

AFG1022
Arbitrary Function Generator
Programmer Manual



077-1057-00

Tektronix

AFG1022
Arbitrary Function Generator
Programmer Manual

Copyright © Tektronix. All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

Contacting Tektronix

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

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Warranty

Tektronix warrants that 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. 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.

[W2 – 15AUG04]

Table of contents

Getting Started	1
Introduction.....	1
Connecting the Interface.....	1
Using TekVISA	1
Where to find more information	2
Syntax and Commands.....	3
Command Syntax.....	3
Backus-Naur Form Definition	3
Command and Query Structure	3
SCPI Commands and Queries.....	5
IEEE 488.2 Common Commands.....	9
Command Groups	10
Command Descriptions.....	13
*CLS (No Query Form).....	13
*IDN? (Query Only).....	13
MMEMory:CATalog? (Query Only).....	14
MMEMory:CDIRectory	15
MMEMory:DELeTe (No Query Form).....	16
*OPT? (Query Only)	16
OUTPut[1 2][:STATe]	17
*RST (No Query Form)	17
[SOURce1]:AM:STATe.....	18
[SOURce1]:BURSt:MODE	18
[SOURce1]:BURSt:NCYCles	19
[SOURce1]:BURSt:STATe.....	19
[SOURce1]:FM:STATe.....	20
[SOURce[1 2]]:FREQuency:CONCurent[:STATe]	20
[SOURce[1 2]]:FREQuency[:CW :FIXed].....	21
[SOURce1]:FREQuency:MODE.....	22
[SOURce1]:FSKey:STATe	23
[SOURce[1 2]]:FUNctIon:EFILe.....	23
[SOURce[1 2]]:FUNctIon[:SHAPE]	24
[SOURce[1 2]]:PHASe[:ADJust]	25
[SOURce1]:PM:STATe	26
[SOURce[1 2]]:VOLTagE[:LEVel][:IMMediate]:OFFSet.....	26
[SOURce[1 2]]:VOLTagE[:LEVel][:IMMediate][:AMPLitude].....	27
SYSTem:ERRor[:NEXT]? (Query Only)	28
TRACe DATA:CATalog? (Query Only).....	29

Table of Contents

TRACe DATA:COPIY (No Query Form)	29
TRACe DATA[:DATA]	30
TRACe DATA[:DATA]:VALue.....	30
TRACe DATA:POINts	31
*WAI (No Query Form).....	32
Command Errors.....	33
Index	34

Getting Started

Introduction

This programmer guide provides information to use commands for remotely controlling your instrument. With this information, write computer programs that will perform functions such as setting the front-panel controls, selecting clock source, setting sampling rate, and exporting data for use in other programs.

Connecting the Interface

The AFG1022 has a USB (type B) connector on the rear panel, as shown in the following figure. This connector conforms to USB-TMC. Attach a USB cable to this connector.

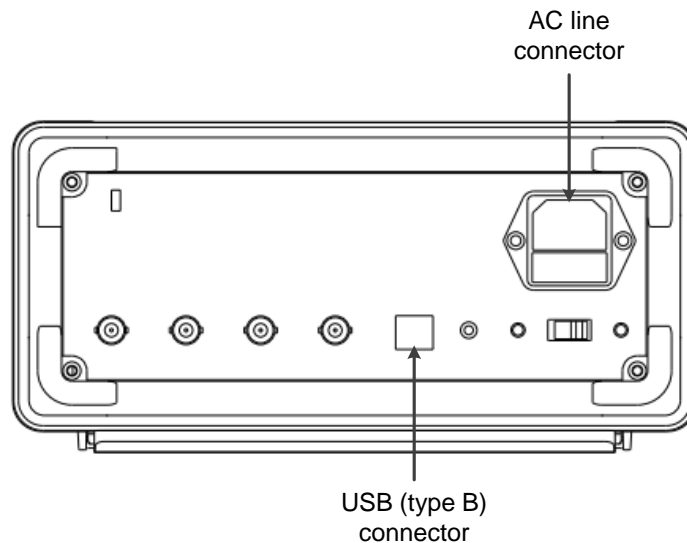


Figure 1: USB (type B) connector

Using TekVISA

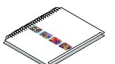


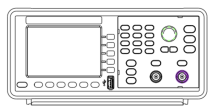






TekVISA is Tektronix implementation of VISA (Virtual Instrument Software Architecture), an industry-standard communication protocol. VISA provides a common standard for software developers so that software from multiple vendors, such as instrument drivers, can run on the same platform. TekVISA is industry-compliant software, available with selected Tektronix instruments. You can use this software to write (or draw) interoperable instrument drivers in a variety of Application Development Environments (ADEs). It implements a subset of Version 2.2 of the VISA specification for controlling USB instrument interface locally.

