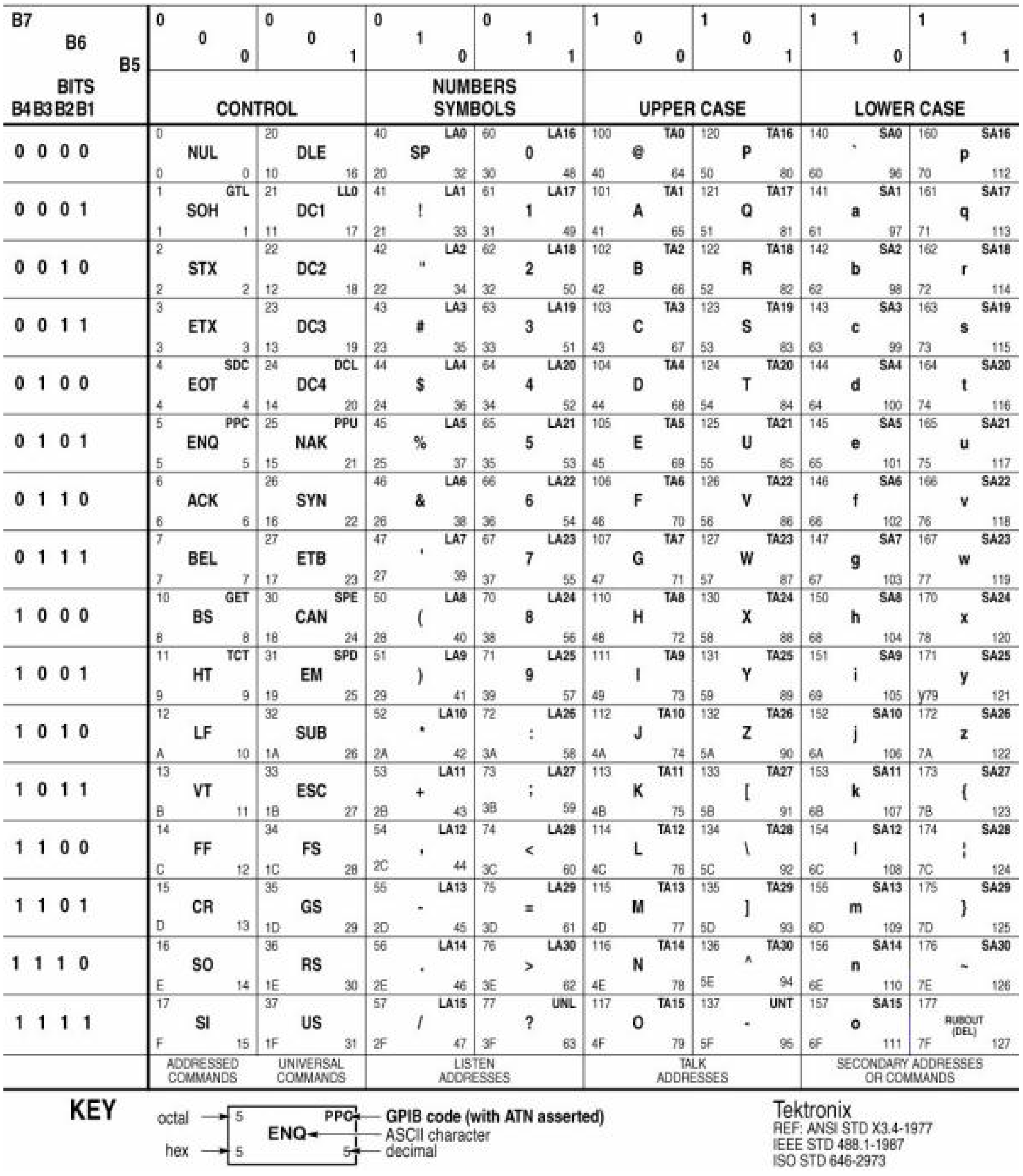
Table 36: Operation complete event

Error code	Error message
-800	Operation complete

Appendices

Character Charts

ASCII & GPIB code chart



B7 B6 B5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
BITS B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 LA0 SP 20 32	60 LA16 0 30 48	100 TA0 @ 40 64	120 TA16 P 50 80	140 SA0 ` 60 96	160 SA16 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LLO 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 y79 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 &lt; 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 &gt; 3E 62	116 TA14 N 4E 78	136 TA30 ^ 5E 94	156 SA14 n 6E 110	176 SA30 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 LA15 / 2F 47	77 UNL ? 3F 63	117 TA15 O 4F 79	137 UNT _ 5F 95	157 SA15 o 6F 111	177 RUBOUT (DEL) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

octal → 5 PPC ← GPIB code (with ATN asserted)

ENQ ← ASCII character

hex → 5 5 ← decimal

Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

GPIB Interface Specifications

GPIB Interface Specifications

This appendix lists and describes the GPIB functions and messages that the arbitrary waveform generator implements.

Interface Functions

The following table lists the GPIB interface functions this instrument implements. Each function is briefly described.

Table 37: 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.
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.

Table 37: GPIB function implementation (cont.)

Interface function	Implemented subset	Capability	Description
Device Trigger (DT)	DT1	Complete	-
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 38: 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.
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.

Table 38: AWG standard interface messages (cont.)

Message	GPIB	Description
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	

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 1999.0	Not defined SCPI 1999.0
AWGControl			
	AWGControl:CLOCk:DRATe(?)		√
	AWGControl:CLOCk:SOURce(?)		√
	AWGControl:CONFigure:CNUMber?		√
	AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)		√
	AWGControl:DC[n][:STATe](?)		√
	AWGControl:DOUTput[n][:STATe](?)		√
	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?	√	
	DIAGnositc:DATA?		√
	DIAGnositc[: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:NORMalize(?)		√
	MMEMory:MDIRectory		√
	MMEMory:MSIS(?)	√	

Group	Command	Defined in SCPI 1999.0	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]]:MARKer[1|2]:DELay(?)		√
	[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]]: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(?)		
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]?	√	

Group	Command	Defined in SCPI 1999.0	Not defined SCPI 1999.0
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]: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?		√

Network Interface Specification

The arbitrary waveform generator supports remote control with the Ethernet interface. This Appendix describes the network interface 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 Ethernet interface.

