Tektronix®

AWG4162
Arbitrary Waveform Generator

Advanced Programmer Manual

077-1309-00

Tektronix®

AWG4162
Arbitrary Waveform Generator

Advanced Programmer Manual

Register now!
Click the following link to protect your product.
➔ www.tek.com/register
www.tek.com
077-1309-00

Contacting Tektronix, Inc.

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

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

Warranty

Tektronix warrants that the product will be free from defects in materials and workmanship for a period of three (3) years from the date of original purchase from an authorized Tektronix distributor. If the product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product. Batteries are excluded from this warranty. Parts, modules and replacement products used by Tektronix for warranty work may be new or reconditioned to like new performance. All replaced parts, modules and products become the property of Tektronix.

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

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

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THE PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

[W16 – 15AUG04]

Table of Contents

Preface ..... **vi**

Documentation vii

General Features viii

Syntax and Commands **1-1**

Command Syntax 1-1

Backus-Naur Form Definition 1-1

Command and Query Structure 1-2

Command Entry 1-2

Command Messages 1-3

Commands 1-4

Queries 1-4

SCPI Commands and Queries 1-4

Creating Commands 1-5

Parameters 1-5

Creating Queries 1-5

Query Responses 1-5

Special Characters 1-6

Abbreviating Commands, Queries, and Parameters 1-6

Concatenating Commands and Queries 1-6

Termination 1-7

Parameter Types 1-8

Block Arguments 1-8

Arbitrary Block Arguments 1-9

Quoted Strings 1-9

Unit and SI Prefixes 1-10

General Rules for Using SCPI Commands 1-11

Command Groups 1-12

Calibration and Diagnostic Commands 1-12

Control Commands 1-13

Event Commands 1-14

Instrument Commands 1-14

Mass Memory Commands 1-15

Output Commands 1-15

Sequence Commands 1-16

Source Commands 1-17

Status Commands 1-18

Subsequence Commands 1-19

Synchronization Commands 1-20

System Commands 1-20

Trigger Commands 1-20

Waveform Commands 1-21

Channel Number Optional Commands 1-23

Waveform Data Format **2-1**

Introduction 2-1

Waveform Data Transfer 2-1

Block Data Format 2-1
Real Format 2-2
Digital Data 2-2
Granularity 2-4
Creating and Working with Sequences 2-4

Advanced Command Descriptions 3-1

ABORt 3-1
ANALOGCHANnel[n]:EVENTS[i]:OPERator[j] 3-1
ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j] 3-2
AWGControl:APPLication:RUN 3-3
AWGControl:APPLication:STATe? 3-3
AWGControl[:CHANnel[n]]:FUNCTionality 3-4
AWGControl[:CHANnel[n]]:RSTate? 3-5
AWGControl[:CHANnel[n]]:WAITstate 3-5
AWGControl:CONFigure:CNUMber? 3-6
AWGControl:CONFigure:SEQuencer:MODE 3-6
AWGControl:EVENt:DJUMp:DEFine 3-7
AWGControl:EVENt:JMODe 3-8
AWGControl:EVENt:TABLe[:IMMediate] 3-8
AWGControl:RMODe 3-9
AWGControl:RUN[:IMMediate] 3-10
AWGControl:SNAMe? 3-10
AWGControl:SREStore 3-11
AWGControl:SSAVe 3-11
AWGControl:STOP[:IMMediate] 3-12
AWGControl:UPDATEdata 3-12
*CAL? 3-13
CALibration[:ALL] 3-14
*CLS 3-14
DIAGnostic:DATA? 3-15
DIAGnostic[:IMMediate] 3-15
EVENTS:TRIGINTIMERPeriod 3-16
EVENTS:TRIGINTHReshold 3-17

*IDN? ... 3-17
INSTrument:COUPle:SOURce ... 3-18
MMEMory:CATalog? ... 3-19
MMEMory:CDIRectory ... 3-20
MMEMory:DATA ... 3-21
MMEMory:DELete ... 3-21
MMEMory:EXPort ... 3-22
MMEMory:IMPort ... 3-23
MMEMory:MDIRectory ... 3-24
MMEMory:MSIS ... 3-25
*OPC ... 3-26
*OPT? ... 3-26
OUTPut[n][:STATe] ... 3-27
*RST ... 3-27
SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO ... 3-28
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr ... 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent ... 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType ... 3-30
SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions ... 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence ... 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent ... 3-32
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform ... 3-33
SEQuence[:CHANnel[n]]:LENGth ... 3-33
SLISt[:CHANnel[n]]:NAME? ... 3-34
SLISt[:CHANnel[n]]:SIZE? ... 3-35
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO ... 3-35
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr ... 3-36
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent ... 3-37
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType ... 3-38
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions ... 3-39
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent ... 3-40
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform ... 3-41
SLISt:SUBSequence:DELete ... 3-41
SLISt:SUBSequence:LENGth ... 3-42

SLISt:SUBSequence:NEW 3-43
SOURce[n]:AMPlitudescale[:IMMediate] 3-43
SOURce[n]:ATTENuation 3-44
SOURce[n]:ATTENuation:TYPe 3-45
SOURce[n]:DAC:RESolution? 3-45
SOURce[n]:DIGital:SKEW[:IMMediate] 3-46
SOURce[n]:DIGital:STATe 3-47
SOURce[n]:MARKer:SKEW[:IMMediate] 3-47
SOURce[n]:MARKer:TYPe 3-48
SOURce[n]:MARKer:VOLTage[:LEVel][:IMMediate][:AMPLitude] 3-49
SOURce[n]:MARKer:VOLTage:SELection[:IMMediate] 3-50
SOURce[n]:OFFSet[:IMMediate] 3-50
SOURce[n]:OUTputtype 3-51
SOURce[n]:SKEW 3-52
SOURce[n]:VOCM[:IMMediate] 3-52
[SOURce[n]:]WAVeform 3-53
SYSTem:ERRor[:NEXT]? 3-54
SYSTem:VERSion? 3-55
TIMING[n]:CLOCKSOUrce 3-55
TIMING[n]:REFCLOCKSOUrce 3-56
TIMING[n]:SAMPLEFReq 3-57
*TRG 3-58
TRIGger[:SEQuence][:IMMediate] 3-58
WLISt:LAST? 3-59
WLISt:LIST? 3-60
WLISt:NAME? 3-60
WLISt:SIZE? 3-61
WLISt:WAVeform:DATA 3-61
WLISt:WAVeform:DELete 3-62
WLISt:WAVeform:DIGITAL:DATA 3-63
WLISt:WAVeform:GRANularity? 3-64
WLISt:WAVeform:LENGth? 3-64
WLISt:WAVeform:LMAXimum? 3-65
WLISt:WAVeform:LMINimum? 3-65

WLISt:WAVeform:NEW 3-66
WLISt:WAVeform:NORMalize 3-67
WLISt:WAVeform:PREDefined? 3-67
WLISt:WAVeform:RESAmple 3-68
WLISt:WAVeform:TYPE? 3-69

Appendix A: Reset Defaults A-1
Reset Defaults A-1

Appendix B: Error Codes B-1
Command Error Codes and Messages B-1
Diagnostic Error Codes B-3

Appendix C: Predefined Waveforms C-1
Waveforms Included with the AWG4162 ..... C-1

Index Index-1

Preface

This manual provides advanced operating information for the following products:

AWG4162

The manual consists of the following sections:

Syntax and Commands (on page 1-1) defines the command syntax and processing conventions and describes command notation.

Waveform Data Format (on page 2-1) describes the Real data format, digital data modes, and waveform data granularity.

Advanced Command Descriptions (on page 3-1) presents detailed information about using each of the commands, including syntax, arguments, and examples.

Reset Defaults (on page A-1) lists the instrument settings after a reset command has been received.

Error Codes (on page B-1) lists the error codes and error descriptions that are returned when there is a command syntax error. It also lists error codes and error descriptions that are returned when there is a diagnostic failure.

Predefined Waveforms (on page C-1) describes the waveforms that already defined in the AWG4162 when shipped.

Documentation

The following table lists related documentation available for your AWG4162. The documentation is available on the document CD-ROM and on the Tektronix Web site (http://www.tek.com/manuals).

Item	Purpose	Location
Compliance and Safety Instructions	Compliance, safety, and basic installation information	Printed and shipped with your instrument and Tektronix Web site (http://www.tek.com/manuals)
Basic Application Help	Basic application operating information	Instrument and PDF on the Tektronix Web site (http://www.tek.com/manuals)
Advanced Application Help	Advanced application operating information	Instrument and PDF on the Tektronix Web site (http://www.tek.com/manuals)
Basic Programmer Manual	Basic programming information	PDF on Tektronix Web site (http://www.tek.com/manuals)
Advanced Programmer Manual	Advanced programming information	PDF on Tektronix Web site (http://www.tek.com/manuals)
Service Manual	Instrument servicing procedures and replaceable parts list	PDF on Tektronix Web site (http://www.tek.com/manuals)
Technical Reference	Instrument specifications and performance verification procedures	PDF on Tektronix Web site (http://www.tek.com/manuals)
Declassification and Security Instructions	Describes how to sanitize security space in the Tektronix AWG4000 Series arbitrary waveform generator hard disk	PDF on Tektronix Web site (http://www.tek.com/manuals)

General Features

The AWG4162 has two working modes:

- Basic (DDS) mode
- Two analog channels
- 600 MHz sine waveforms
- 2.5 GS/s, 14-bit, 16 kpts arbitrary waveforms
- Amplitude up to 5 Vp-p into 50 Ω load
- Advanced (Arbitrary) mode
- Two analog channels
- 16/32-bit digital channels (optional)
- 1/16/32/64 Mpts per channel arbitrary waveform memory (optional)
- Up to 750 MHz bandwidth
- SFDR < -60 dBc

This manual describes how to use the AWG4162 in advanced mode.

Syntax and Commands

This section provides the following information:

- Command Syntax (on page 1-1) defines the command syntax and processing conventions.
- Command Groups (on page 1-11) describes command groups and lists the commands by function.

The next section, Advanced Command Descriptions (on page 3-1), describes in detail the notation of each of the advanced commands in alphabetical order.

Command Syntax

You can control the operations and functions of the arbitrary waveform generator through the LAN or USBTMC 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 Command Groups (on page 1-11) for a listing of the commands by command group, or use the index to locate a specific command.

Backus-Naur Form Definition

This manual may describe commands and queries using the Backus-Naur Form (BNF) notation. The following table defines the standard BNF symbols.

BNF symbols and meanings

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

Command and Query Structure

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

Most commands have both a set form and a query form. The query form of the command is the same as the set form except that it ends with a question mark. For example, the set command `FILEsystem:CWDirectory` has a query form, `FILEsystem:CWDirectory?`. Not all commands have both a set and a query form; some commands are set only and some are query only.

A few commands have both a set and query action. For example, the `*CAL?` command runs a self-calibration program on the instrument, then returns the result of the calibration.

Command Entry

Follow these general rules when entering commands:

- Enter commands in uppercase or lowercase
- 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 that consist of just a combination of white space characters and line feeds.

Command 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 consist of five element types.

Command message elements

Symbol	Meaning
<Header>	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. The beginning colon can never be used with command headers beginning with a star (*).
<Mnemonic>	A header subfunction. Some command headers have only one mnemonic. If a command header has multiple mnemonics, they are always separated from each other by a colon (:) character.
<Argument>	A quantity, quality, restriction, or limit associated with the header. Not all commands have an argument, while other commands have multiple arguments. A <space> separates arguments from the header. A <comma> separates arguments from each other.
<Comma>	A single comma between arguments of multiple-argument commands. It may optionally have white-space characters before and after the comma.
<Space>	A white-space character between command header and argument. It may optionally consist of multiple white-space characters.

The following figure shows the five command message elements.

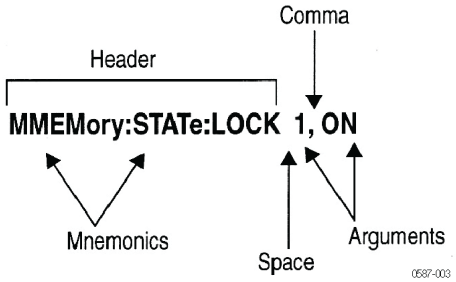

Figure 1: AWG4162 command message elements

Commands

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

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

A command header is made up 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 of 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 arbitrary waveform generator to return information about its status or settings. Queries have the structure:

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

You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level.

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 as shown in the following figure that represents a subsystem. The top level of the tree is the root node; it is followed by one or more lower-level nodes.

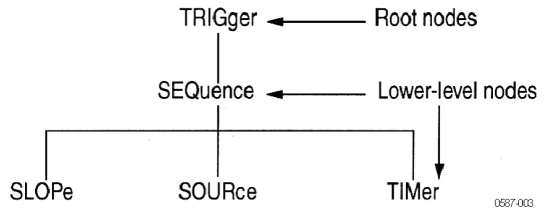

Figure 2: Example of SCPI subsystem hierarchy tree

You can create commands and queries from these subsystem hierarchy trees. Commands specify actions for the instrument to perform. Queries return measurement data and information about parameter settings.

Creating Commands

SCPI commands are created by stringing together the nodes of a subsystem hierarchy and separating each node by a colon.

In the figure above, `TRIGger` is the root node, and `SEQuence`, `SLOPe`, and `TIMer` are lower-level nodes.

To create a SCPI command, start with the root node `TRIGger` and move down the tree structure, adding nodes until you reach the end of a branch. Most commands and some queries have parameters; you must include a value for these parameters. If you specify a parameter value that is out of range, the parameter will be set to a default value. The command descriptions list the valid values for all parameters.

For example, `TRIGger:SEQuence:SOURce INTernal` is a valid SCPI command created from the hierarchy tree. (See the figure in SCPI Commands and Queries (on page 1-4).)

Parameters

Parameters are indicated by angle brackets, such as <file_name>.There are several different types of parameters. (See the table in Parameter Types (on page 1-8).) The parameter type is listed after the parameter. Some parameter types are defined specifically for the arbitrary waveform generator command set and some are defined by SCPI.

Creating Queries

To create a query, start at the root node of a tree structure, move down to the end of a branch, and add a question mark. `TRIGger:SEQuence:SOURce?` is an example of a valid SCPI query using the hierarchy tree in the figure in SCPI Commands and Queries (on page 1-4).

Query Responses

The query causes the arbitrary waveform generator to return information about its status or settings. When a query is sent to the arbitrary waveform generator, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format, as shown in the following table.

Query response examples

Query Command	Response
SOURce:PULSe:DCYcle?	50.0
OUTPut:POLarity?	NORM

Special Characters

The Line Feed (LF) character (ASCII 10), and all characters in the range of ASCII 127 through 255 are defined as special characters. These characters are used in arbitrary block arguments only; using these characters in other parts of any command yields unpredictable results.

Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these short forms as a combination of uppercase and lowercase letters. The uppercase letters indicate the accepted short form of a command. As shown in the following figure, you can create a short form by using only the uppercase letters. The accepted short form and the long form are equivalent and request the same action of the instrument.

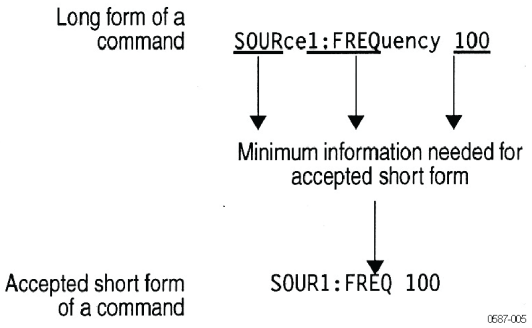

Figure 3: Example of abbreviating a command

NOTE. *The numeric suffix of a command or query may be included in either the long form or short form; the arbitrary waveform generator defaults to "1" if no suffix is used.*

Concatenating Commands and Queries

You can chain (concatenate) several commands or queries together into a single message. The instrument executes chained commands in the order received.