Installation Use an internet browser to access the Tektronix Web site (www.tektronix.com) and download the current TekVISA to your PC. Unzip the downloaded file in a temporary directory of your choice and run *Setup.exe*.

NOTE: The details on TekVISA concepts and operations are explained in the TekVISA Programmer Manual that can be also found on the Tektronix Web site.

Where to find more information

The following table lists related documentation available for your instrument. The documentation is available on the Product Documentation CD and on the Tektronix Web site (www.tektronix.com/manuals).

Item	Purpose	Location
Important safety and compliance instructions	Compliance and safety instructions	 +  +  WWW.Tektronix.com
Built-in Help	UI Help and Operation	
Quick Start User Manual	Unpacking, Installation, Tutorials, Operation, and Overviews	 +  WWW.Tektronix.com
Programmer Manual	Menu Structures, User Interface, and Programming Information	 +  WWW.Tektronix.com
Technical Reference	Specifications and performance verification procedures	 +  WWW.Tektronix.com

Syntax and Commands

Command Syntax

You can control the operations and functions of the instrument through the USB 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* (See page 10.) for a listing of the commands by command group, or use the index to locate a specific command.

Backus-Naur Form Definition

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

Table 1: BNF symbols and meanings

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

Command and Query Structure

- Overview** Commands consist of set commands and query commands (usually simply 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 `MMEMory: CDI Rectory` has a query form `MMEMory: CDI Rectory?`. Not all commands have both a set and a query form; some commands are set only and some are query only.
- 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.

Table 2: 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. Arguments are separated from the header by a <Space>. Arguments are separated from each other by a <Comma>.
<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.

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 instrument to return information about its status or settings. Queries have the structure:

[:]<Header>?

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

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.

Query Responses When a query is sent to the instrument, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format.

- Command Entry** Follow these general rules when entering commands:
- Enter commands in upper or lower case.
 - 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 consists of just a combination of white space characters and line feeds.

SCPI Commands and Queries

The instrument 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. The top level of the tree is the root node; it is followed by one or more lower-level nodes.

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

- Creating Commands** SCPI commands are created by stringing together the nodes of a subsystem hierarchy and separating each node by a colon.
- To create a SCPI command, start with the root node 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.
- 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.
- Query Responses** The query causes the instrument to return information about its status or settings. When a query is sent to the instrument, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format.
- Parameter Types** Every parameter in the command and query descriptions is of a specified type. (See Table 3.) The parameters are enclosed in brackets, such as <value>.

The parameter type is listed after the parameter and is enclosed in parentheses, for example, (boolean). Some parameter types are defined specifically for the instrument command set and some are defined by SCPI.

Table 3: Parameter types used in syntax descriptions

Parameter type	Description	Example
arbitrary block ¹	A specified length of arbitrary data	#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><&EOL>
boolean	Boolean numbers or values	ON or ≠ 0 OFF or 0
discrete	A list of specific values	MIN, MAX
binary	Binary numbers	#B0110
octal	Octal numbers	#Q57, #Q3
hexadecimal ²	Hexadecimal numbers (0-9, A, B, C, D, E, F)	#H AA, #H1
NR1 ² numeric	Integers	0, 1, 15, -1
NR2 ^{2,3} numeric	Decimal numbers	1.2, 3.141516, -6.5
NR3 ² numeric	Floating point numbers	3.1415E-9, -16.1E5
Nrf ² numeric	Flexible decimal number that may be type NR1, NR2 or NR3	See NR1, NR2, and NR3 examples
string ⁴	Alphanumeric characters (must be within quotation marks)	"Testing 1, 2, 3"

¹ Defined in ANSI/IEEE 488.2 as "Definite Length Arbitrary Block Response Data."

² An ANSI/IEEE 488.2-1992-defined parameter type.

³ Some commands and queries will accept an octal or hexadecimal value even though the parameter type is defined as NR1.

⁴ Defined in ANSI/IEEE 488.2 as "String Response Data."

Special Characters

The Line Feed (LF) character or the New Line (NL) character (ASCII 10), and all characters in the range of ASCII 127-255 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 upper and lower case letters. The upper case letters indicate the accepted short form of a command. As shown in the following figure, create a short form by using only the upper case letters. The accepted short form and the long form are equivalent and request the same action of the instrument.