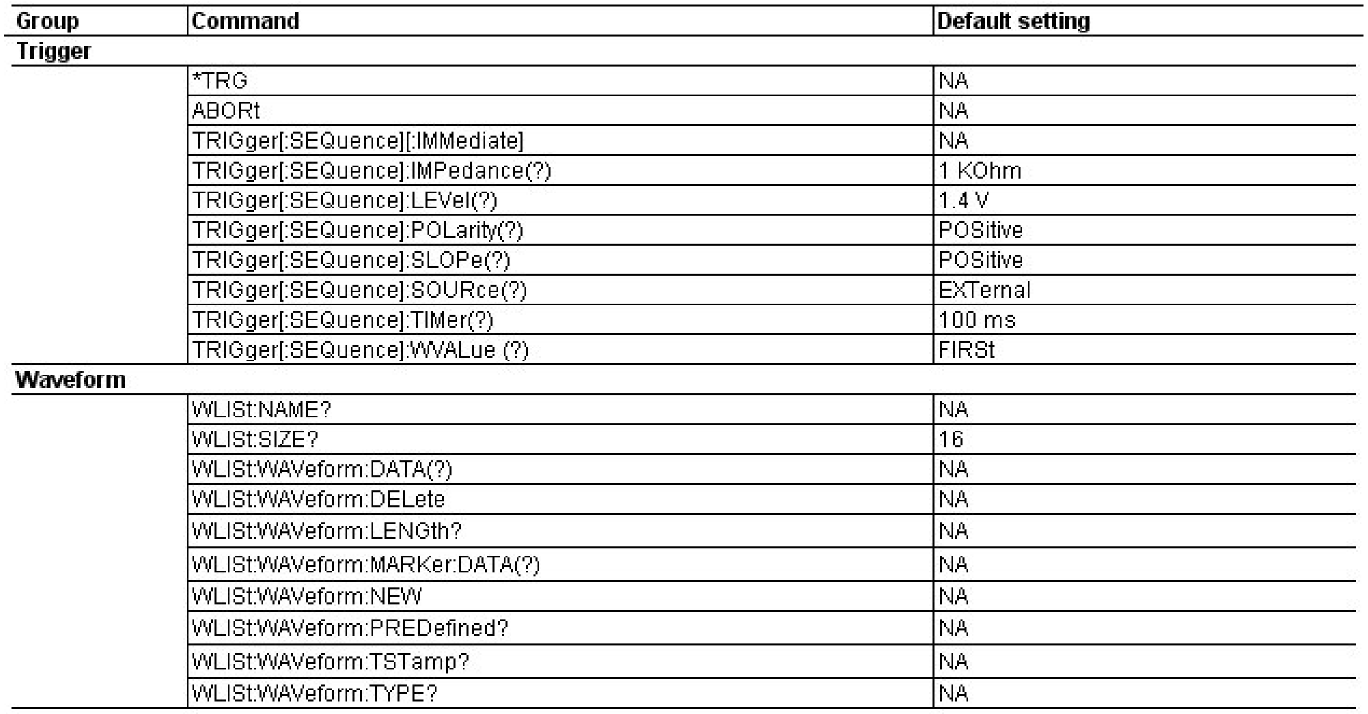
- The Line Feed (LF) code is needed as a terminator at the end of a message.
- The IEEE 488.1 standard (for example, Device Clear, Service Request, etc.) is not supported.
- The Message Exchange Control Protocol in the IEEE 488.2 is not supported. However, common commands such as *ESE and the event handling 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.

Factory Initialization Settings

The following tables list the default settings for the each command.

Group	Command	Default setting
AWGControl		
	AWGControl:CLOCk:DRATe(?)	1 (AWG710X), 2 (AWG705X)
	AWGControl:CLOCk:SOURce(?)	INTernal
	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: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](?)	1
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
	MMEMory:IMPort:PARameter:NORMalize(?)	NONE
	MMEMory:MDIRectory	\
	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]]:DAC:RESolution(?)	8
	[SOURce[n]]:FUNCtion:USER(?)	"", "C:"
	[SOURce[n]]:MARKer[1|2]:DELay(?)	0
	[SOURce[n]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	1 V (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]]:SKEW(?)	0 s
	[SOURce[n]]:VOLTage[:LEVel][:IMMediate][:AMPLitude](?)	1 V (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(?)	""
Synchronization		
	*OPC(?)	NA
	*WAI	NA



Group	Command	Default setting
Trigger		
	*TRG	NA
	ABORt	NA
	TRIGger[:SEQuence][:IMMediate]	NA
	TRIGger[:SEQuence]:IMPedance(?)	1 KOhm
	TRIGger[:SEQuence]:LEVel(?)	1.4 V
	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?	16
	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

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.

AWG7000 Series Command Group	AWG400	AWG500	AWG600/700	Note
AWGControl				
AWGControl:CLOCk:DRATe(?)				
AWGControl:CLOCk:SOURce(?)	✓	✓		AWG400 / AWG500 only
AWGControl:CONFigure:CNUMber?				
AWGControl:DC[n][:STATe](?)				
AWGControl:DC[n]:VOLTage[:LEVel][:IMMediate]:OFFSet(?)				
AWGControl:DOUTput[n][:STATe](?)	✓	✓	✓	
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(?)				

AWG7000 Series Command Group	AWG400	AWG500	AWG600/700	Note
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:NORMalize(?)				
MMEMory:MDIRectory	✓	✓	✓	
MMEMory:MSIS(?)		See Note		Drive letter, instead of "MAIN" or "FLOPpy"
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[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]]: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]]: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(?)				

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]: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