To create a chained message, first create a command or query, add a semicolon (;), and then add more commands or queries and semicolons until the message is complete. If the command following a semicolon is a root node, precede it with a colon (:). The following figure illustrates a chained message consisting of several commands and queries. The chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

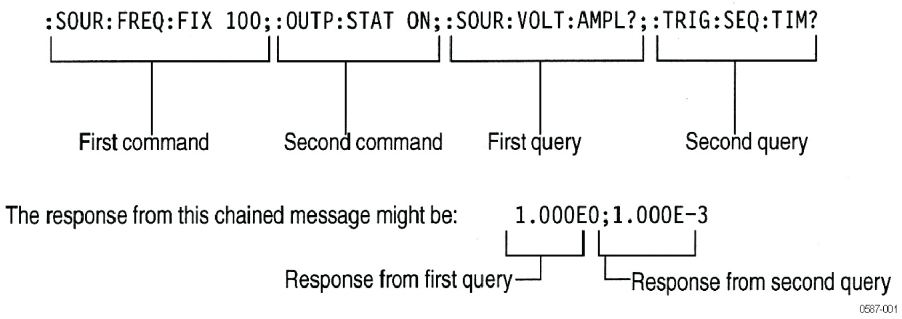

Figure 4: Example of chaining commands and queries

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes. In the following figure, the second command has the same root node (TRIGger:SEQuence) as the first command, so these nodes can be omitted.

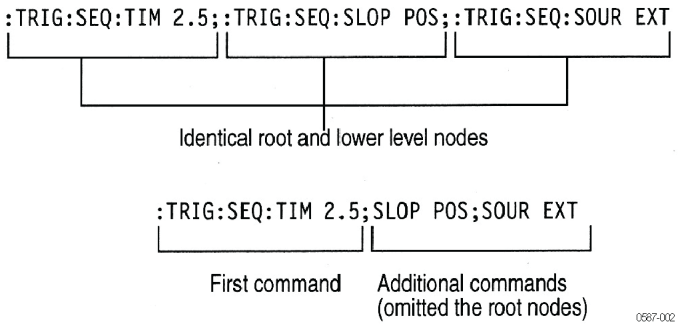

Figure 5: Example of omitting root and lower-level nodes in a chained message

NOTE. *Never precede a star (*) command with a semicolon or colon.*

Termination

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

The instrument also allows white space before the terminator. For example, it allows CR LF.

Parameter Types

Every parameter in the command and query descriptions is of a specified type. The parameters are enclosed in brackets, such as <file_name>. The parameter type is listed after the parameter and is enclosed in parentheses, for example, (Boolean). Some parameter types are defined specifically for the arbitrary waveform generator command set and some are defined by SCPI.

Parameter types used in syntax descriptions

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

Block Arguments

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

Block symbols and their meanings

Symbol	Meaning
<NZDig>	A nonzero digit character in the range of 1 through 9
<Dig>	A digit character, in the range of 0 through 9
<DChar>	A character with the hexadecimal equivalent of 00 through FF (0 through 255 decimal) that represents actual data

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

Arbitrary Block Arguments

An arbitrary block argument is defined as:

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

or

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

- <NZDig> specifies the number of <Dig> elements that follow. Taken together, the <NZDig> and <Dig> elements form a decimal integer that specifies how many <DChar> elements follow.
- #0 means that the <Block> is an indefinite length block.
- The <terminator> ends the block.

NOTE. *The AWG4162 does not support the indefinite format (a block starts with #0).*

Quoted Strings

Some commands accept or return data in the form of a quoted string, which is 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 shown in the following table.

String and symbol 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:

- Use the same type of quote character to open and close the string. For example: "this is a valid string".
- You can mix quotation marks within a string as long as you follow the previous rule. For example, "this is an 'acceptable' string".
- You can include a quote character within a string by repeating the quote. For example: "here is a " " mark".
- Strings can have uppercase or lowercase characters.
- 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.

Here are some invalid strings:

- "Invalid string argument' (quotes are not of the same type)
- "test<EOI>" (termination character is embedded in the string)

Unit and SI Prefixes

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 International System of Units 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.

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

Example units

Symbol	Meaning
Hz	hertz (frequency)
V	voltage (V)

The SI prefixes, which must be included, are shown in the following table. You can enter both uppercase and lowercase characters.

SI prefixes and their indexes

SI prefix[1]	Z	A	F	P	N	U[2]	M	K	MA[3]	G	T	PE	EX
Corresponding power	10^{-21}	10^{-18}	10^{-15}	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{+3}	10^{+6}	10^{+9}	10^{+12}	10^{+15}	10^{+18}

1. The prefix m/M indicates 10^{-3} when the decimal numeric argument denotes voltage or time, but indicates 10^{6} when it denotes frequency.
2. The prefix u/U is used instead of "μ".
3. When the unit is "Hz", "M" may be used instead of "MA" so that the frequency can be represented by "MHz".

You can omit a unit in a command, but you must include the unit when using an SI prefix. For example, frequency of 15 MHz can be described as follows:

15.0E6, 1.5E7Hz, 15000000, 15000000Hz, 15MHz

("15M" is not allowed.)

NOTE. *You can use either lowercase or uppercase unit prefixes. The following examples have the same result, respectively.*

170mHz, 170mHz, 170MHz
250mv, 250mV, 250MV

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

General Rules for Using SCPI Commands

The following are three general rules for using SCPI commands, queries, and parameters:

- You can use single (' ') or double (" ") quotation marks for quoted strings, but you cannot use both types of quotation marks for the same string.

Correct: "This string uses quotation marks correctly."

Correct: 'This string also uses quotation marks correctly.'

Incorrect: "This string does not use quotation marks correctly.'

- You can use uppercase, lowercase, or a mixture of both cases for all commands, queries, and parameters.

```
:SOURCE:FREQUENCY 10MHZ
```

is the same as

```
 :source:frequency 100mhz
```

and

```
SOURCE:frequency 10MHZ
```

NOTE. *Literal strings (quoted) are case-sensitive. For example, file names are case-sensitive.*

- No embedded spaces are allowed between or within nodes.

Correct `:OUTPUT:FILTER:LPASS:FREQUENCY 200MHZ`

Incorrect `:OUTPUT: FILTER: LPASS:FREQUENCY 200MHZ`

Command Groups

This section lists the commands organized by functional group. The Command Descriptions section lists all commands alphabetically.

Calibration and Diagnostic Commands

Use the following calibration commands to calibrate and diagnose the arbitrary waveform generator.

Command	Description
*CAL? (on page 3-13)	Does an internal calibration of the arbitrary waveform generator and returns a status that indicates whether the calibration was completed successfully.
CALibration[:ALL] (on page 3-14)	Does a full calibration of the arbitrary waveform generator. The query form does a full calibration and returns a status indicating the success or failure of the operation.
DIAGnostic:DATA? (on page 3-15)	Returns the results of a self-test.
DIAGnostic[:IMMediate] (on page 3-15)	Executes the selected self-test routines.

Control Commands

Use the following commands to control operating modes.

Control commands

Command	Description
AWGControl:APPLication:RUN (on page 3-3)	Executes the specified application.
AWGControl:APPLication:STATe? (on page 3-3)	Returns the running state of the specified application.
AWGControl[:CHANnel[n]]:FUNCTionality (on page 3-4)	Sets or returns the channel [n] functionality.
AWGControl[:CHANnel[n]]:RSTate? (on page 3-5)	Returns the state of the arbitrary waveform generator or sequencer on channel [n].
AWGControl:CONFigure:CNUMber? (on page 3-6)	Returns the number of channels available on the instrument.
AWGControl:CONFigure:SEQuencer:MODE (on page 3-6)	Sets or returns sequencer mode.
AWGControl:EVENt:DJUMp:DEFine (on page 3-7)	Associates an event pattern with the jump target for Dynamic Jump.
AWGControl:EVENt:JMODe (on page 3-8)	Sets or returns the event jump mode.
AWGControl:EVENt:TABLe[:IMMediate] (on page 3-8)	Generates an event forcibly in the table jump mode.
AWGControl:RMODe (on page 3-9)	Sets or returns the run mode of the arbitrary waveform generator.
AWGControl:RUN[:IMMediate] (on page 3-10)	Initiates the output of a waveform or a sequence.
AWGControl:SNAMe? (on page 3-10)	Returns the current setup file name of the arbitrary waveform generator.
AWGControl:SREStore (on page 3-11)	Restores the arbitrary waveform generator's settings from a specified settings file.
AWGControl:SSAVe (on page 3-11)	Saves the arbitrary waveform generator's settings to a specified settings file.

Command	Description
AWGControl:STOP[:IMMediate] (on page 3-12)	Stops the output of a waveform or a sequence.
AWGControl:UPDATEdata (on page 3-12)	Uploads all data and settings to the instrument.

Event Commands

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

Command	Description
EVENTS:TRIGINTHReshold (on page 3-17)	Sets or returns the trigger event threshold.
EVENTS:TRIGINTIMERPeriod (on page 3-16)	Sets or returns the trigger event timer.

Instrument Commands

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

Command	Description
EVENTS:TRIGINTHReshold (on page 3-17)	Sets or returns the trigger event threshold.
EVENTS:TRIGINTIMERPeriod (on page 3-16)	Sets or returns the trigger event timer.

Mass Memory Commands

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

Command	Description
MMEMory:CATalog? (on page 3-19)	Returns the current contents and state of the mass storage media.
MMEMory:CDIRectory (on page 3-20)	Sets or returns the current directory of the file system on the arbitrary waveform generator.
MMEMory:DATA (on page 3-21)	Stores or returns block data to/from the file in the current mass storage device.
MMEMory:DELete (on page 3-21)	Deletes a file or directory from the instrument's hard disk.
MMEMory:EXPort (on page 3-22)	Exports a waveform file from the arbitrary waveform generator setup.
MMEMory:IMPort (on page 3-23)	Imports a file into the arbitrary waveform generator's setup as a waveform.
MMEMory:MDIRectory (on page 3-24)	Creates a new directory in the current path on the mass storage system.
MMEMory:MSIS (on page 3-25)	Selects a mass storage device used by all MMEMory commands.

Output Commands

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

Command	Description
OUTPut[n][:STATe] (on page 3-27)	Sets or returns the output state of the arbitrary waveform generator.

Sequence Commands

Use the following sequence commands to define and edit a waveform sequence. For more information about using sequences, see Creating and Working with Sequences (on page 2-4).

Command	Description
SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO (on page 3-28)	Sets the GOTO address for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr (on page 3-29)	Sets the jump address for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent (on page 3-29)	Sets the jump event for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType (on page 3-30)	Sets the jump type for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions (on page 3-31)	Sets the repetitions for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence (on page 3-31)	Sets or returns the subsequence for a sequence element [i].
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent (on page 3-32)	Sets the wait event for sequence element [i] of channel [n].
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform (on page 3-33)	Sets the waveform for sequence element [i] of channel [n].
SEQuence[:CHANnel[n]]:LENGth (on page 3-33)	Returns or sets the sequencer size for channel [n].

Source Commands

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

Command	Description
ANALOGCHANnel[n]:EVENTS[i]:OPERator[j] (on page 3-1)	Sets the operator event for event [i], channel [n].
ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j] (on page 3-2)	Sets the source event for event [i], channel [n].
AWGControl[:CHANnel[n]]:WAITstate (on page 3-5)	Sets the output value for the waiting state.
SOURce[n]:AMPlitudescale[:IMMediate] (on page 3-43)	Sets or returns the amplitude scale channel [n].
SOURce[n]:ATTENuation (on page 3-44)	Sets or returns the programmable attenuation for channel [n].
SOURce[n]:ATTENuation:TYPe (on page 3-45)	Sets or returns the attenuator control type for channel [n].
SOURce[n]:DIGital:STATe (on page 3-47)	Enables or disables the digital output channels. The query form returns the state of the digital output channels (enabled or disabled).
SOURce[n]:DIGital:SKEW[:IMMediate] (on page 3-46)	Sets or returns the digital skew value in seconds.
SOURce[n]:MARKer:SKEW[:IMMediate] (on page 3-47)	Sets or returns the marker delay.
SOURce[n]:MARKer:TYPe (on page 3-48)	Sets or returns the marker output type (analog or digital)
SOURce[n]:MARKer:VOLTage[:LEVel][:IMMediate][:AMPLitude] (on page 3-49)	Sets or returns the marker output level in volts.
SOURce[n]:MARKer:VOLTage:SELection[:IMMediate] (on page 3-50)	Sets or returns the marker output type (analog or digital).
SOURce[n]:OFFSet[:IMMediate] (on page 3-50)	Sets or returns the offset value in volts.
SOURce[n]:OUTputtype (on page 3-51)	Sets or returns the output channel [n] type.

Command	Description
SOURce[n]:SKEW (on page 3-52)	Sets or returns the analog skew value in seconds.
SOURce[n]:VOCM[:IMMediate] (on page 3-52)	Sets or returns VOCM value in volts.
[SOURce[n]:]WAVeform (on page 3-53)	Sets or returns the output waveform from the current waveform list for each channel when Run Mode is not Sequence.
TIMING[n]:CLOCKSOUrce (on page 3-55)	Sets or returns the clock source for channel [n] in Hertz.
TIMING[n]:REFCLOCKSOUrce (on page 3-56)	Sets or returns the reference clock source for channel [n] in Hertz.
TIMING[n]:SAMPLEFReq (on page 3-57)	Sets or returns the sampling frequency for channel [n] in Hertz.

Status Commands

The external controller uses the status commands to coordinate operation between the arbitrary waveform generator and other devices on the bus.

Command	Description
*CLS (on page 3-14)	Clears all event registers and queues.

Subsequence Commands

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

Command	Description
SLISt[:CHANnel[n]]:NAME? (on page 3-34)	Returns the name of the subsequence corresponding to the specified index in the subsequence list.
SLISt[:CHANnel[n]]:SIZE? (on page 3-35)	Returns the size of the subsequence list on channel [n].
SLISt:SUBSequence:DELete (on page 3-41)	Deletes the subsequence from the currently loaded setup.
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO (on page 3-35)	Sets or queries the GOTO address for subsequence element [i] of channel [n].
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr (on page 3-36)	Sets or queries the jump address for <subseq_name> element [i] of channel [n].
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent (on page 3-37)	Sets or queries jump event for <subseq_name> element [i] of channel [n].
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType (on page 3-38)	Sets or queries jump type for subsequence element [i] of channel [n].
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions (on page 3-39)	Sets or gets repetitions for <subseq_name> element [i] of channel [n].
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent (on page 3-40)	Sets or queries wait event for <subseq_name> element [i]
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform (on page 3-41)	Sets or returns the waveform for an element of the subsequence.
SLISt:SUBSequence:LENGth (on page 3-42)	This command and query sets or returns the size of the subsequence.
SLISt:SUBSequence:NEW (on page 3-43)	Creates a new subsequence.

Synchronization Commands

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

Command	Description
*OPC (on page 3-26)	Ensures the completion of the first command before the second command is issued.

System Commands

Use the following system commands to control miscellaneous instrument functions.

Command	Description
*IDN? (on page 3-17)	Returns identification information for the arbitrary waveform generator.
*OPT? (on page 3-26)	Returns the AWG4162 installed options and its serial number.
*RST (on page 3-27)	Resets the arbitrary waveform generator to its default state.
SYSTem:ERRor[:NEXT]? (on page 3-54)	Retrieves and returns data from the error and event queues.
SYSTem:VERSion? (on page 3-55)	Returns the SCPI version number to which the command conforms.