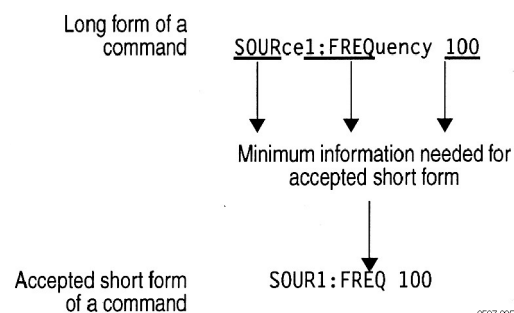


Figure 2: Example of abbreviating a command

NOTE: *The numeric suffix of a command or query may be included in either the long form or short form; the instrument will default to "1" if no suffix is used.*

Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, 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 single chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

If a command or query has the same root and lower-level nodes as the previous command or query, you can omit these nodes.

Unit and SI Prefix

If the decimal numeric argument refers to amplitude, frequency, 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 Systeme International d'Unites standard.) For example, use the input format 200 mV or 1.0 MHz instead of 200.0E-3 or 1.0E+6, respectively, to specify voltage or frequency.

The following table lists the available units.

Table 4: Available units

Symbol	Meaning
DEG	degree (phase)
Hz	hertz (frequency)
PCT	percent (%)
s	second (time)
V	volt
Vpp	volt

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

15.0E6, 1.5E7Hz, 15000000, 15000000Hz, 15MHz, etc.
("15M" is not allowed.)

General rules for using SCPI commands

Here 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 upper case, lower case, or a mixture of both cases for all commands, queries, and parameters.

SOURCE1: FREQUENCY 10MHZ

is the same as

source1: frequency 100mhz

and

SOURCE1: frequency 10MHZ

NOTE: Literal strings (quoted) are case sensitive, for example, file names.

- No embedded spaces are allowed between or within nodes.

correct **SOURCE1: FREQUENCY 10MHZ**

incorrect **SOURCE1: FREQUENCY 10MHZ**

IEEE 488.2 Common Commands

Description ANSI/IEEE Standard 488.2 defines the codes, formats, protocols, and usage of common commands and queries used on the interface between the controller and the instruments. The instrument complies with this standard.

Command and Query Structure The syntax for an IEEE 488.2 common command is an asterisk (*) followed by a command and, optionally, a space and parameter value. The syntax for an IEEE 488.2 common query is an asterisk (*) followed by a query and a question mark. All of the common commands and queries are listed in the last part of the *Syntax and Commands* section. The following are examples of common commands:

- *CLS

The following are examples of common queries

- *IDN?

Command Groups

This section lists the commands organized by functional group. The Command Descriptions section lists all commands alphabetically. (See page 13.)

Mass Memory Commands

Mass Memory commands let you change mass memory attributes. The following table lists and describes the Mass Memory commands.

Table 5: Mass Memory commands

Command	Description
MMEMory:CATalog?	Query the status of mass memory
MMEMory:CDIRectory	Set/query current directory
MMEMory:DELeTe	Delete file or directory in mass memory

Output Commands

Output commands let you set output attributes. The following table lists and describes the Output commands.

Table 6: Output commands

Command	Description
OUTPut[1 2][:STATe]	Set/query output on or off

Source Commands Source commands let you set waveform output parameters. The following table lists and describes the Source commands.

Table 7: Source commands

Command	Description
[SOURce1]:AM:STATe	Set/query amplitude modulation status
[SOURce1]:BURSt:MODE	Set/query burst mode
[SOURce1]:BURSt:NCYCles	Set/query burst mode waveform output cycle
[SOURce1]:BURSt:STATe	Set/query burst mode status
[SOURce1]:FM:STATe	Set/query frequency modulation status
[SOURce[1 2]]:FREQuency:CONCurent[:STATe]	Set/query concurrent change of frequency
[SOURce[1 2]]:FREQuency[:CW]:FIXed]	Set/query output waveform frequency
[SOURce1]:FREQuency:MODE	Set/query sweep status
[SOURce1]:FSKey:STATe	Set/query FSK status
[SOURce[1 2]]:FUNCTion:EFILe	Set/query EFILE name
[SOURce[1 2]]:FUNCTion[:SHAPE]	Set/query output waveform
[SOURce[1 2]]:PHASe[:ADJust]	Set/query output waveform phase
[SOURce1]:PM:STATe	Set/query phase modulation status
[SOURce[1 2]]:VOLTage[:LEVel] [:IMMediate]: OFFSet	Set/query output offset voltage Set/query
[SOURce[1 2]]:VOLTage[:LEVel] [:IMMediate][: AMPLitude]	output amplitude

Status Commands Status commands let you determine the status of the instrument. The following table lists and describes the Status commands.

Table 8: Status commands

Command	Description
*CLS	Clear all event registers and queues

System Commands System commands let you control miscellaneous instrument functions. The following table lists and describes the System commands.