Trigger Commands

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

Command	Description
*TRG (on page 3-58)	Generates a force trigger event.
ABORt (on page 3-1)	Releases the force trigger.
TRIGger[:SEQuence][:IMMediate] (on page 3-58)	Generates a force trigger event.

Waveform Commands

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

Command	Description
WLISt:LAST? (on page 3-59)	Returns the name of the most recently added waveform in the waveform list.
WLISt:LIST? (on page 3-60)	Returns the names of the waveforms in the waveform list.
WLISt:NAME? (on page 3-60)	Returns the waveform name of an element in the waveform list.
WLISt:SIZE? (on page 3-61)	Returns the size of the waveform list.
WLISt:WAVeform:DATA (on page 3-61)	Transfers waveform data (analog, markers, and digital) from the external controller into the waveform list or from the waveform list to the external control program.
WLISt:WAVeform:DELete (on page 3-62)	Deletes the waveform from the currently loaded setup.
WLISt:WAVeform:DIGITAL:DATA (on page 3-63)	Transfers waveform digital data from the external controller into the waveform list or from the waveform list to the external control program.
WLISt:WAVeform:GRANularity? (on page 3-64)	Returns the granularity of sample points required for a valid waveform.
WLISt:WAVeform:LENGth? (on page 3-64)	Returns the size of the waveform.
WLISt:WAVeform:LMAXimum? (on page 3-65)	Returns the maximum number of waveform sample points allowed.
WLISt:WAVeform:LMINimum? (on page 3-65)	Returns the minimum number of waveform sample points required for a valid waveform.
WLISt:WAVeform:NEW (on page 3-66)	Creates a new empty waveform in the waveform list of the current setup.

Command	Description
WLISt:WAVeform:NORMalize (on page 3-67)	Normalizes a waveform that exists in the waveform list of the current setup.
WLISt:WAVeform:PREDefined? (on page 3-67)	Returns true or false based on whether the waveform is predefined.
WLISt:WAVeform:RESAmple (on page 3-68)	Resamples a waveform that exists in the waveform list of the current setup.
WLISt:WAVeform:TYPE? (on page 3-69)	Returns the type of the waveform.

Channel Number Optional Commands

The following commands have an optional parameter for the channel number (CHANnel[n]):

- AWGControl[:CHANnel[n]]:FUNCTionality
- AWGControl[:CHANnel[n]]:WAITstate
- TIMING[n]:CLOCKSOUrce
- TIMING[n]:SAMPLEFReq
- TIMING[n]:REFCLOCKSOUrce
- ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j]
- ANALOGCHANnel[n]:EVENTS[i]:OPERator[j]
- SEQuence[:CHANnel[n]]:LENGth
- SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions
- SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent
- SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent
- SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO
- SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr
- SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType
- SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence
- SLIST[:CHANnel[n]]:SIZE?
- SLIST[:CHANnel[n]]:NAME?
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPEvent
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr
- SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType

As single sequencer, the instrument can be programmed without the optional parameter; in this case the Channel 1 settings for the commands in this list will be automatically copied to Channel 2 after sending the AWGControl:UPDATEdata command.

Waveform Data Format

Introduction

In remote, the AWG4162 works like one temporary project. It provides two ways to load waveform data into the waveform list in remote mode:

- **Transfer waveform data in chunks.** This is a convenient way to send waveform data from the external controller into the waveform list.
- **Import waveform files from the hardware disk on the AWG4162.** Put legal waveform files in a related directory at first.

Related samples are located in the AWG4162 setup directory "..\Tektronix\AWG4000 Advanced\SCPI."

Waveform Data Transfer

Byte Order During Transfer

Waveform data is always transferred in LSB first format.

Transferring Waveforms in Chunks

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

The `WLISt:WAVeform:DATA` command accepts parameters that make it possible for control programs to send data in chunks. 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.

Block Data Format

Block data is a transmission format that is suitable for the transmission of large amounts of data. A command using a block data parameter with a definite length has the following structure:

Example: HEADer:HEADer #45168xxxxxxxx

The hash symbol (#) introduces the data block. The next number indicates how many of the following digits describe the length of the data block. In the example above, the number 4 following the hash symbol (#) means that the four following digits indicate the length to be 5168 bytes. The data bytes follow. During the transmission of these data bytes, all End or other control signals are ignored until all bytes are transmitted.

Real Format

Waveforms in Real format retain normalized values as they are. The format for waveform analog data in Real format is IEEE754 single-precision.

NOTE. *This format is the same as the format used for retaining single-precision values on a normal PC.*

Waveform data in Real format is as follows:

Analog Waveform Sample - IEEE754 single-precision format (32 bits)																
byte offset 3				**byte offset 2**				**byte offset 1**				**byte offset 0**				
Digital Waveform Sample																
	byte offset 5								**byte offset 4**							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Low Speed mode – DPA/DPB	A15 / B15	A14 / B14	A13 / B13	A12 / B12	A11 / B11	A10 / B10	A9/ B9	A8/ B8	A7/ B7	A6/ B6	A5/ B5	A4/ B4	A3/ B3	A2/ B2	A1/ B1	A0/ M1 B0/ M2
High Speed mode – DPA/DPB	x	x	x	x	x	x	x	x	A7/ B7	A6/ B6	A5/ B5	A4/ B4	A3/ B3	A2/ B2	A1/ B1	A0/ M1 B0/ M2
Disabled – DPA/DPB	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	M1/ M2

Digital Data

The digital bus mode can be configured as: Low Speed, High Speed, or Disabled.

- **Low Speed Mode:** 16 bits are available on DPA and DPB. The digital outputs sampling rate is ¼ of the analog sampling rate, so the length of the digital samples must be ¼ of the analog waveform length. In Low Speed mode, one digital sample (Byte 4 and Byte 5) of every four analog samples is relevant; the other bytes will be ignored. You can repeat the sample or insert a 0 value.

Example: analog waveform length = 100 samples ➔ digital waveform length = 25 samples ➔ waveform data in Real format bytes length will be 6 × analog waveform length = 600 bytes ➔ Each "Mixed Sample" (analog + digital data) is made of 6 bytes, where the first four bytes represent the analog waveform and the last two bytes represent the digital waveform.

MSN = Mixed Sample Number
AN = Analog Sample Number
DN = Digital Sample Number

MS0 Byte0 A0	MS0 Byte1 A0	MS0 Byte2 A0	MS0 Byte3 A0	**MS0 Byte4 D0**	**MS0 Byte5 D0**
MS1 Byte0 A1	MS1 Byte1 A1	MS1 Byte2 A1	MS1 Byte3 A1	MS1 Byte4 (D0)	MS1 Byte5 (D0)
MS2 Byte0 A2	MS2 Byte1 A2	MS2 Byte2 A2	MS2 Byte3 A2	MS2 Byte4 (D0)	MS2 Byte5 (D0)
MS3 Byte0 A3	MS3 Byte1 A3	MS3 Byte2 A3	MS3 Byte3 A3	MS3 Byte4 (D0)	MS3 Byte5 (D0)
MS4 Byte0 A4	MS4 Byte1 A4	MS4 Byte2 A4	MS4 Byte3 A4	**MS4 Byte4 D1**	**MS4 Byte5 D1**

- **High Speed Mode:** 8 bits are available on DPA and DPB. The digital outputs sampling rate is ½ of the analog sampling rate, so the length of the digital samples must be ½ of the analog waveform length. In High Speed mode, one digital sample (Byte 4 and Byte 5) of every two analog samples is relevant; the other bytes will be ignored. You can repeat the sample or insert a 0 value.

Example: analog waveform length = 100 samples ➔ digital waveform length = 50 samples ➔ waveform data in Real format bytes length will be 6 × analog waveform length = 600 bytes ➔ Each "Mixed Sample" (analog + digital data) is made of 6 bytes, where the first four bytes represent the analog waveform and the last two bytes represent the digital waveform.

MSN = Mixed Sample Number
AN = Analog Sample Number
DN = Digital Sample Number

MS0 Byte0 A0	MS0 Byte1 A0	MS0 Byte2 A0	MS0 Byte3 A0	**MS0 Byte4 D0**	**MS0 Byte5 D0**
MS1 Byte0 A1	MS1 Byte1 A1	MS1 Byte2 A1	MS1 Byte3 A1	MS1 Byte4 (D0)	MS1 Byte5 (D0)
MS2 Byte0 A2	MS2 Byte1 A2	MS2 Byte2 A2	MS2 Byte3 A2	**MS2 Byte4 D1**	**MS2 Byte5 D1**
MS3 Byte0 A3	MS3 Byte1 A3	MS3 Byte2 A3	MS3 Byte3 A3	MS3 Byte4 (D1)	MS3 Byte5 (D1)

- **Disabled:** no digital outputs are available; the byte offset 4 Bit0 sets the value for Marker1 (Digital POD A) or Marker2 (Digital POD B).

NOTE. *If there is no digitial option, the marker is in high-speed mode. In addition, with option DO16, Marker 2 speed will follow the speed set for Marker 1.*

Granularity

Waveform length must be multiple of 64 (< 320 points) or 16 (≥ 320 points).

The minimum waveform length is 64 samples. You can also send the `WLISt:WAVeform:GRANularity?` command to query granularity.

Creating and Working with Sequences

The sequence commands provide a way to create and edit the waveform sequences in the instruments. When the instrument runs a sequence, it outputs the waveforms in the order defined in the sequence.

To run a sequence, the instrument must be first put in the Sequence mode. This can be done by using either the instrument interface or the `AWGControl:RMODe SEQuence` command. Once the instrument is in the Sequence mode, it uses either the hardware or the software sequencer to execute the sequence.

NOTE. *There is only one sequence defined for an instrument. This is common to all channels. The Multi-Sequencer is not available using the SCPI commands.*

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 shown in the following table.

Sequence element parameter name	Default value	Remote command to query or set the parameter
CH 1 Waveform	" "	SEQuence:CHANnel[n]:ELEMent[i]:WAVEform
CH 2 Waveform	" "	SEQuence:CHANnel[n]:ELEMent[i]:WAVEform
Iteration Count	1	SEQuence:CHANnel[n]:ELEMent[i]:REPetitions
Go to address	0	SEQuence:CHANnel[n]:ELEMent[i]:GOTO
Wait Event	NONE	SEQuence:CHANnel[n]:ELEMent[i]:WAITEvent
TYPE Event Jump target index	ASYNC	SEQuence:CHANnel[n]:ELEMent[i]:JUMPType
Jump Event	NONE	SEQuence:CHANnel[n]:ELEMent[i]:JUMPEvent
Jump Address	0	SEQuence:CHANnel[n]:ELEMent[i]:JUMPAddr

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

Advanced Command Descriptions

ABORt

Using this command is the equivalent of releasing the Force Trig button on the AWG4162 front panel. There is no query form of this command.

Group

Trigger

Related Commands

TRIGger[:SEQuence][:IMMediate] (on page 3-58)
*TRG (on page 3-58)

Syntax

```
ABORt
```

Arguments

None

Examples

```
ABORt
```

Releases the force trigger button.

ANALOGCHANnel[n]:EVENTS[i]:OPERator[j]

This command sets the operator event for event [i] and channel [n]. There is no query form of this command.

Note that:

- The value of ANALOGCHANnel indicates the channel number = 1|2
- The value of EVENTS indicates the event number = 0|1|2|3|4|5|6
- The value of OPERator indicates the operator number = 1|2|3

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

ANALOGCHANnel[n]EVENTS[i]:SOUrce[j] (on page 3-2)

Syntax

```
ANALOGCHANnel[n]:EVENTS[i]:OPERator[j]
```

Arguments

```
<type>={AND|OR|XOR|NAND|NOR|XNOR}
```

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Examples

```
ANALOGCHANnel1:EVENTS2:OPERator2 AND
```

For channel 1, event 2, set operator 2 to AND.

ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j]

This command sets the source event for event [i] and channel [n]. There is no query form of this command.

Note that:

- The value of ANALOGCHANNEL indicates the channel number = 1|2
- The value of EVENTS indicates the event number = 0|1|2|3|4|5|6
- The value of SOUrce indicates the operator number = 1|2|3|4

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Source

Related Commands

ANALOGCHANnel[n]:EVENTS[i]:OPERator[j] (on page 3-1)

Syntax

```
ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j]
```

Arguments

```
<type>={FALSE|TRUE|TRGIN|TIMER|FORCETRIGger|MARKER1|NOTTRGIN|NOTTimer|
   NOTFORCETRIGger}
```

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Examples

```
ANALOGCHANnel1:EVENTS2:SOUrce4 FORCETRIGger
```

Set the Channel 1, Source 4 event to force trigger.

AWGControl:APPLication:RUN

This command executes the specified application. There is no query form of this command.

Group

Control

Related Commands

AWGControl:APPLication:STATe? (on page 3-3)

Syntax

```
AWGControl:APPLication:RUN <application_name>
```

Arguments

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

Specifies the application to be executed.

Examples

```
AWGCONTROL:APPLICATION:RUN “RFXPRESS”
```

Runs the RFXPRESS application.

AWGControl:APPLication:STATe?

This command returns the running state of the specified application. This command is query only.

Group

Control

Related Commands

AWGControl:APPLication:RUN (on page 3-3)

Syntax

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

Arguments

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

Returns

```
<Boolean>
```

- 0 indicates False
- 1 indicates True

Examples

```
AWGCONTROL:APPLICATION:STATE? “RFXPRESS”
```

Query the state of the RFXPRESS application.

Example return: 1

AWGControl[:CHANnel[n]]:FUNCTionality

This command sets or returns the channel functionality. [n] is the channel number = 1|2.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Control

Related Commands

None

Syntax

```
AWGControl[:CHANnel[n]]:FUNCTionality <funct_an>, <funct_dig>
AWGControl[:CHANnel[n]]:FUNCTionality?
```

Arguments

```
<funct_an>,<funct_dig>
funct_an::={ARBitrary}
funct_dig::={NONE|HIGHSPeed|LOWSPeed}
```

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

```
<funct_an>, <funct_dig>
funct_an::={ARBitrary}
funct_dig::={NONE|HIGHSPeed|LOWSPeed}
```

Examples

```
AWGControl:CHANnel1:FUNCTionality ARBitrary,LOWSPeed
```

Set channel 1 in arbitrary mode and digital mode to low speed.

```
AWGControl:FUNCTionality ARBitrary,LOWSPeed
```

Set channel 1 and channel 2 in arbitrary mode and digital mode to low speed.

AWGControl[:CHANnel[n]]:RSTate?

This command returns the state of the arbitrary waveform generator or sequencer. This command is query only.

Note that the value of CHANnel indicates the channel number = 1|2.

Group

Control

Related Commands

None

Syntax

```
AWGControl[:CHANnel[n]]:RSTate?
```

Arguments

None

Returns

<NR1>

- 0 indicates that the instrument has stopped.
- 1 indicates that the instrument is waiting for a trigger.
- 2 indicates that the instrument is running.

Examples

```
AWGControl:CHANnel1:RSTate?
```

Example return: 2, which indicates that channel 1 is running.

AWGControl[:CHANnel[n]]:WAITstate

This command sets output value for the waiting state. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

None

Syntax

```
AWGControl[:CHANnel[n]]:WAITstate {FIRSTsample|LASTsample}
```

Arguments

<type>:={FIRSTsample|LASTsample}

- LAST: Output state on the last sample of the previous waveform
- FIRST: Output state on the first sample of the new waveform

Available on sequence or triggered mode only.

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Examples

```
AWGControl:CHANnel1:WAITstate FIRSTsample
```

Set the Channel 1 WAITstate output to the first sample of the new waveform.

AWGControl:CONFigure:CNUMber?

This command returns the number of available channels on the instrument. It returns the count of channels even when they are disabled. This command is query only.

Group

Control

Related Commands

None

Syntax

```
AWGControl:CONFigure:CNUMber?
```

Returns

```
<NR1>
```

Examples

```
AWGCONTROL:CONFIGURE:CNUMBER?
```

Example return: 2.

AWGControl:CONFigure:SEQuencer:MODE

This command sets or returns sequencer mode.

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Control

Related Commands

None

Syntax

```
AWGControl:CONFigure:SEQuencer:MODE {SINgle}
AWGControl:CONFigure:SEQuencer:MODE?
```

Arguments

`SINgle` sets the AWG4162 in single sequencer mode.

Returns

SIN

Examples

AWGControl:CONFigure:SEQuencer:MODE SINgle

Sets the AWG4162 to single sequencer mode.

AWGControl:CONFigure:SEQuencer:MODE?

Example return: SIN.

AWGControl:EVENt:DJUMp:DEFine

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

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Control

Related Commands

None

Syntax

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

Arguments

event_pattern::=<NR1>. Values range from 0 to 255. This parameter specifies the event pattern to make an event jump. The event bits are mapped to the integer value as follows:

Event bits MSB 76543210 LSB

jump_target::=<NR1>. Values range from 1 to the maximum sequence length. This parameter specifies the sequence index as the jump target. Default value = 1; at *RST, all definitions are cancelled. Refer to *RST (on page 3-27).

Examples

AWGControl:EVENt:DJUMp:DEFine 15,3

Sets the jump target index to the third sequence element for the event pattern 00001111.

AWGControl:EVENt:DJUMp:DEFine? 15

Example return: 3.

AWGControl:EVENt:JMODe

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Control

Related Commands

None

Syntax

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

Arguments

`<jump_mode>::={EJUMp|DJUMp}`

EJUMp sets the jump mode to Event Jump. The jump targets defined in the sequence definition table will be used as the jump target.

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

At *RST, this returns EJUMp. Refer to *RST (on page 3-27).

Examples

```
AWGControl:EVENt:JMODe DJUMP
```

Sets the jump mode to Dynamic Jump.

AWGControl:EVENt:TABLe[:IMMediate]

This command generates an event forcibly in the table jump mode. There is no query form of this command.

Group

Control

Related Commands

None

Syntax

```
AWGControl:EVENt:TABLe[:IMMediate][<table_entry>]
```

Arguments

```
<table_entry>
<table_entry>::=<NR1>
```

Examples

```
AWGCONTROL:EVENT:TABLE 10
```

Strobe the 10th entry of the dynamic jump table.

AWGControl:RMODe

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Control

Related Commands

AWGControl:RUN[:IMMediate] (on page 3-10)
AWGControl:STOP[:IMMediate] (on page 3-12)

Syntax

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

Arguments

- `CONTinuous` sets the run mode to Continuous.
- `TRIGgered` sets the run mode to Triggered.
- `GATed` sets the run mode to Gated.
- `SEQuence` sets the run mode to Sequence.

At *RST, this value is SEQuence. Refer to *RST (on page 3-27).

The following table lists the run modes and their descriptions:

Argument	Description
`CONTinuous`	Selects the continuous mode, which continuously outputs the waveform. External triggers, including the FORCE TRIGGER button and the corresponding remote commands, have 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 when a start event is received and stops the output when a stop event is received.
`SEQuence`	Selects the sequence mode, which outputs the waveform according to the sequence.

Returns

CONT|TRIG|GAT|SEQ

Examples

AWGControl:RMODe TRIGgered

Sets the instrument run mode to Triggered.

AWGControl:RMODe?

Returns CONT if the instrument is in continuous mode.

AWGControl:RUN[:IMMediate]

This command initiates the output of a waveform or a sequence. This is equivalent to pressing the Run/Stop button on the front panel. The instrument can be put in the run state only when output waveforms are assigned to channels. There is no query form of this command.

Group

Control

Related Commands

AWGControl:STOP[:IMMediate] (on page 3-12)
[SOURce[n]:]WAVeform (on page 3-53)

Syntax

AWGControl:RUN[:IMMediate]

Examples

AWGCONTROL:RUN

Put the instrument in the run state.

AWGControl:SNAMe?

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

Group

Control

Related Commands

AWGControl:SSAVe (on page 3-11)
AWGControl:SREStore (on page 3-11)

Syntax

AWGControl:SNAMe?

Returns

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

Examples

```
AWGCONTROL:SNAME?
```

Query the current setup name.

Example return:

\\temp\a1.awg ,D:

AWGControl:SREStore

This command restores the arbitrary waveform generator's settings 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. There is no query form of this command.

Group

Control

Related Commands

AWGControl:SNAMe? (on page 3-10)
AWGControl:SSAVe (on page 3-11)

Syntax

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

Arguments

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

Examples

```
AWGControl:SREStore "\temp\x.awg", "D:"
```

AWGControl:SSAVe

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. There is no query form of this command.

Group

Control

Related Commands

AWGControl:SREStore (on page 3-11)
AWGControl:SNAMe? (on page 3-10)

Syntax

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

Arguments

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

Examples

```
AWGControl:SSAVE "\temp\x.awg ", "D:"
```

Save the current setup to "D:\temp\x.awg".

AWGControl:STOP[:IMMediate]

This command stops the output of a waveform or a sequence. There is no query form of this command.

Group

Control

Related Commands

AWGControl:RUN[:IMMediate] (on page 3-10)

Syntax

```
AWGControl:STOP[:IMMediate]
```

Examples

```
AWGControl:STOP:IMM
```

Stops the output of a waveform.

AWGControl:UPDATEdata

This command uploads all data and settings to the instrument. There is no query form of this command.

Group

Control

Related Commands

None

Syntax

```
AWGControl:UPDATEdata [{ALL|SET|DATA}]
```

Arguments

<type>:={ALL|SET|DATA}

- ALL: Uploads instruments settings, waveform, and sequencer data
- SET: Uploads instruments settings
- DATA: Uploads waveform and sequencer data

If <type> is omitted, the command will upload both settings and data into the instrument.

Examples

```
AWGControl:UPDATEdata
```

*CAL?

This command does a self-calibration of the arbitrary waveform generator and returns a status that indicates whether the calibration was completed successfully.

Group

Calibration

Related Commands

CALibration[:ALL]? (on page 3-14)

Syntax

```
*CAL?
```

Arguments

None

Returns

<NR1>

Examples

```
*CAL?
```

Performs a calibration and returns results.

Example return: 0, which indicates that the calibration completed without any errors.

-315 (Calibration Error) indicates that the calibration failed.

CALibration[:ALL]

This command does a self-calibration of the arbitrary waveform generator. In its query form, the command does a self-calibration and returns a status indicating the success or failure of the operation.

Group

Calibration

Related Commands

*CAL? (on page 3-13)

Syntax

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

Arguments

None

Returns

<NR1>

Examples

```
CALibration?
```

0 indicates no error.

-315 (Calibration Error) indicates that the calibration failed.

*CLS

This command clears all the event registers and queues that are used in the arbitrary waveform generator status and event reporting system. There is no query form of this command.

Group

Status

Related Commands

None

Syntax

```
*CLS
```

Arguments

None

Examples

```
*CLS
```

Clear all the event registers and queues.

DIAGnostic:DATA?

This command returns the result of the last executed self-test. If no self-test is executed, DIAGNOSTIC:IMMEDIATE? returns default 0. This command is query only.

Group

Diagnostic

Related Commands

DIAGnostic[:IMMediate] (on page 3-15)

Syntax

```
DIAGnostic:DATA?
```

Arguments

None

Returns

<NR1>

Examples

```
DIAGnostic:DATA?
```

0 indicates no error.

-314 (Diagnostic Error) indicates that the self-test failed.

DIAGnostic[:IMMediate]

This command executes the selected self-test routines.

Group

Diagnostic

Related Commands

DIAGnostic:DATA? (on page 3-15)

Syntax

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

Arguments

None

Returns

<NR1>

Examples

DIAGNOSTIC:IMMEDIATE

Executes the self-test routines.

DIAGNOSTIC:IMMEDIATE?

Returns the result of the last executed self-test.

Sending DIAGNOSTIC:IMMEDIATE? has the same result as sending DIAGnostic:DATA?.

0 indicates no error.

EVENTS:TRIGINTIMERPeriod

This command sets or returns the trigger-in event timer.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Event

Related Commands

None

Syntax

EVENTS:TRIGINTHReshold <NR3>
EVENTS:TRIGINTHReshold?

Arguments

<NR3>

Range is between 1 and 2.684354M µs.

Returns

<NR3>

Examples

EVENTS:TRIGINTIMERPeriod 1

Sets the timer period at 1 µs.

EVENTS:TRIGINTHReshold

This command sets or returns trigger-in event threshold.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Event

Related Commands

None

Syntax

```
EVENTS:TRIGINTHReshold <NR2>
EVENTS:TRIGINTHReshold?
```

Arguments

`<NR2>`

Range is between -10 V to +10 V

Returns

`<NR2>`

Examples

```
EVENTS:TRIGINTHReshold 5.5V
```

Sets the input threshold level to 5.5 V.

*IDN?

This command returns identification information for the arbitrary waveform generator. This command is query only.

Group

System

Related Commands

None

Syntax

```
*IDN?
```

Arguments

None

Returns

```
<Manufacturer>,<Model>,<Application>,<Serial Number>,<SCPI
   Version>,<Firmware Version>
```

Where:

```
<Manufacturer>::=TEKTRONIX
<Model>::={AWG4162}
<Application>::{ADVANCED}
<Serial Number>::{XXXXXXX} (indicates an actual serial number)
<SCPI Version>::=SCPI:99.0 FV:1.0
<Firmware Version>::=FV:x.x.x.x (x.x.x. is system software version)
```

Examples

```
*IDN?
```

Example return:

TEKTRONIX, MWG4062ADVANCED,C000003,SCPI:99.0,FV:1.6.3.3

INSTrument:COUPle:SOURce

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

NOTE. *When coupling is done, CH1 can be coupled to CH2. When ALL is used, all other channels get the parameters of CH1. When coupling is done, CH1 parameters are copied to CH2 parameters. This cannot be changed. On two-channel models, ALL is equivalent to PAIR.*

The parameters that will be coupled are:

- VOCM
- offset
- amplitude scale
- output type
- marker output level
- attenuation type
- attenuation value

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Instrument

Related Commands

None

Syntax

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

Arguments

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

PAIR – CH1 to CH2

ALL – CH1 to CH2

Returns

<state>

Examples

INSTRUMENT:COUPLE:SOURCE ALL

Couples the channel 1 (CH1) parameters and Channel 2 (CH2) parameters if the instrument is a two-channel model.

MMEMory:CATalog?

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

Group

Mass Memory

Related Commands

MMEMory:CDIRectory (on page 3-20)
MMEMory:MSIS (on page 3-25)

Syntax

MMEMory:CATalog? [<msus>]

Arguments

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

NOTE. *This is an absolute path.*

Returns

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

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

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

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

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

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

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

Examples

```
MMEMORY:CATALOG? "D:\"
MMEMORY:CATALOG? "D:\temp"
```

Example return:

484672,3878652, "SAMPLE1.AWG,2948"

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

MMEMory:CDIRectory

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

Group

Mass memory

Related Commands

None

Syntax

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

NOTE. *The command changes the directory in the current mass storage unit.*

Arguments

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

NOTE. *This is an absolute path.*

Returns

```
<directory_name>
```

It returns an absolute path.

Examples

```
MMEMory:CDIRectory "C:\"
MMEMory:CDIRectory "C:\temp"
```

Changes the current directory to "C:\temp".

MMEMory:DATA

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

- This command has a limit of 6,000,000 bytes of data.
- The query form of this command inserts the LF character ("0A" bytes) at the end of the transmission.

Group

Mass memory

Related Commands

MMEMory:CDIRectory (on page 3-20)
MMEMory:MSIS (on page 3-25)

Syntax

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

Arguments

`<file_name>,<block_data>`

NOTE. *<file_name> only contains the file name. The file is always transferred to or queried from the current path.*

Returns

`Block_data` – IEEE 488.2 data block

`file_name` – String containing a file name and path

Examples

```
MMEMORY:DATA "FILE1",#41024XXXXX...
```

Loads data into the file named FILE1.

MMEMory:DELete

This command deletes a file or directory from the instrument's hard disk. When used on a directory, this command succeeds only if the directory is empty. There is no query form of this command.

Group

Mass memory

Related Commands

MMEMory:CDIRectory (on page 3-20)
MMEMory:MSIS (on page 3-25)

Syntax

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

Arguments

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

NOTE. *<file_name> can be a relative path or absolute path.*

Examples

```
MMEM:DEL "SETUP1.AWG"
```

Deletes SETUP1.AWG in the current directory.

```
MMEM:DEL "\temp\FILE1", "D:"
```

Deletes D:\temp\FILE1.

MMEMory:EXPort

This command exports a waveform file from the arbitrary waveform generator setup. There is no query form of this command.

CAUTION. 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 format is:

- TXT – Text file with analog data

NOTE. *Mixed, analog, and digital waveforms can be exported. If the optional parameter is null, only the analog part will be exported.*

Group

Mass memory

Related Commands

MMEMory:IMPort (on page 3-23)

Syntax

```
MMEMory:EXPort <wfm_name>,<filename>,<type>[,<wfm_type>]
```

Arguments

```
<wfm_name>,<filename>,<type>
<wfm_name>::=<string>
<filename>::=<string>
```

NOTE. *<filename>::=<string> is an absolute path.*

```
<type> = {TXT}
<wfm_type>={MIXED,ANALOG,DIGITAL}
```

Examples

```
MMEMORY:EXPORT "sine1024", " D:\temp\sine1024.txt",TXT,ANALOG
```

Exports a waveform named "sine1024" to "D:\temp\sine1024.txt" with analog data.

MMEMory:IMPort

This command imports a file into the arbitrary waveform generator's setup as a waveform. There is no query form of this command.

CAUTION. If the waveform name already exists, it will be overwritten without warning. The file name can contain a path and drive letter.

The supported file formats are:

- TXT – Text file with analog/digital data
- WFM –Tekscope Series waveform (depends on the scope model)
- PAT – AWG Series pattern file (depends on the AWG model)
- TFW – AFG3000 Series waveform file format
- MAT – Matlab .mat file format. The Matlab file format needs to follow this format:

```
NumPoints = 2400; %Waveform length
† = (0:1:NumPoints-1)'; %Define † vector
waveform = single(sin(2*pi*1/NumPoints*†)); %Create single
sinewave
%% Save Waveform
Waveform_Name_1 = 'SINE'; %Name Waveform
Waveform_Data_1 = waveform; %Assign waveform data
Waveform_Sampling_Rate_1 = 2.4e9; %You can specify sample rate in
S/s
Waveform_Amplitude_1 = 0.300; %and amplitude in V
save('SingleCycleSine', '*_1', '-v7.3'); %Save all variables
ending in _1 to .mat file
```

- DIG – text file .txt (comma-separated value) to import the digital values only. The import/export file format for digital waveforms is a comma separated value file in which each column represents the samples of one digital channel.
- For the TXT type file, the first column is always imported into analog on one channel. If it has digital columns, they will be always imported from D0 on the same channel. For the DIG type file, the digital columns will be always imported from D0 on one channel.