Table 9: System commands

Command	Description
*IDN?	Return identification information
*OPT?	Return option information
*RST	Reset
SYSTem:ERRor[:NEXT]?	Return error event queue

Synchronization Commands Synchronization commands let you synchronize the operation of the instrument. The following table lists and describes the Synchronization commands.

Table 10: Synchronization commands

Command	Description
*WAI	Wait to continue

Trace Commands Trace commands let you set the edit memory and user waveform memory. The following table lists and describes the Trace commands.

Table 11: Trace commands

Command	Description
TRACe DATA:CATalog?	Return user waveform memory status
TRACe DATA:COpy	Copy edit memory (or user waveform memory) content to user waveform memory (or edit memory)
TRACe DATA[:DATA]	Set/query waveform data to edit memory
TRACe DATA[:DATA]:VALue	Set/query waveform data in edit memory
TRACe DATA:POINts	Set/query number of points for waveform data in edit memory

Command Descriptions

Commands either set or query instrument values. Some commands both set and query, some only set, and some only query.

Manual Conventions

This manual uses the following conventions:

- No Query Form indicates set-only commands
- A question mark (?) appended to the commands and Query Only indicates query-only commands
- Fully spells out headers, mnemonics, and arguments with the minimal spelling shown in upper case; for example, to use the abbreviated form of the DISPLAY:BRIGhtness command, just type DISP:BRIG
- Syntax of some commands varies, depending on the model of instrument you are using; differences are noted

*CLS (No Query Form)

This command clears all the event registers and queues, which are used in the instrument status and event reporting system.

Group	Status
Syntax	*CLS
Arguments	None
Examples	*CLS clears all the event registers and queues.

*IDN? (Query Only)

This query-only command returns identification information on the instrument.

Group	System
Syntax	*IDN?
Arguments	None
Returns	<p><Manufacturer>, <Model>, <Serial Number>, <Firmware Level> where: <Manufacturer>::= TEKTRONIX <Model>::={AFG1022} <Serial Number> <Firmware Level></p>
Examples	<p>*IDN? might return the following response: TEKTRONIX, AFG1022, 1331030, V1. 24</p>

MMEMory:CATalog? (Query Only)

This query-only command returns the current state of the mass storage system (USB memory).

Group	Mass Memory
Syntax	MMEMory: CATal og?
Related Commands	MMEMory:CDIRectory
Arguments	None
Returns	<p><NR1>, <NR1>[, <file_name>, <file_type>, <file_size>]... where:</p>

The first <NR1> indicates that the total amount of storage currently used, in bytes.
The second <NR1> indicates that the free space of mass storage, in bytes.

<file_name> is the name of directory or file. If the name exceeds 22 characters in length, it will be shortened to 8 characters (without suffix) in 8.3 name format.

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

<file_size> is the size of the file, in bytes. This value will be 0 for directory.

Examples The USB memory includes the Case and PWS4000-Main-CPU-Update folders, a SAMPLE1.tfw file, and a Test.zip file. The directory name PWS4000-Main-CPU-Update will be shortened to PWS400~1.

MMEMory: CATal og? might return the following response:

```
32751616, 27970560, "Case, DIR, 0", "PWS400~1, DIR, 0", "SAMPLE1. tfw,
, 5412", "Test. zi p, , 1735"
```

MMEMory:CDIRectory

This command changes the current working directory in the mass storage system.

Group Mass Memory

Syntax **MMEMory: CDI Rectory** [**<di rectory_name>**]
MMEMory: CDI Rectory?

Arguments <directory_name>::=<string> indicates the current working directory for the mass storage system.

Returns <directory_name>::=<string>

Examples **MMEMory: CDI Rectory** **"/AFG/WORK0"**
changes the current directory to /AFG/WORK0.

MMEMory:DELete (No Query Form)

This command deletes a file or directory from the mass storage system. If a specified file in the mass storage is not allowed to overwrite or delete, this command causes an error. You can delete a directory if it is empty.

Group	Mass Memory
Syntax	MMEMory: DELete <file_name>
Arguments	<file_name>::=<string> specifies a file to be deleted and should include full path.
Examples	MMEMory: DELete "/AFG/WORK0/TEK001.tfw" deletes the specified file from the /AFG/WORK directory.

*OPT? (Query Only)

This query-only command returns a list of the options installed in your instrument.

Group	System
Syntax	*OPT?
Arguments	None
Returns	<OPT>[,<OPT>[,<OPT>[,<OPT>]]]
Examples	*OPT? might return 0, which indicates no option is installed in the instrument.

OUTPut[1|2][:STATe]

This command sets or query the instrument output state for the specified channel.

Group	Output
Syntax	<code>OUTPut [1 2] [: STATe] { ON OFF <NR1> }</code> <code>OUTPut [1 2] [: STATe] ?</code>
Arguments	ON or <NR1>≠0 enables the instrument output. OFF or <NR1>=0 disables the instrument output.
Returns	<NR1>
Examples	<code>OUTPut 1: STATe ON</code> sets the instrument CH 1 output to ON.

*RST (No Query Form)

This command resets the instrument to the factory default settings.

Group	System
Syntax	<code>*RST</code>
Arguments	None
Examples	<code>*RST</code> resets the instrument settings to the factory defaults.

[SOURce1]:AM:STATe

This command enables or disables AM modulation for the specified channel. The query command returns the state of AM modulation.

Group	Source
Syntax	[SOURce1]: AM: STATe {ON OFF <NR1>} [SOURce1]: AM: STATe?
Arguments	If [SOURce1] are omitted, CH 1 is specified automatically. ON or <NR1>≠0 enables AM modulation. OFF or <NR1>=0 disables AM modulation.
Returns	<NR1>
Examples	SOURce1: AM: STATe ON enables the CH 1 AM modulation.

[SOURce1]:BURSt:MODE

This command sets or queries the burst mode for the specified channel.

Group	Source
Syntax	[SOURce1]: BURSt: MODE {TRIGgered GATed} [SOURce1]: BURSt: MODE?
Arguments	TRIGgered means that triggered mode is selected for burst mode. GATed means that gated mode is selected for burst mode.
Returns	TRIG GAT
Examples	SOURce1: BURSt: MODE TRIGgered

selects triggered mode.

[SOURce1]:BURSt:NCYCLes

This command sets or queries the number of cycles (burst count) to be output in burst mode for the specified channel. The query command returns 9.9E+37 if the burst count is set to INFINITY.

Group	Source
Syntax	[SOURce1]: BURSt: NCYCLes {<cycles> INFi ni ty MI Ni mum MAXi mum} [SOURce1]: BURSt: NCYCLes? {MI Ni mum MAXi mum}
Arguments	<cycles>::=<NRf> where: <NRf> is the burst count. The burst count ranges from 1 to 50,000. INFINITY sets the burst count to infinite count. MINimum sets the burst count to minimum count. MAXimum sets the burst count to maximum count.
Returns	<cycles>
Examples	SOURce1: BURSt: NCYCLes 2 sets the CH 1 burst count to 2.

[SOURce1]:BURSt:STATe

This command enables or disables the burst mode for the specified channel. The query command returns the state of burst mode.

Group	Source
Syntax	[SOURce1]: BURSt: STATe {ON OFF <NR1>} [SOURce1]: BURSt: STATe?

Arguments	ON or <NR1>≠0 enables the burst mode. OFF or <NR1>=0 disables the burst mode.
Returns	<NR1>
Examples	SOURce1: BURSt: STATe ON enables the burst mode for the CH 1.

[SOURce1]:FM:STATe

This command enables or disables FM modulation. The query command returns the state of FM modulation.

Group	Source
Syntax	[SOURce1]: FM: STATe {ON OFF <NR1>} [SOURce1]: FM: STATe?
Arguments	ON or <NR1>≠0 enables FM modulation. OFF or <NR1>=0 disables FM modulation.
Returns	<NR1>
Examples	SOURce1: FM: STATe ON enables the CH 1 FM modulation.

[SOURce[1|2]]:FREQuency:CONCurent[:STATe]

This command enables or disables the function to copy the frequency (or period) of one channel to another channel.

The [SOURce[1 | 2]]: FREQuency: CONCurent command copies the frequency (or period) of the channel specified by the header suffix to another channel. If you specify CH 1 with the header, the CH 1 frequency will be copied to CH 2.

When the concurrent copy function is enabled, the FreqLock function is also enabled automatically. You can use general knob to adjust frequency (or period) of the two channels synchronously.

The `[SOURCE[1|2]]:FREQUENCY:CONCURRENT?` command returns “0” (off) or “1” (on).

Group	Source
Syntax	<code>[SOURCE[1 2]]:FREQUENCY:CONCURRENT</code> <code>{ON OFF <NR1>}</code> <code>[SOURCE[1 2]]:FREQUENCY:CONCURRENT?</code>
Arguments	ON or <code><NR1>≠0</code> enables the concurrent copy function. OFF or <code><NR1>=0</code> disables the concurrent copy function.
Returns	<code><NR1></code>
Examples	<code>SOURCE1:FREQUENCY:CONCURRENT ON</code> copies the frequency value of CH 1 to CH 2.

`[SOURCE[1|2]]:FREQUENCY[:CW]:FIXED`

This command sets or queries the frequency of output waveform for the specified channel. This command is available when the Run Mode is set to other than Sweep.