NOTE. *When you import a DIG format file, the analog waveform values will be set to 0.*

Group

Mass memory

Related Commands

MMEMory:EXPort (on page 3-22)

Syntax

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

Arguments

```
<wfm_name>,<filename>,<type>
<wfm_name>::=<string>
<filename>::=<string>
```

NOTE. *<filename>::=<string> is an absolute path.*

```
<type> = {TXT|WFM|PAT|TFW|MAT|DIG}
```

Examples

```
MMEMORY:IMPORT "sine1024", "D:\temp\sine1024.txt",txt
```

Imports a waveform file named "sine1024".

MMEMory:MDIRectory

This command creates a new directory in the current path on the mass storage system. There is no query form of this command.

Group

Mass memory

Related Commands

MMEMory:CDIRectory (on page 3-20)
MMEMory:MSIS (on page 3-25)

Syntax

```
MMEMory:MDIRectory <directory_name>
```

NOTE. *The <directory_name> should be a relative path that is created starting from the selected current path.*

Arguments

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

NOTE. *<directory_name>::=<string> is a relative path.*

Examples

```
MMEMory:MDIRectory "WAVEFORM"
```

Creates the directory "WAVEFORM" in the current path.

```
MMEMory:MDIRectory "WAVEFORM\MIXED"
```

Creates the directory "MIXED" in the current path file folder named "WAVEFORM".

MMEMory:MSIS

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

Group

Mass memory

Related Commands

None

Syntax

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

Arguments

```
<msus>::=<string>
```

Where:

<msus> = mass storage unit specifier

Returns

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

Examples

```
MMEMory:MSIS?
```

Example return: "X:"

*OPC

This command is used to ensure that the first command is complete before the second command is issued. Always returns 1 on this instrument.

Group

Synchronization

Related Commands

None

Syntax

```
*OPC
*OPC?
```

Arguments

None

Returns

<NR1>

Where:

<NR1>=1 Indicates that all pending operations are complete.

Examples

```
*OPC?
```

Example return if all pending OPC operations are finished: 1.

*OPT?

This command returns a list of the options and installed memory in the arbitrary waveform generator. This command is query only.

Group

System

Related Commands

None

Syntax

```
*OPT?
```

Arguments

None

Returns

```
SERIAL_NUMBER: "XXXXXXXX";;MEMORY_OPT:<opt1>;;DIGITAL_OPT:<opt2>
<opt1>::= {1_MEG|16_MEG|32_MEG|64_MEG}
<opt2>::= {NO_DIGITAL|8_16_DIGITAL|16_32_DIGITAL}
```

Examples

*OPT?

Example return:

"C000003";;MEMORY_OPT:64_MEG;;DIGITAL_OPT:16_32_DIGITAL

OUTPut[n][:STATe]

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

Group

Output

Syntax

```
OUTPut[n][:STATe] {0|1|ON|OFF}
OUTPut[n][:STATe]?
```

Arguments

0|1|ON|OFF

The value of [n] indicates the output number.

Returns

0|1

Examples

OUTPut1:STATe ON

Set the arbitrary waveform generator channel 1 (CH 1) output to ON.

OUTPut1:STATE?

Example return: 1, which means the channel 1 output is ON.

*RST

This command resets the instrument to the factory default settings. Internal data structures such as waveform, sequencer, and settings are set to their default settings as soon as *RST is sent. There is no query form of this command.

New waveform, sequencer, and settings are not applied until an AWGControl:UPDATEdata command is received.

Group

System

Related Commands

None

Syntax

*RST

Arguments

None

Examples

*RST

Reset the arbitrary waveform generator settings to the factory defaults.

SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO

This command sets the GOTO address for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Sequence

Related Commands

None

Syntax

SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO <NR1>

Arguments

<NR1>

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Examples

SEQuence:ELEMent2:GOTO 34

Sets the GOTO address to 34 on channel 1 and channel 2 sequencer element 2.

SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr

This command sets the jump address for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr <NR1>
```

Arguments

`<NR1>`

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Examples

```
SEQuence:ELEMent3:JUMPAddr 23
```

Sets the jump address to 23 on channel 1 and channel 2 sequencer element 3.

SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent

This command sets the jump event for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent {0|1|2|3|4|5|6|NONE}
```

Arguments

<event_number>:={0|1|2|3|4|5|6|NONE}

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Examples

```
SEQuence:ELEMent2:JUMPEvent 4
```

Sets the jump event to 4 on channel 1 and channel 2 sequencer element 2.

SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType

This command sets the jump type for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence:[CHANnel[n]]:ELEMent[i]:JUMPType {ASYNC|SYNC}
```

Arguments

ASYNC|SYNC

- ASYNC: Asynchronous jump – the sequencer does not wait for the completion of the current waveform execution
- SYNC: Synchronous jump

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Returns

ASYNC|SYNC

Examples

```
SEQuence:ELEMent3:JUMPType ASYNC
```

Sets the jump type to ASYNC on channel 1 and channel 2 sequencer element 3.

SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions

This command sets repetitions for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions {INFinite|<NR1>}
```

Arguments

`INFinite|<NR1>`

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number. The minimum value of <NR1> is 1 and the maximum value is 2097151.

Examples

```
SEQuence:ELEMent5:REPetitions INFinite
```

Sets the repetitions to infinite on channel 1 and channel 2 sequencer element 5.

SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SEQuence[:CHANNEL[n]]:ELEMent[i]:SUBSequence <subseq_name>
SEQuence[:CHANNEL[n]]:ELEMent[i]:SUBSequence?
```

Arguments

<subseq_name>::=string

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Returns

<subseq_name>

Examples

SEQuence:ELEMent2:SUBSequence "SUB1"

Sets the channel 1 and channel 2 sequencer element 2 with subsequence SUB1.

SEQuence:ELEMent2:SUBSequence?

Example return: "SUB1"

SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent

This command sets the wait event for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Sequence

Related Commands

None

Syntax

SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent {0|1|2|3|4|5|6|NONE}

Arguments

<event_number>:={0|1|2|3|4|5|6|NONE}

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the element number.

Examples

SEQuence:ELEMent3:WAITEvent 2

Set channel 1 and channel 2 sequence element 3 wait event to 2.

SEQuence:CHANnel[n]:ELEMent[i]:WAVeform

This command sets the waveform for sequence element [i] of channel [n]. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform <waveform_name>
```

Arguments

```
<waveform_name>::=<string>
```

The value of [n] indicates the channel number.

The value of [i] indicates the subsequence element number.

NOTE. *The length of the waveforms specified for a sequence element on both channels must be equal.*

Examples

```
SEQuence:CHANnel1:ELEMent3:WAVeform "WFN1"
```

Sets the channel 1 sequencer element 3 to "WFN1".

SEQuence[:CHANnel[n]]:LENGth

This command returns or sets the sequencer size for channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Sequence

Related Commands

None

Syntax

```
SEQuence[:CHANnel[n]]:LENGth <NR1>
SEQuence[:CHANnel[n]]:LENGth?
```

Arguments

<NR1>

Where:

<NR1> = The sequence length. The maximum length is 16384.

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

<NR1>

Examples

SEQuence:LENGth 100

Sets the channel 1 and channel 2 sequencer length to 100 entries.

SEQuence:LENGth?

Example return: 100.

SLISt[:CHANnel[n]]:NAME?

This command returns the name of the subsequence corresponding to the specified index in the subsequence list. This command is query only.

Group

Subsequence

Related Commands

None

Syntax

SLISt[:CHANnel[n]]:NAME? <Index>

Arguments

<Index>::=<NR1>

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

<string>

Examples

SLISt:CHANnel:NAME? 3

Example return: "SUB1"

SLISt[:CHANnel[n]]:SIZE?

This command returns the size of the subsequence list on channel [n]. This command is query only.

Group

Subsequence

Related Commands

None

Syntax

```
SLISt[:CHANnel[n]]:SIZE?
```

Arguments

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

```
<NR1>
```

Examples

```
SLISt:CHANnel1:SIZE?
```

Example return: 1000

SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO

This command sets or queries the GOTO address for subsequence element [i] of channel [n].

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO <subseq_name>,<NR1>
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO? <subseq_name>
```

Arguments

```
<subseq_name>,<NR1>
<subseq_name>::=<string>
```

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [n] indicates the channel number.

Returns

<NR1>

Examples

```
SLISt:SUBSequence:ELEMent2:GOTO "SUB1", 20
```

Sets channel 1 and channel 2 subsequence "SUB1" element 2 GOTO address to 20.

```
SLISt:SUBSequence:ELEMent2:GOTO? "SUB1"
```

Example return: 20

SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr

This command sets or queries the jump address for <subseq_name> element [i] of channel [n].

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr <subseq_name>,<NR1>
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr? <subseq_name>
```

Arguments

```
<subseq_name>,<NR1>
<subseq_name>::=<string>
```

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the subsequence element number.

Returns

<NR1>

Examples

```
SLISt:SUBSequence:ELEMent2:JUMPAddr "SUB1", 5
```

Sets channel 1 and channel 2 subsequence "SUB1" element 2 jump address to 5.

```
SLISt:SUBSequence:ELEMent2:JUMPAddr? "SUB1"
```

Example return: 5

SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent

This command sets or queries the jump event for <subseq_name> element [i] for channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent
   <subseq_name>,<event_number>
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent? <subseq_name>
```

Arguments

```
<subseq_name>,<event_number>
<subseq_name>
<subseq_name>::=<string>
<event_number>:={0|1|2|3|4|5|6|NONE}
```

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the subsequence element number.

Returns

```
{0|1|2|3|4|5|6|NONE}
```

Examples

```
SLISt:SUBSequence:ELEMent2:JUMPEvent "SUB1", 3
```

Sets channel 1 and channel 2 subsequence "SUB1", element 2 jump event to 3.

```
SLISt:SUBSequence:ELEMent2:JUMPEvent? "SUB1"
```

Example return: 3, which indicates the element 2 jump event is set to event 3.

SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType

This command sets or queries the jump type for subsequence element [i] of channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType <subseq_name>,
   {ASYNC|SYNC}
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:JUMPType? <subseq_name>
```

Arguments

```
<subseq_name>, {ASYNC|SYNC}
<subseq_name>
<subseq_name>::<string>
ASYNC|SYNC
```

- ASYNC: Asynchronous jump. The sequencer does not wait for the completion of the current waveform execution.
- SYNC: Synchronous jump.

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the subsequence element number.

Returns

```
ASYNC|SYNC
```

Examples

```
SLISt:SUBSequence:ELEMent3:JUMPType "SUB1", ASYNC
```

Sets channel 1 and channel 2 subsequence "SUB1", element 3 jump type to asynchronous.

```
SLISt:SUBSequence:ELEMent4:JUMPType?
```

Example return: ASYNC

SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions

This command sets or gets the number of repetitions for <subseq_name> element [i] of channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[n]:REPetitions
   <subseq_name>,{INFinite|<NR1>}
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[n]:REPetitions? <subseq_name>
```

Arguments

```
<subseq_name>,{INFinite|<NR1>}
<subseq_name>::=<string>
```

`INFinite` sets the subsequence to repeat indefinitely.

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the subsequence element number.

Returns

```
INFinite|<NR1>
```

Examples

```
SLISt:SUBSequence:ELEMent2:REPetitions "SUB1", 5
```

Sets channel 1 and channel 2, subsequence "SUB1", element 2 to 5 repetitions.

```
SLISt:SUBSequence:ELEMent3:REPetitions? "SUB1"
```

Example return: 5

SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent

This command sets or queries the wait event for <subseq_name> element [i].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent
   <subseq_name>,<event_number>
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:WAITEvent? <subseq_name>
```

Arguments

```
<subseq_name>,{0|1|2|3|4|5|6|NONE}
<subseq_name>
<subseq_name>::=<string>
```

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

The value of [i] indicates the subsequence element number.

Returns

```
0|1|2|3|4|5|6|NONE
```

Examples

```
SLISt:SUBSequence:ELEMent2:WAITEvent "SUBS1", 2
```

Sets channel 1 and channel 2 subsequence "SUBS1", element 2 wait event to 2.

```
SLISt:SUBSequence:ELEMent2:WAITEvent? "SUBS1"
```

Example return: 2

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

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform
   <subseq_name>,<wfm_name>
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform? <subseq_name>
```

Arguments

```
<subseq_name>,<wfm_name>
<subseq_name>::=<string>
<wfm_name>::=<string>
```

The value of [n] indicates the channel number.

The value of [i] indicates the subsequence element number.

NOTE. *In Single Sequencer mode, the CHANnel[n] part of the command should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

```
<wfm_name>
```

Examples

```
SLISt:SUBSequence:CHANnel1:ELEMent2:WAVeform "TEST", "WF1"
```

Sets channel 1 subsequence "TEST" element 2 to use waveform "WF1".

```
SLISt:SUBSequence:CHANnel1:ELEMent2:WAVeform? "TEST"
```

Example return: "WF1"

SLISt:SUBSequence:DELete

This command deletes the subsequence from the currently loaded setup. There is no query form of this command.

Group

Subsequence

Related Commands

None

Syntax

```
SLISt:SUBSequence:DELete {<subseq_name>|ALL}
```

Arguments

```
<subseq_name>|ALL
<subseq_name>::=<string>
```

ALL: Delete all the subsequences.

Examples

```
SLISt:SUBSequence:DELete "SUB1"
```

Deletes subsequence "SUB1" from both channels of the sequencer.

SLISt:SUBSequence:LENGth

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

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

Arguments

```
<subseq_name>,<NR1>
<subseq_name>
<subseq_name>::=<string>
```

Returns

```
<NR1>
```

Examples

```
SLISt:SUBSequence:LENGth "Test",1000
```

Sets the subsequence named "Test" to a length of 1000 samples for both channels.

```
SLISt:SUBSequence:LENGth? "Test"
```

Example return: 1000

SLISt:SUBSequence:NEW

This command creates a new subsequence. There is no query form of this command.

> ***NOTE.*** *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Subsequence

Related Commands

None

Syntax

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

Arguments

```
<subseq_name>,<length>
<subseq_name>::=<string>
<length>::=<NR1>
```

Returns

None

Examples

```
SLISt:SUBSequence:NEW "test",64
```

Creates a subsequence named "test" with a length of 64 samples on both channels.

SOURce[n]:AMPlitudescale[:IMMediate]

This command sets or returns the amplitude scale of channel [n].

> ***NOTE.*** *This command can be used only after the* `AWGControl:UPDATEdata` *command is called successfully. The channels must be initialized correctly before sending this command. If they are not initialized first, you will get an error, "No channels available."*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:AMPlitudescale <NR2>
SOURce[n]:AMPlitudescale?
```

Arguments

<NR2>

The value of [n] indicates the channel number.

The minimum value is 0 and the maximum value is 200.

Returns

<NR2>

Examples

SOURce1:AMPlitudescale 95

Sets the channel 1 amplitude scale to 95%.

SOURce1:AMPlitudescale?

Example return: 95

SOURce[n]:ATTENuation

This command and query sets or returns the programmable attenuation for channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:ATTENuation <NR2>
SOURce[n]:ATTENuation?
```

Arguments

- [n] is the channel number = 1 | 2
- <NR2> range is 0.0 to 40

Returns

<NR2>

Examples

SOURce1:ATTENuation 3.5

Set 3.5 dB attenuation on channel 1.

SOURce1:ATTENuation?

Example return: 3.5

SOURce[n]:ATTENuation:TYPe

This command and query sets or returns the attenuator control type for channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:ATTENuation {MANual,AUTO}
SOURce[n]:ATTENuation?
```

Arguments

- [n] is the channel number = 1 | 2
- `MANual` = manual attenuator control
- `AUTO` = automatic attenuator control

Returns

`<NR2>`

Examples

`SOURce1:ATTENuation:TYPe MAN`

Set the attenuator control to manual.