The setting range of output frequency depends on the type of output waveform. If you change the type of output waveform, it might change the output frequency because changing waveform types impacts on the setting range of output frequency. The resolution is 1 μ Hz or 12 digits. For more information on the setting range, refer to the *AFG1022 Arbitrary Function Generator Specifications and Performance Verification Technical Reference*.

Group	Source
Syntax	<code>[SOURCE[1 2]]:FREQUENCY[:CW]:FIXED</code> {<frequency> MINimum MAXimum} <code>[SOURCE[1 2]]:FREQUENCY[:CW]:FIXED?</code> {MINimum MAXimum}
Arguments	<code><frequency>::=<NRf>[<units>]</code>

where:

<NRf> is the output frequency.

<units>::=[Hz | kHz | MHz]

Returns <frequency>

Examples **SOURce1: FREQuency: FIXed 500kHz**

sets the CH 1 output frequency to 500 kHz when the Run Mode is set to other than Sweep.

[SOURce1]:FREQuency:MODE

This command sets or queries the frequency sweep state. You can select sine, square or ramp waveform for sweep.

Group Source

Syntax **[SOURce1]: FREQuency: MODE {CW|FIXed|SWEep}**
[SOURce1]: FREQuency: MODE?

Related Commands [\[SOURce\[1|2\]\]:FREQuency\[:CW\]:FIXed](#)

Arguments CW|FIXed means that the frequency is controlled by the [SOURce1]:FREQuency[:CW]:FIXed command. The sweep is invalid.

SWEep means that the output frequency is controlled by the sweep command set. The sweep is valid.

Returns CW|FIXed|SWEep

Examples **SOURCE1: FREQUENCY: MODE SWEEP** specifies the sweep command set for controlling the CH 1 output frequency.

[SOURCE1]:FSKey:STATe

This command enables or disables FSK modulation. The query command returns the state of FSK modulation. You can select a sine, square, ramp, or arbitrary waveform as the carrier waveform.

Group	Source
Syntax	[SOURCE1]:FSKey:STATe {ON OFF <NR1>} [SOURCE1]:FSKey:STATe?
Arguments	ON or <NR1>≠0 enables FSK modulation. OFF or <NR1>=0 disables FSK modulation.
Returns	<NR1>
Examples	SOURCE1:FSKey:STATe ON enables the CH 1 FSK modulation.

[SOURCE[1|2]]:FUNCTION:EFILe

This command sets or queries an EFILe name used as an output waveform. A file name must be specified in the mass storage system. This command returns “ ” if there is no file in the mass storage.

Group	Source
Syntax	[SOURCE[1 2]]:FUNCTION:EFILe <file_name> [SOURCE[1 2]]:FUNCTION:EFILe?
Arguments	<file_name>::=<string> specifies a file name in the mass storage system. The <file_name> includes path. Path separators are forward slashes (/).

NOTE: The <file_name> argument is case sensitive.

Returns <file_name>

Examples SOURCE1:FUNCTION:EFILe "SAMPLE1"
sets a file named "SAMPLE1" in the mass storage.

[SOURCE[1|2]]:FUNCTION[:SHAPE]

This command sets or queries the shape of the output waveform. When the specified user memory is deleted, this command causes an error if you select the user memory.

Group Source

Syntax [SOURCE[1|2]]:FUNCTION[:SHAPE] {SINusoid|SQUare|PULSe|RAMP|PRNoise|<Built_in>|USER[0]|USER1|...|USER255|EMEMory|EFILe}
[SOURCE[1|2]]:FUNCTION[:SHAPE]?

Arguments <Built_in>::={StairDown|StairUp|Stair Up&Dwn|Trapezoid|RoundHalf|AbsSine|AbsHalfSine|ClippedSine|ChoppedSine|NegRamp|OscRise|OscDecay|CodedPulse|PosPulse|NegPulse|ExpRise|ExpDecay|Sinc|Tan|Cotan|SquareRoot|X^2|HaverSine|Lorentz|Ln(x)|X^3|CauchyDistr|BesselJ|BesselY|ErrorFunc|Airy|Rectangle|Gauss|Hamming|Hanning|Bartlett|Blackman|Laylight|Triangle|DC|Heart|Round|Chirp|Rhombus|Cardiac}

NOTE: The arguments defined in <Built_in> can not be abbreviated, all the upper and lower case letters are needed.

The following table shows the combination of modulation type and the shape of output waveform.

	Sine, Square, Ramp	Pulse	Noise	Arb
AM	√			√
FM	√			√
PM	√			√
FSK	√			√
Sweep	√			
Burst	√	√		√

If you specify `EFILe` when there is no `EFILe` or the `EFILe` is not yet defined, this command causes an error.