`SOURce1:ATTENuation?`

Example return: MAN

SOURce[n]:DAC:RESolution?

This command returns the DAC resolution. This command is query only.

NOTE. *DAC supports 14-bit resolution only.*

Group

Source

Related Commands

None

Syntax

`[SOURce[n]]:DAC:RESolution?`

Return is 14.

Arguments

None

Returns

<NR1>

Examples

SOURce1:DAC:RESOLUTION?

Return: 14, which means that the channel 1 resolution is 14 bits.

SOURce[n]:DIGital:SKEW[:IMMediate]

This command sets or returns the digital skew value in seconds.

Group

Source

Related Commands

None

Syntax

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

NOTE. *This command can be used only after the* `AWGControl:UPDATEdata` *command is called successfully.*

Arguments

<NR3>

The value of [n] indicates the channel number.

Returns

<NR3>

Examples

SOURce1:DIGital:SKEW
SOURce1:DIGital:SKEW 1E-9

Sets channel 1 skew to 1 ns.

SOURce1:DIGital:SKEW?

Example return: 1E-9

SOURce[n]:DIGital:STATe

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

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Output

Related Commands

None

Syntax

```
SOURce[n]DIGital:STATe {0|1|OFF|ON}
SOURce[n]DIGital:STATe
```

Arguments

`0|1|OFF|ON`

The value of [n] indicates the output number.

Returns

`0|1`

Examples

`SOURce1:DIGital:STATe ON`

Set the digital output of source 1 to the ON state.

`SOURce1:DIGital:STATe?`

Example return: 1, which means that the source 1 digital output is set to ON.

SOURce[n]:MARKer:SKEW[:IMMediate]

This command and query sets or returns the marker delay.

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:MARKer:SKEW <NR3>
SOURce[n]:MARKer:SKEW?
```

NOTE. *This command can be used only after the* `AWGControl:UPDATEdata` *command is called successfully.*

Arguments

<NR3>

The value of [n] indicates the channel number.

Returns

<NR3>

Examples

SOURce1:MARKer:SKEW 1E-9

Sets the marker delay to 1 ns.

SOURce1:MARKer:SKEW?

Example return: 1E-3

SOURce[n]:MARKer:TYPe

This command and query sets or returns the marker type for channel [n].

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:MARKer:TYPe{ANalog,DIGital}
SOURce[n]:MARKer:TYPe?
```

Arguments

[n] is the channel number = 1 | 2

- `ANalog` = the marker output signal is connected to the front panel SMA connector.
- `DIGital` = the marker output signal is connected to the front panel Digital connector

NOTE. *The maximum update rate for the analog marker output is 156.25 MHz.*

Returns

<NR2>

Examples

SOURce1:MARKer:TYPe AN

Set the marker type to analog.

SOURce1:MARKer:TYPe?

Example return: AN

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

This command sets or returns the marker output level in volts.

Group

Source

Related Commands

None

Syntax

SOURce[n]:MARKer:VOLTage <NR2>

Arguments

The marker voltage is calculated on a 50 Ω load. The minimum value is 1 V at a 50 Ω load and the maximum value is 2.5 V at a 50 Ω load.

The value of [n] indicates the channel number.

Returns

<NR2>

Examples

SOURce1:MARKer:VOLTage 1.5

Set the source 1 output marker level to 1.5 V.

SOURce1:MARKer:VOLTage?

Example return: 1.5

SOURce[n]:MARKer:VOLTage:SELection[:IMMediate]

This command sets or returns the marker out selection.

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:MARKer:VOLTage:SELection {LINE|LOW|HIGH}
SOURce[n]:MARKer:VOLTage:SELection?
```

Arguments

`LINE|LOW|HIGH`

- `LINE`: Link the marker out signal to Probe A / Probe B digital line 0 – A(0),B(0)
- `LOW`: Set the marker level to low
- `HIGH`: Set the marker level to high

The value of [n] indicates the channel number.

Returns

`LINE|LOW|HIGH`

Examples

```
SOURce1:MARKer:VOLTage:SELection LINE
```

Sets the channel 1 marker voltage selection to LINE.

```
SOURce1:MARKer:VOLTage:SELection?
```

Example return: LINE

SOURce[n]:OFFSet[:IMMediate]

This command sets or returns offset value in volts.

NOTE. *This command can be used only after the `AWGControl:UPDATEdata` command is called successfully. The channels must be initialized correctly before sending this command. If they are not initialized first, you will get an error, "No channels available."*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:OFFSet <NR3>
SOURce[n]:OFFSet?
```

Arguments

<NR3>

The value of [n] indicates the channel number.

NOTE. *Offset value has no effect on AC coupled output.*

Returns

<NR3>

Examples

SOURce1:OFFSet 0.5

Sets channel 1 offset to 0.5 V.

SOURce1:OFFSet?

Example return: 0.5

SOURce[n]:OUTputtype

This command sets or returns the output channel [n] type.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:OUTputtype {DCDirect|DCAmplified |AC}
SOURce[n]:OUTputtype?
```

Arguments

DCDirect|DCAmplified |AC

- DCDirect: Direct DAC output
- DCAmplified: DC coupled amplified output
- AC: AC coupled output

The value of [n] indicates the channel number.

Returns

DCDirect|DCAmplified|AC

Examples

SOURce1:OUTputtype AC

Set the channel 1 output type to AC.

SOURce[n]:OUTputtype?

Example return: AC

SOURce[n]:SKEW

This command sets or returns skew value in seconds.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

None

Syntax

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

NOTE. *This command can be used only after the* `AWGControl:UPDATEdata` *command is called successfully.*

Arguments

<NR3>

The value of [n] indicates the channel number.

Returns

<NR3>

Examples

```
SOURce1:SKEW
SOURce1:SKEW 1E-9
```

Sets channel 1 skew to 1 ns.

```
SOURce1:SKEW?
```

Example return: 1E-9

SOURce[n]:VOCM[:IMMediate]

This command sets or returns the VOCM value in volts.

NOTE. *This command can be used only after the* `AWGControl:UPDATEdata` *command is called successfully. The channels must be initialized correctly before sending this command. If they are not initialized first, you will get an error, "No channels available."*

Group

Source

Related Commands

None

Syntax

```
SOURce[n]:VOCM <NR2>
SOURce[n]:VOCM?
```

Arguments

<NR2>

The value of [n] indicates the channel number.

The default value is 0 V.

NOTE. *VOCM value has no effect on AC coupled output.*

Returns

<NR2>

Examples

SOURce1:VOCM 0.2

Sets the VOCM to 0.4V.

SOURce1:VOCM?

Example return: 0.2

[SOURce[n]:]WAVeform

This command and query sets or returns the output waveform from the current waveform list for each channel when run mode is not Sequence.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Source

Related Commands

None

Syntax

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

Arguments

<wfm_name>::=<string>

The value of [n] indicates the channel number. If omitted, interpreted as 1.

Returns

<wfm_name>

Examples

SOURCE1:WAVeform "SINE100"

Loads a predefined waveform named "Sine100" into channel 1 memory.

SOURce1:WAVeform?

Example return: "Sine100"

SYSTem:ERRor[:NEXT]?

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

Group

System

Related Commands

None

Syntax

SYSTem:ERRor[:NEXT]?

Arguments

None

Returns

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

0 – No Error

Error / event number <NR1>
error / event description <string>

Examples

SYSTEM:ERROR:NEXT?

Example return: 0, No error

This response indicates that there are currently no errors.

SYSTem:VERSion?

This command returns the SCPI version number to which the command conforms. This command is query only.

Group

System

Related Commands

None

Syntax

```
SYSTem:VERSion?
```

Arguments

None

Returns

```
<NR2>::=YYYY.V
```

Examples

```
SYSTEM:VERSION?
```

Example return: 1999.0

TIMING[n]:CLOCKSOUrce

This command sets or returns the sampling clock source for channel [n] in Hertz.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Source

Related Commands

TIMING[n]:REFCLOCKSOUrce (on page 3-56)
TIMING[n]:SAMPLEFReq (on page 3-57)

Syntax

```
TIMING[n]:CLOCKSOUrce {INT|<NR3>}
```

Arguments

- INT: Internal sampling clock
- <NR3>: External sampling clock. If external clock source, the range is 1.25 GHz to 2.5 GHz.

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

CAUTION. To prevent damage to other instruments when using an external sampling clock in remote, change to the internal sampling clock before disconnecting the external sampling clock.

Returns

INT|<NR3>

Examples

TIMING:CLOCKSOUrce 2.1E9

Set the sampling clock source to external at 2.1 GHz.

TIMING:CLOCKSOUrce?

Example return: 2.1E9

TIMING[n]:REFCLOCKSOUrce

This command sets or returns the reference clock source for channel [n] in Hertz.

NOTE. *This command takes effect only after the AWGControl:UPDATEdata command is called.*

Group

Source

Related Commands

None

Syntax

```
TIMING[n]:REFCLOCKSOUrce {INT|<NR3>}
TIMING[n]:REFCLOCKSOUrce?
```

Arguments

- INT: Internal reference clock source
- <NR3>: External reference clock

If external reference clock, the range is 10 MHz to 80 MHz. The value of [n] indicates the channel number.

Default value = INT

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

CAUTION. To prevent damage to other instruments when using an external sampling clock in remote, change to the internal sampling clock before disconnecting the external sampling clock.

Returns

`INT|<NR3>`

Examples

`TIMING:REFCLOCKSOUrce 10E6`

Set the channel 1 reference clock to 10 MHz (external).

`TIMING:REFCLOCKSOUrce?`

Example output: 10E6

TIMING[n]:SAMPLEFReq

This command sets or returns sampling frequency for channel [n] in Hertz.

NOTE. *This command is valid only when the internal sampling clock is selected, and only takes effect after the `AWGControl:UPDATEdata` command is called.*

Group

Source

Related Commands

None

Syntax

```
TIMING[n]:SAMPLEFReq <NR3>
TIMING[n]:SAMPLEFReq?
```

Arguments

`<NR3>`

The value of [n] indicates the channel number.

NOTE. *In Single Sequencer mode, the [n] should be omitted because the instruction is common to both channels. If it is not omitted, the same value is set on both channels.*

Returns

<NR3>

Examples

```
TIMING:SAMPLEFReq 2.5E9
```

Sets the sampling frequency to 2.5 GHz

```
TIMING:SAMPLEFReq?
```

Example return: 2.5E9

*TRG

This command generates a trigger event. There is no query form of this command.

This is equivalent to pressing the Force Trigger button on the front panel.

NOTE. *In triggered run mode, the event (button release) is cleared automatically. In all other run modes, it is necessary to send an ABORt command to clear the event.*

Group

Trigger

Related Commands

TRIGger[:SEQuence][:IMMediate]

Syntax

```
*TRG
```

Arguments

None

Examples

```
*TRG
```

Generates a force trigger event.

TRIGger[:SEQuence][:IMMediate]

This command generates a force trigger event. There is no query form of this command.

This is equivalent to using the *TRG command.

NOTE. *In triggered run mode, the event (button release) is cleared automatically. In all other run modes, it is necessary to send an ABORt command to clear the event.*

Group

Trigger

Related Commands

*TRG (on page 3-58)

Syntax

```
TRIGger[:SEQuence][:IMMediate]
```

Arguments

None

Returns

None

Examples

```
TRIGger:SEQuence:IMMediate
```

Generates the force trigger event.

WLISt:LAST?

This command returns the name of the most recently added waveform in the waveform list. This command is query only.

Group

Waveform

Related Commands

None

Syntax

```
WLISt:LAST?
```

Arguments

None

Returns

```
[<waveform_name>]
<waveform_name>::=<string>
```

Examples

```
WLISt:LAST?
```

Example return: "sine_wf"

WLISt:LIST?

This command returns the names of the waveforms in the waveform list. This command is query only.

Group

Waveform

Related Commands

None

Syntax

```
WLISt:LIST?
```

Arguments

None

Returns

```
[<waveform_name>][,<waveform_name>]…
<waveform_name>::=<string>
```

Examples

```
WLISt:LIST?
```

Example return: "sine_wf, square_wf"

WLISt:NAME?

This command returns the waveform name of an element in the waveform list. The query form of this command can be used to query the waveform name in the waveform list. This command is query only.

Group

Waveform

Related Commands

None

Syntax

```
WLISt:NAME? <Index> Related Commands
<Index>::=<NR1>
```

Arguments

None

Returns

`<string>::=<wfm_name>` is the waveform name specified by <index>.

Examples

```
WLIST:NAME? 21
```

Example return: "untitled21"

WLISt:SIZE?

This command returns the size of the waveform list (number of waveforms). This command is query only.

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

Related Commands

WLISt:WAVeform:NEW (on page 3-66)

Syntax

```
WLISt:SIZE?
```

Arguments

None

Returns

<NR1>

At *RST, this returns the number of predefined waveforms. Refer to *RST (on page 3-27).

Examples

```
WLIST:SIZE?
```

Example return when user-defined waveform list is empty: 25

WLISt:WAVeform:DATA

This command transfers waveform data from the external controller into the waveform list or from the waveform list to the external control program.

NOTE. *Before transferring data to the instrument, a waveform must be created using the* `WLISt:WAVeform:NEW` *command.*

Use this command to set both analog, digital and marker data.

Using StartIndex and Size, part of a waveform can be transferred at a time. Very large waveforms can be transferred in chunks.

Waveform data is always transferred in the LSB first format.

The instrument supports floating-point format. Floating-point waveform data points occupy six bytes, so the total bytes will be six times the size of the waveform. The first four bytes of each data point represent the floating-point representation of the sample value and the fifth/sixth bytes store the digital and marker data. The marker data occupy the least significant bit of the fifth byte.

The minimum size of the waveform must be 64 points and the maximum size depends on the instrument model and installed options.

This command has a limit of 6,000,000 bytes of data.

The query form of this command inserts the LF character ("0A" bytes) at the end of the transmission.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Waveform

Related Commands

WLISt:WAVeform:NEW (on page 3-66)

Syntax

```
WLISt:WAVeform:DATA
<wfm_name>,<StartIndex>,<Size>,<block_data>
WLISt:WAVeform:DATA? <wfm_name>
```

Arguments

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

Returns

```
<block_data>
```

Examples

```
WLISt:WAVeform:DATA "TestWfm",0,1024,#46144xxxx…
```

This example transfers waveform data to a waveform called "TestWfm" created earlier using the `WLISt:WAVeform:NEW` command. The data size is 1024 points (6144 bytes) and the start index is 1 (the first data point).

The total bytes will be six times the size of the waveform.

WLISt:WAVeform:DELete

This command deletes the waveform from the currently loaded setup. There is no query form of this command.

CAUTION. The waveform will be deleted even if it is a part of the sequence. 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.

Group

Waveform

Related Commands

WLISt:SIZE? (on page 3-61)

Syntax

```
WLISt:WAVeform:DELete {<wfm_name>|ALL}
```

Arguments

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

Returns

None

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 named "Test1".

WLISt:WAVeform:DIGITAL:DATA

This command sets or queries the waveform digital and marker data.

NOTE. *This command returns or sends only the digital and marker data for the waveform.*
Each set of digital+marker data occupies two bytes. The least significant bit of the first byte is used for marker1/marker2.
When used on a waveform with [n] data points, you get only 2 X n bytes.
Block data can be sent in batches using the "Size" and "StartIndex" parameters.

This command has a limit of 6,000,000 bytes of data.

The query form of this command inserts the LF character ("0A" bytes) at the end of the transmission.

NOTE. *This command takes effect only after the `AWGControl:UPDATEdata` command is called.*

Group

Waveform

Related Commands

WLISt:WAVeform:NEW (on page 3-66)

Syntax

```
WLISt:WAVeform:DIGITAL:DATA
   <wfm_name>,<StartIndex>,<Size>,<block_data>
WLISt:WAVeform:DIGITAL:DATA? <wfm_name>
```

Arguments

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

Returns

<block_data>

Examples

WLISt:WAVeform:DIGITAL:DATA "TestDigital",0,64,#3128xxxx…

This example transfers waveform data to a waveform called "TestDigital" created earlier using the WLISt:WAVeform:NEW command. The data size is 64 points (128 bytes) and the start index is 1 (the first data point).

The total bytes will be two times the size of the waveform.

WLISt:WAVeform:GRANularity?

This command returns the granularity of sample points required for a valid waveform. This command is query only.

Group

Waveform

Related Commands

None

Syntax

WLISt:WAVeform:GRANularity?

Arguments

None

Returns

<NR1>

Examples

WLISt:WAVeform:GRANularity?

Example return: 64

WLISt:WAVeform:LENGth?

This command returns the size of the waveform. The returned value represents data points (not bytes). This command is query only.

Group

Waveform

Related Commands

WLISt:WAVeform:NEW (on page 3-66)

Syntax

WLISt:WAVeform:LENGth? <wfm_name>

Arguments

<wfm_name>::=<string>

Returns

<NR1>

Examples

WLIST:WAVeform:LENGth? "Sine 360"

Example return: 360

WLISt:WAVeform:LMAXimum?

This command returns the maximum number of waveform sample points allowed. This command is query only.

Group

Waveform

Related Commands

None

Syntax

WLISt:WAVeform:LMAXimum?

Arguments

None

Returns

<NR1>

Examples

WLISt:WAVeform:LMAXimum?

Return: 67108864 (if 64M option is installed)

WLISt:WAVeform:LMINimum?

This command returns the minimum number of waveform sample points required for a valid waveform. This command is query only.

Group

Waveform

Related Commands

None

Syntax

```
WLISt:WAVeform:LMINimum?
```

Arguments

None

Returns

<NR1>

Examples

```
WLISt:WAVeform:LMINimum?
```

Example return: 64

WLISt:WAVeform:NEW

This command creates a new empty waveform in the waveform list of the current setup. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Waveform

Related Commands

WLISt:WAVeform:DATA (on page 3-61)

Syntax

```
WLISt:WAVeform:NEW <wfm_name>,<Size>
```

Arguments

```
<wfm_name>::=<string>
<Size>::=<NR1>
```

Returns

None

Examples

```
WLISt:WAVeform:NEW "Test1", 1024
```

Creates a new waveform called Test1 with 1024 points.

WLISt:WAVeform:NORMalize

This command normalizes a waveform that exists in the waveform list of the current setup. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Waveform

Related Commands

None

Syntax

```
WLISt:WAVeform:NORMalize <wfm_name>,<Type>
```

Arguments

```
<wfm_name>::=<string>
<Type>::={FSCale|ZREFerence}
```

Returns

None

Examples

```
WLISf:WAVeform:NORMalize "Untitled25", FSC
```

Normalizes the waveform named "Untitled25".

WLISt:WAVeform:PREDefined?

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

NOTE. *Predefined waveforms have a fixed length and name. Therefore, renaming or deleting them is not possible.*
Creating a new waveform with the same name as the predefined waveform is not possible.
Data of a predefined waveform can be transferred to an external controller using the `WLISt:WAVeform:DATA` *command.*

Group

Waveform

Related Commands

None

Syntax

```
WLISt:WAVeform:PREDefined? <wfm_name>
```

Arguments

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

Returns

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

Examples

```
WLISt:WAVeform:PREDefined? "Sine_2048_P"
```

Example return: 1, which indicates that it is a predefined waveform.

WLISt:WAVeform:RESAmple

This command resamples a waveform that exists in the waveform list of the current setup. There is no query form of this command.

NOTE. *This command takes effect only after the* `AWGControl:UPDATEdata` *command is called.*

Group

Waveform

Related Commands

None

Syntax

```
WLISt:WAVeform:RESAmple <wfm_name>,<Size>
```

Arguments

```
<wfm_name::=<string>
<Size>::=<NR1>
```

Returns

None

Examples

```
WLISt:WAVeform:RESAmple "Untitled25", 1024
```

Resamples the waveform named "Untitled25", if it exists, to 1024 points.

WLISt:WAVeform:TYPE?

This command returns the type of the waveform. Currently, it always returns REAL. This command is query only.

Group

Waveform

Related Commands

WLISt:WAVeform:NEW (on page 3-66)

Syntax

```
WLISt:WAVeform:TYPE? <wfm_name>
```

Arguments

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

Returns