If you change the type of output waveform, it might change the output frequency because changing waveform types impacts the setting range of output frequency.

`USER[0]|USER1|...|USER255|EMEMory`

A user defined waveform saved in the user waveform memory or the `EMEMory` can be selected as an output waveform.

`EFILe`

`EFILe` is specified as an output waveform.

Returns `SIN|SQU|PULS|RAMP|PRN|<Built_in>|`
`USER0|USER1|...|USER255|EMEMory|EFILe`

Examples `SOURce1: FUNCti on: SHAPe SQUare`
selects the shape of CH 1 output waveform to square waveform.

[SOURce[1|2]]:PHASe[:ADJust]

This command sets or queries the phase of output waveform for the specified channel. You can set the value in radians or degrees. If no units are specified, the default is RAD. The query command returns the value in RAD.

This command is supported when the `FreqLock` function is enabled. You can enable the `FreqLock` function using the `[SOURce[1|2]]:FREQuency:CONCurent[:STATe]` command.

Group Source

Syntax `[SOURce[1|2]]:PHASe[:ADJust] {<phase>|MI Ni mum|MAXi mum}`
`[SOURce[1|2]]:PHASe[:ADJust]? {MI Ni mum|MAXi mum}`

Related Commands [\[SOURce\[1|2\]\]:FREQuency:CONCurent\[:STATe\]](#)

Arguments `<phase>::=<NR3>[<units>]`
where:

<NR3> is the phase of output waveform.

<units>::=[RAD | DEG]

If <units> are omitted, RAD is specified automatically. The setting ranges are:

RAD: 0 to +2 PI, relative to phase value

DEG: 0 to +360, relative to phase value

Returns <phase>

Examples `SOURce1: PHASe: ADJust MAXi mum`
sets the maximum value for the phase of CH 1 output waveform.

[SOURce1]:PM:STATe

This command enables or disables PM modulation. The query command returns the state of PM modulation. You can select a sine, square, ramp, or arbitrary waveform as the carrier waveform.

Group Source

Syntax `[SOURce1]: PM: STATe {ON|OFF|<NR1>}`
`[SOURce1]: PM: STATe?`

Arguments ON or <NR1>≠0 enables PM modulation.
OFF or <NR1>=0 disables PM modulation.

Returns <NR1>

Examples `SOURce1: PM: STATe ON`
enables the CH 1 PM modulation.

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

This command sets or queries the offset level for the specified channel.

Group	Source
Syntax	[SOURce[1 2]]:VOLTage[:LEVel][:IMMediate]:OFFSet {<vol tage> MINi mum MAXi mum} [SOURce[1 2]]:VOLTage[:LEVel][:IMMediate]:OFFSet? {MINi mum MAXi mum}
Arguments	<voltage>::=<NRf>[<units>] where: <NRf> is the offset voltage level. <units>::=[mV V]
Returns	<vol tage>
Examples	SOURce1:VOLTage:LEVel:IMMediate:OFFSet 500mV sets the CH 1 offset level to 500 mV.

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

This command sets or queries the output amplitude for the specified channel.

Units	Amplitude resolution
Vpp	0.1 mVp-p or four digits

You can set the units of output amplitude by using the bezel menu selection.

Group	Source
Syntax	[SOURce[1 2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] {<ampl i tude> MINi mum MAXi mum} [SOURce[1 2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]? {MINi mum MAXi mum}

Arguments	<p><amplitude>::=<NRf>[<units>]</p> <p>where:</p> <p><NRf> is the output amplitude.</p> <p><units>::=[Vpp]</p>
Returns	<amplitude>
Examples	<p>SOURce1: VOLTage: LEVel: IMMEDIATE: AMPLi tude 1Vpp</p> <p>sets the CH 1 output amplitude to 1 Vpp.</p>

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

This query-only command returns the contents of the Error/Event queue.

Group	System
Syntax	SYSTem: ERRor[: NEXT]?
Arguments	None
Returns	<p><Error/event number>::=<NR1></p> <p><Error/event description>::=<string></p>
Examples	<p>SYSTEM: ERROR: NEXT?</p> <p>might return the following response:</p> <p>- 201, "Invalid while in local "</p> <p>If the instrument detects an error or an event occurs, the event number and event message will be returned.</p>

TRACe|DATA:CATalog? (Query Only)

This query-only command returns the names of user waveform memory and edit memory.

Group	Trace
Syntax	TRACe DATA:CATalog?
Arguments	None
Returns	<string> A series of strings separated by commas is returned. Each string is enclosed within quotation marks.
Examples	TRACe DATA:CATALOG? might return "USER0","USER4","EMEM"

TRACe|DATA:COPY (No Query Form)

This command copies the contents of edit memory (or user waveform memory) to a specified user waveform memory (or edit memory).