```
REAL
```

Examples

```
WLISt:WAVeform:TYPE? "Ramp1000"
```

Return: REAL

Appendix A: Reset Defaults

Reset Defaults

The following table lists the instrument default state after a *RST command is received.

Channel FUNCTionality	ARBitrary, NONE
Run Mode	SEQuence
Vocm	0
Output type	DCAmplified
Amplitude Scale	100
Offset	0
Marker Output Level	1
Marker Output Selection	LINE
Marker Output Skew	0
Marker Digital	False
Attenuation	0
Attenuation Manual	False
Reference Clock Source	INTernal
Sampling Clock Source	INTernal
Sampling Rate	2.5E9
Wait State	LAST
Trigger-In Threshold	1.65
Trigger-In Timer	1
Analog Skew	0
Dynamic Jump Table	It is an array of 256 elements initialized with 1
Jump Mode	EJUMP
Events Operator	OPERator1 = AND, OPERator2 = AND, OPERator3 = AND
Events Source	SOUrce1= FORCETRIGger, SOUrce2=TRUE, SOUrce3=TRUE, SOUrce4=TRUE
DIGital Skew	0
Channel Parameters Couple	NONE

OUTPut State	ON
Waveform List	Null
Sequencer List	Null

Appendix B: Error Codes

Command Error Codes and Messages

Command errors are returned when there is a syntax error in the command.

Error code	Error message
-350	Queue Overflow
-300	System Error
-301	No channels available
-302	Error to parsing waveform block data
-303	Waveform not found or not defined
-304	Sequencer not defined or incorrect data
-305	Waveform definition error
-306	Waveform out of range
-307	License option error
-308	Invalid run mode
-309	Subsequence error
-310	File Error
-311	Out of range error
-312	Application error
-313	Data type error
-314	Diagnostic error
-315	Calibration error
-316	Error on loading setting into the instrument
-317	Error on writing the serial number
-318	Error on reading the status of the digital input pins
-319	Sequencer Entry Error
-320	Attenuator Error
-321	Predefined Waveform Error (it is not possible to delete or create a predefined waveform)
-322	Waveform Granularity Error

Error code	Error message
-323	Digital port disabled error: the digital port is disabled. At the remote system startup the digital port is disabled; if the option is present, the customer can enable it with the commands AWGControl:CHANnel[n]:FUNCTionality ARB,LOWSPeed
0	No error
5	Too many numeric suffices in Command Spec
10	No Input Command to parse
14	Numeric suffix is invalid value
16	Invalid value in numeric or channel list, e.g. out of range
17	Invalid number of dimensions in a channel list
20	Parameter of type Numeric Value overflowed its storage
30	Wrong units for parameter
40	Wrong type of parameter(s)
50	Wrong number of parameters
60	Unmatched quotation mark (single/double) in parameters
65	Unmatched bracket
70	Command keywords were not recognized
200	No entry in list to retrieve (number list or channel list)
210	Too many dimensions in entry to be returned in parameters

Diagnostic Error Codes

Diagnostic error codes are returned when a diagnostic failure occurs.

Error code	Error message
1400000	TEST_USB_CONNECTION FAILED
1401000	TEST_NVMEMORY_ACCESS FAILED
1402000	TEST_LOAD_CONTROL_FPGA_FW FAILED
1403000	TEST_CONTROL_FPGA_REGISTRY_ACCESS FAILED
1404000	TEST_I2C_EPROM FAILED
1405000	TEST_LOAD_CHANNEL_FPGA_DIAGNOSTIC_FW FAILED_CHANNEL_A
1405010	TEST_CHANNEL_FPGA_REGISTRY_ACCESS FAILED_CHANNEL_A
1405020	TEST_CHANNEL_MEMORY FAILED_CHANNEL_A
1405030	TEST_CHANNEL_SPI_DAC_9739 FAILED_CHANNEL_A
1405040	TEST_CHANNEL_ADC_LTC2265_COMMUNICATION FAILED_CHANNEL_A
1406000	TEST_LOAD_CHANNEL_FPGA_DIAGNOSTIC_FW FAILED_CHANNEL_B
1406010	TEST_CHANNEL_FPGA_REGISTRY_ACCESS FAILED_CHANNEL_B
1406020	TEST_CHANNEL_MEMORY FAILED_CHANNEL_B
1406030	TEST_CHANNEL_SPI_DAC_9739 FAILED_CHANNEL_B
1406040	TEST_CHANNEL_ADC_LTC2265_COMMUNICATION FAILED_CHANNEL_B
1407000	TEST_DEVICE_INITIALIZATION FAILED_CHANNEL_A
1408000	TEST_CHANNEL_ADC_LTC2265 FAILED_CHANNEL_A
1408010	TEST_CHANNEL_PLL FAILED_CHANNEL_A
1408020	TEST_CHANNEL_ADF4351 FAILED_CHANNEL_A
1408031	TEST_CHANNEL_OFFSET_DC_AMP FAILED_CHANNEL_A_BRANCH_N
1408041	TEST_CHANNEL_VOCM_DC_AMP FAILED_CHANNEL_A_BRANCH_N
1408051	TEST_CHANNEL_VAO_DC_AMP FAILED_CHANNEL_A_BRANCH_N
1408061	TEST_CHANNEL_OFFSET_GAIN_DC_AMP_CHANNEL_A_BRANCH_N
1408071	TEST_CHANNEL_DIFFERENTIAL_OFFSET_DC_DIRECT FAILED_CHANNEL_A_BRANCH_N
1408081	TEST_CHANNEL_OFFSET_DC_DIRECT FAILED_CHANNEL_A_BRANCH_N
1408091	TEST_CHANNEL_VOCM_DC_DIRECT FAILED_CHANNEL_A_BRANCH_N
1408101	TEST_CHANNEL_VAO_DC_DIRECT FAILED_CHANNEL_A_BRANCH_N
1408111	TEST_CHANNEL_ATTENUATOR FAILED_CHANNEL_A_BRANCH_N

Error code	Error message
1408032	TEST_CHANNEL_OFFSET_DC_AMP FAILED_CHANNEL_A_BRANCH_P
1408042	TEST_CHANNEL_VOCM_DC_AMP FAILED_CHANNEL_A_BRANCH_P
1408052	TEST_CHANNEL_VAO_DC_AMP FAILED_CHANNEL_A_BRANCH_P
1408062	TEST_CHANNEL_OFFSET_GAIN_DC_AMP_CHANNEL_A_BRANCH_P
1408072	TEST_CHANNEL_DIFFERENTIAL_OFFSET_DC_DIRECT FAILED_CHANNEL_A_BRANCH_P
1408082	TEST_CHANNEL_OFFSET_DC_DIRECT FAILED_CHANNEL_A_BRANCH_P
1408092	TEST_CHANNEL_VOCM_DC_DIRECT FAILED_CHANNEL_A_BRANCH_P
1408102	TEST_CHANNEL_VAO_DC_DIRECT FAILED_CHANNEL_A_BRANCH_P
1408112	TEST_CHANNEL_ATTENUATOR FAILED_CHANNEL_A_BRANCH_P
1409000	TEST_CHANNEL_ADC_LTC2265 FAILED_CHANNEL_B
1409010	TEST_CHANNEL_PLL FAILED_CHANNEL_B
1409020	TEST_CHANNEL_ADF4351 FAILED_CHANNEL_B
1409031	TEST_CHANNEL_OFFSET_DC_AMP FAILED_CHANNEL_B_BRANCH_N
1409041	TEST_CHANNEL_VOCM_DC_AMP FAILED_CHANNEL_B_BRANCH_N
1409051	TEST_CHANNEL_VAO_DC_AMP FAILED_CHANNEL_B_BRANCH_N
1409061	TEST_CHANNEL_OFFSET_GAIN_DC_AMP_FAILED_CHANNEL_B_BRANCH_N
1409071	TEST_CHANNEL_DIFFERENTIAL_OFFSET_DC_DIRECT FAILED_CHANNEL_B_BRANCH_N
1409081	TEST_CHANNEL_OFFSET_DC_DIRECT FAILED_CHANNEL_B_BRANCH_N
1409091	TEST_CHANNEL_VOCM_DC_DIRECT FAILED_CHANNEL_B_BRANCH_N
1409101	TEST_CHANNEL_VAO_DC_DIRECT FAILED_CHANNEL_B_BRANCH_N
1409111	TEST_CHANNEL_ATTENUATOR FAILED_CHANNEL_B_BRANCH_N
1409032	TEST_CHANNEL_OFFSET_DC_AMP FAILED_CHANNEL_B_BRANCH_P
1409042	TEST_CHANNEL_VOCM_DC_AMP FAILED_CHANNEL_B_BRANCH_P
1409052	TEST_CHANNEL_VAO_DC_AMP FAILED_CHANNEL_B_BRANCH_P
1409062	TEST_CHANNEL_OFFSET_GAIN_DC_AMP_FAILED_CHANNEL_B_BRANCH_P
1409072	TEST_CHANNEL_DIFFERENTIAL_OFFSET_DC_DIRECT FAILED_CHANNEL_B_BRANCH_P
1409082	TEST_CHANNEL_OFFSET_DC_DIRECT FAILED_CHANNEL_B_BRANCH_P

Error code	Error message
1409092	TEST_CHANNEL_VOCM_DC_DIRECT FAILED_CHANNEL_B_BRANCH_P
1409102	TEST_CHANNEL_VAO_DC_DIRECT FAILED_CHANNEL_B_BRANCH_P
1409112	TEST_CHANNEL_ATTENUATOR FAILED_CHANNEL_B_BRANCH_P

Appendix C: Predefined Waveforms

Waveforms Included with the AWG4162

This section describes the waveforms that already defined in the AWG4162 when shipped.

Waveform name	Description
Triangle_10000_P	Triangle waveform, 10000 samples, 0.6 Vpp
Triangle_2048_P	Triangle waveform, 2048 samples, 0.6 Vpp
Triangle_64_P	Triangle waveform, 64 samples, 0.6 Vpp
DC_2048_P	DC Level, 2048 samples, 0.3 V
Ramp_10000_P	Ramp waveform, 10000 samples, 0.3 V
Ramp_2048_P	Ramp waveform, 2048 samples, 0.3 V
Ramp_64_P	Ramp waveform, 64 samples, 0.3 V
Sine_10000_P	Sine waveform, 10000 samples, 0.6 Vpp
Sine_2048_P	Sine waveform, 2048 samples, 0.6 Vpp
Sine_64_P	Sine waveform, 64 samples, 0.6 Vpp
Square_10000_P	Square waveform, 10000 samples, 0.6 Vpp
Square_2048_P	Square waveform, 2048 samples, 0.6 Vpp
Square_64_P	Square waveform, 64 samples, 0.6 Vpp
ClockFMax_2048_P	Clock waveform, 2048 samples, 0.6 Vpp

Index

*****

*CAL? • 3-13
*CLS • 3-14
*IDN? • 3-17
*OPC • 3-26
*OPT? • 3-26
*RST • 3-27
*TRG • 3-58

[

[SOURce[n]:]WAVeform • 3-53

A

Abbreviating Commands, Queries, and Parameters • 1-6
ABORt • 3-1
Advanced Command Descriptions • 3-1
Analog
  ANALOGCHANnel[n]:EVENTS[i]:OPERator[j] • 3-1
  ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j] • 3-2
ANALOGCHANnel[n]:EVENTS[i]:OPERator[j] • 3-1
ANALOGCHANnel[n]:EVENTS[i]:SOUrce[j] • 3-2
Appendix A
  Reset Defaults • A-1
Appendix B
  Error Codes • B-1
Appendix C
  Predefined Waveforms • C-1
Arbitrary Block Arguments • 1-9
AWG control
  AWGControl:APPLication:RUN • 3-3
  AWGControl:CONFigure:CNUMber? • 3-6
  AWGControl:CONFigure:SEQuencer:MODE • 3-6
  AWGControl:EVENt:DJUMp:DEFine • 3-7
  AWGControl:EVENt:JMODe • 3-8
  AWGControl:EVENt:TABLe[:IMMediate] • 3-8
  AWGControl:RMODe • 3-9
  AWGControl:RUN[:IMMediate] • 3-10
  AWGControl:SNAMe? • 3-10
  AWGControl:SREStore • 3-11
  AWGControl:SSAVe • 3-11
  AWGControl:STOP[:IMMediate] • 3-12
  AWGControl:UPDATEdata • 3-12
  AWGControl[
    CHANnel[n]]
      RSTate? • See AWGControl[:CHANnel[n]]:FUNCTionality
  AWGControl[:CHANnel[n]]:FUNCTionality • 3-4
  AWGControl[:CHANnel[n]]:WAITstate • 3-5
AWGControl:APPLication:RUN • 3-3
AWGControl:APPLication:STATe? • 3-3
AWGControl:CONFigure:CNUMber? • 3-6
AWGControl:CONFigure:SEQuencer:MODE • 3-6
AWGControl:EVENt:DJUMp:DEFine • 3-7
AWGControl:EVENt:JMODe • 3-8
AWGControl:EVENt:TABLe[:IMMediate] • 3-8
AWGControl:RMODe • 3-9
AWGControl:RUN[:IMMediate] • 3-10
AWGControl:SNAMe? • 3-10
AWGControl:SREStore • 3-11
AWGControl:SSAVe • 3-11
AWGControl:STOP[:IMMediate] • 3-12
AWGControl:UPDATEdata • 3-12
AWGControl[:CHANnel[n]]:FUNCTionality • 3-4
AWGControl[:CHANnel[n]]:RSTate? • 3-5
AWGControl[:CHANnel[n]]:WAITstate • 3-5

B

Backus-Naur Form Definition • 1-1
Block Arguments • 1-8
Block Data Format • 2-1

C

calibration
  *CAL? • 3-13
  CALibration[:ALL] • 3-14
Calibration and Diagnostic Commands • 1-12
CALibration[:ALL] • 3-14
channel
  INSTrument:COUPle:SOURce • 3-18
Channel Number Optional Commands • 1-23
Clear
  *CLS • 3-14
Clock
  TIMING[n]:CLOCKSOUrce • 3-55
  TIMING[n]:REFCLOCKSOUrce • 3-56
Command and Query Structure • 1-2
Command Entry • 1-2
Command Error Codes and Messages • B-1
Command Groups • 1-12
Command Messages • 1-3
Command Syntax • 1-1
Commands • 1-4
Concatenating Commands and Queries • 1-6
Control Commands • 1-13
Creating and Working with Sequences • 2-4
Creating Commands • 1-5
Creating Queries • 1-5

D

DAC
SOURce[n]:DAC:RESolution? • 3-45
Delete
SLISt:SUBSequence:DELete • 3-41
WLISt:WAVeform:DELete • 3-62
Diagnostic
DIAGnostic:DATA? • 3-15
DIAGnostic[:IMMediate] • 3-15
Diagnostic Error Codes • B-3
DIAGnostic:DATA? • 3-15
DIAGnostic[:IMMediate] • 3-15
Digital
SOURce[n]:DIGital:SKEW[:IMMediate] • 3-46
SOURce[n]:DIGital:STATe • 3-47
WLISt:WAVeform:DIGITAL:DATA • 3-63
Digital Data • 2-2
Documentation • vii

E

Error
SYSTem:ERRor[:NEXT]? • 3-54
Event Commands • 1-14
Events
*TRG • 3-58
EVENTS:TRIGINTHReshold • 3-17
EVENTS:TRIGINTIMERPeriod • 3-16
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-32
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-40
EVENTS:TRIGINTHReshold • 3-17
EVENTS:TRIGINTIMERPeriod • 3-16

G

General Features • viii
General Rules for Using SCPI Commands • 1-11
Granularity • 2-4

I

Identify
*IDN? • 3-17
Instrument Commands • 1-14
INSTrument:COUPle:SOURce • 3-18
Introduction • 2-1

J

Jump
List
WLISt:LAST? • 3-59
WLISt:NAME? • 3-60
WLISt:SIZE? • 3-61
WLISt:WAVeform:DATA • 3-61
WLISt:WAVeform:RESAmple • 3-68
List:WLISt:LIST? • 3-60
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-30
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-36
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent • 3-37
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-38

M

Marker
SOURce[n]:MARKer:SKEW[:IMMediate] • 3-47
SOURce[n]:MARKer:TYPe • 3-48
SOURce[n]:MARKer:VOLTage:SELection[:IMMediate] • 3-50
SOURce[n]:MARKer:VOLTage[:LEVel][:IMMediate][:AMPLitude] • 3-49
Mass memory
MMEMory:CATalog? • 3-19
MMEMory:CDIRectory • 3-20
MMEMory:DATA • 3-21
MMEMory:DELete • 3-21
MMEMory:EXPort • 3-22
MMEMory:IMPort • 3-23
MMEMory:MDIRectory • 3-24
MMEMory:MSIS • 3-25
Mass Memory Commands • 1-15
MMEMory:CATalog? • 3-19
MMEMory:CDIRectory • 3-20
MMEMory:DATA • 3-21
MMEMory:DELete • 3-21
MMEMory:EXPort • 3-22
MMEMory:IMPort • 3-23
MMEMory:MDIRectory • 3-24
MMEMory:MSIS • 3-25

N

New
SLISt:SUBSequence:NEW • 3-43
WLISt:WAVeform:NEW • 3-66

O

Offset
SOURce[n]:OFFSet[:IMMediate] • 3-50

Output
OUTPut[n][:STATe] • 3-27
SOURce[n]:OUTputtype • 3-51
Output Commands • 1-15
OUTPut[n][:STATe] • 3-27

P

Parameter Types • 1-8
Parameters • 1-5
Preface • vi

Q

Queries • 1-4
Query Responses • 1-5
Quoted Strings • 1-9

R

Real Format • 2-2
Reset
*RST • 3-27
Reset Defaults • A-1

S

Sample rate
TIMING[n]:SAMPLEFReq • 3-57
WLISt:WAVeform:LMAXimum? [<SampleRate>] • 3-65
WLISt:WAVeform:LMINimum? [<SampleRate>] • 3-65
SCPI Commands and Queries • 1-4
SCPI compliance
SYSTem:VERSion? • 3-55
Sequence
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform • 3-33
SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO • 3-28
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-30
SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions • 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence • 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-32
SEQuence[:CHANnel[n]]:LENGth • 3-33
Sequence Commands • 1-16
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform • 3-33
SEQuence[:CHANnel[n]]:ELEMent[i]:GOTO • 3-28
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPEvent • 3-29
SEQuence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-30
SEQuence[:CHANnel[n]]:ELEMent[i]:REPetitions • 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:SUBSequence • 3-31
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-32
SEQuence[:CHANnel[n]]:LENGth • 3-33
Skew
SOURce[n]:DIGital:SKEW[:IMMediate] • 3-46
SOURce[n]:MARKer:SKEW[:IMMediate] • 3-47
SOURce[n]:SKEW • 3-52
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform • 3-41
SLISt:SUBSequence:DELete • 3-41
SLISt:SUBSequence:LENGth • 3-42
SLISt:SUBSequence:NEW • 3-43
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO • 3-35
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-36
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent • 3-37
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-38
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions • 3-39
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-40
SLISt[:CHANnel[n]]:NAME? • 3-34
SLISt[:CHANnel[n]]:SIZE? • 3-35
Source
[SOURce[n]:]WAVeform • 3-53
SOURce[n]:AMPlitudescale[:IMMediate] • 3-43
SOURce[n]:ATTENuation • 3-44
SOURce[n]:ATTENuation:TYPe • 3-45
SOURce[n]:DAC:RESolution? • 3-45
SOURce[n]:DIGital:SKEW[:IMMediate] • 3-46
SOURce[n]:DIGital:STATe • 3-47
SOURce[n]:MARKer:SKEW[:IMMediate] • 3-47
SOURce[n]:MARKer:TYPe • 3-48
SOURce[n]:MARKer:VOLTage:SELection[:IMMediate] • 3-50
SOURce[n]:MARKer:VOLTage[:LEVel][:IMMediate][:AMPLitude] • 3-49
SOURce[n]:OFFSet[:IMMediate] • 3-50
SOURce[n]:SKEW • 3-52

SOURce[n]:VOCM[:IMMediate] • 3-52
Source Commands • 1-17
SOURce[n]:AMPlitudescale[:IMMediate] • 3-43
SOURce[n]:ATTENuation • 3-44
SOURce[n]:ATTENuation:TYPe • 3-45
SOURce[n]:DAC:RESolution? • 3-45
SOURce[n]:DIGital:SKEW[:IMMediate] • 3-46
SOURce[n]:DIGital:STATe • 3-47
SOURce[n]:MARKer:SKEW[:IMMediate] • 3-47
SOURce[n]:MARKer:TYPe • 3-48
SOURce[n]:MARKer:VOLTage:SELection[:IMMediate] • 3-50
SOURce[n]:MARKer:VOLTage[:LEVel][:IMMediate][:AMPLitude] • 3-49
SOURce[n]:OFFSet[:IMMediate] • 3-50
SOURce[n]:OUTputtype • 3-51
SOURce[n]:SKEW • 3-52
SOURce[n]:VOCM[:IMMediate] • 3-52
Special Characters • 1-6
Status
*CLS • 3-14
Status Commands • 1-18
Subsequence
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform • 3-41
SLISt:SUBSequence:DELete • 3-41
SLISt:SUBSequence:LENGth • 3-42
SLISt:SUBSequence:NEW • 3-43
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:GOTO • 3-35
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPAddr • 3-36
SLISt:SUBSequence[:CHANNEL[n]]:ELEMent[i]:JUMPEvent • 3-37
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:JUMPType • 3-38
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:REPetitions • 3-39
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-40
SLISt[:CHANnel[n]]:NAME? • 3-34
SLISt[:CHANnel[n]]:SIZE? • 3-35
Subsequence Commands • 1-19
Synchronization
*OPC • 3-26
Synchronization Commands • 1-20
Syntax and Commands • 1-1
System
*IDN? • 3-17
*OPT? • 3-26
*RST • 3-27
SYSTem:VERSion? • 3-55
System Commands • 1-20
SYSTem:ERRor[:NEXT]? • 3-54
SYSTem:VERSion? • 3-55

T

Termination • 1-7
Timing
TIMING[n]:CLOCKSOUrce • 3-55
TIMING[n]:REFCLOCKSOUrce • 3-56
TIMING[n]:SAMPLEFReq • 3-57
TIMING[n]:CLOCKSOUrce • 3-55
TIMING[n]:REFCLOCKSOUrce • 3-56
TIMING[n]:SAMPLEFReq • 3-57
Trigger
*TRG • 3-58
TRIGger[:SEQuence][:IMMediate] • 3-58
Trigger Commands • 1-20
TRIGger[:SEQuence][:IMMediate] • 3-58

U

Unit and SI Prefixes • 1-10

V

VOCM
SOURce[n]:VOCM[:IMMediate] • 3-52

W

Wait
SEQuence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-32
SLISt:SUBSequence[:CHANnel[n]]:ELEMent[i]:WAITEvent • 3-40
Waveform
[SOURce[n]:]WAVeform • 3-53
SEQuence:CHANnel[n]:ELEMent[i]:WAVeform • 3-33
SLISt:SUBSequence:CHANnel[n]:ELEMent[i]:WAVeform • 3-41
WLISt:LAST? • 3-59
WLISt:LIST? • 3-60
WLISt:NAME? • 3-60
WLISt:SIZE? • 3-61
WLISt:WAVeform:DATA • 3-61
WLISt:WAVeform:DELete • 3-62
WLISt:WAVeform:DIGITAL:DATA • 3-63
WLISt:WAVeform:GRANularity? • 3-64
WLISt:WAVeform:LENGth? • 3-64
WLISt:WAVeform:LMAXimum? [<SampleRate>] • 3-65
WLISt:WAVeform:LMINimum? [<SampleRate>] • 3-65
WLISt:WAVeform:NEW • 3-66
WLISt:WAVeform:NORMalize • 3-67
WLISt:WAVeform:PREDefined? • 3-67

WLISt:WAVeform:RESAmple • 3-68
WLISt:WAVeform:TYPE? • 3-69
Waveform Commands • 1-21
Waveform Data Format • 2-1
Waveform Data Transfer • 2-1
Waveforms Included with the AWG4162 • C-1
WLISt:LAST? • 3-59
WLISt:LIST? • 3-60
WLISt:NAME? • 3-60
WLISt:SIZE? • 3-61
WLISt:WAVeform:DATA • 3-61
WLISt:WAVeform:DELete • 3-62
WLISt:WAVeform:DIGITAL:DATA • 3-63
WLISt:WAVeform:GRANularity? • 3-64
WLISt:WAVeform:LENGth? • 3-64
WLISt:WAVeform:LMAXimum? • 3-65
WLISt:WAVeform:LMINimum? • 3-65
WLISt:WAVeform:NEW • 3-66
WLISt:WAVeform:NORMalize • 3-67
WLISt:WAVeform:PREDefined? • 3-67
WLISt:WAVeform:RESAmple • 3-68
WLISt:WAVeform:TYPE? • 3-69

Contact Information:

Australia* 1 800 709 465
Austria 00800 2255 4835
Balkans, Israel, South Africa and other ISE Countries +41 52 675 3777
Belgium* 00800 2255 4835
Brazil +55 (11) 3759 7627
Canada 1 800 833 9200
Central East Europe / Baltics +41 52 675 3777
Central Europe / Greece +41 52 675 3777
Denmark +45 80 88 1401
Finland +41 52 675 3777
France* 00800 2255 4835
Germany* 00800 2255 4835
Hong Kong 400 820 5835
India 000 800 650 1835
Indonesia 007 803 601 5249
Italy 00800 2255 4835
Japan 81 (3) 6714 3010
Luxembourg +41 52 675 3777
Malaysia 1 800 22 55835
Mexico, Central/South America and Caribbean 52 (55) 56 04 50 90
Middle East, Asia, and North Africa +41 52 675 3777
The Netherlands* 00800 2255 4835
New Zealand 0800 800 238
Norway 800 16098
People's Republic of China 400 820 5835
Philippines 1 800 1601 0077
Poland +41 52 675 3777
Portugal 80 08 12370
Republic of Korea +82 2 6917 5000
Russia / CIS +7 (495) 6647564
Singapore 800 6011 473
South Africa +41 52 675 3777
Spain* 00800 2255 4835
Sweden* 00800 2255 4835
Switzerland* 00800 2255 4835
Taiwan 886 (2) 2656 6688
Thailand 1 800 011 931
United Kingdom / Ireland* 00800 2255 4835
USA 1 800 833 9200
Vietnam 12060128

*** European toll-free number. If not accessible, call:** +41 52 675 3777

Find more valuable resources at TEK.COM

077-1309-00