Group	Trace
Syntax	TRACe DATA: COPY <trace_name>, EMEMemory TRACe DATA: COPY EMEMemory, {USER[0] USER1 ... USER255}
Arguments	<trace_name>::={USER[0] USER1 ... USER255} This command is invalid when <trace_name> is being output.
Examples	DATA: COPY USER0, EMEMemory copies the waveform data in the edit memory to the user waveform memory USER0. DATA: COPY EMEMemory, USER0

copies the waveform data in the user waveform memory USER0 to the edit memory.

TRACe|DATA[:DATA]

This command transfers the waveform data from the external controller to the edit memory in the instrument. The query command returns the binary block data.

Group Trace

Syntax TRACe|DATA[: DATA] EMEMory, <bi nary_bl ock_data>
 TRACe|DATA[: DATA]? EMEMory

Arguments <binary_block_data>
 where <binary_block_data> is the waveform data in binary format.

Returns <bi nary_bl ock_dat a>

Examples DATA: DATA EMEMory, #42000<DAB><DAB>... <DAB>

transmits a waveform to the edit memory in the instrument. The block data element #42000 indicates that 4 is the number of digits in 2000 (byte count) and the 2000 bytes of binary data are to be transmitted.

TRACe|DATA[:DATA]:VALue

This command sets or queries the data value at the specified point in the edit memory.

Group Trace

Syntax TRACe|DATA[: DATA]: VALue EMEMory, <poi nt>, <data>
 TRACe|DATA[: DATA]: VALue? EMEMory, <poi nt>

Arguments <poi nt>: : =<NR1>
 where:

<NR1> is the specified point number in the edit memory.

<data>: : =<NR1>

where:

<NR1> is the data value for the specified point number.

Returns <NR1>

Examples DATA: DATA: VALue EMEMemory, 500, 2047

sets the data value to 2047 for the point number 500 in the edit memory.

DATA: DATA: VALue? EMEMemory, 500

might return "2047".

This example indicates that the data value of point number 500 is set to 2047.

TRACe|DATA:POINTS

This command sets or queries the number of data points for the waveform created in the edit memory.

Group Trace

Syntax TRACe|DATA: POINts EMEMemory[, <poi nts>|MI Ni mum|MAXi mum]
TRACe|DATA: POINts? EMEMemory{, MI N|MAX}

Arguments <poi nts>: : =<NR1>

where

<NR1> sets the number of points for the waveform created in the edit memory that ranges from 2 to 8192.

Returns <NR1>

Examples DATA: POINts EMEMemory, 500

sets the waveform data points to 500 in the edit memory.

*WAI (No Query Form)

This command prevents the instrument from executing further commands or queries until all pending commands that generate an OPC message are complete.

Group Synchronization

Syntax *WAI

Arguments None

Examples *WAI

prevents the instrument from executing any further commands or queries until all pending commands that generate an OPC message are complete.

Command Errors

The following table shows the error messages generated by improper command syntax. Check that the command is properly formed and that it follows the rules in the Syntax and Commands.

Table 12: Command messages

Code	Message
0 (indicates no error)	
-101	Invalid character
-102	Syntax error
-108	Parameter not allowed
-201	Invalid while in local

Index

C

*CLS, 13

I

*IDN?, 13

M

MMEMory:CATalog?, 14

MMEMory:CDIRectory, 15

MMEMory:DELeTe, 16

O

*OPT?, 16

OUTPut[1|2][:STATe], 17

R

*RST, 17

S

[SOURce1]:AM:STATe, 18

[SOURce1]:BURSt:MODE, 18

[SOURce1]:BURSt:NCYCles, 19

[SOURce1]:BURSt:STATe, 19

[SOURce1]:FM:STATe, 20

[SOURce1]:FREQuency:MODE, 22

[SOURce[1|2]]:FREQuency:CONCurrenT[:STATe], 20

[SOURce[1|2]]:FREQuency[:CW|:FIXed], 21

[SOURce1]:FSKey:STATe, 23

[SOURce[1|2]]:FUNCTion:EFILe, 23

[SOURce[1|2]]:FUNCTion[:SHAPE], 24

[SOURce[1|2]]:PHASe[:ADJust], 25

[SOURce1]:PM:STATe, 26

[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:
OFFSet, 26

[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]
[: AMPLitude], 27

SYSTem:ERRor[:NEXT]?, 28

T

TRACe|DATA:CATalog?, 29

TRACe|DATA:COPIY, 29

TRACe|DATA:POINts, 31

TRACe|DATA[:DATA], 30

TRACe|DATA[:DATA]:VALue, 30

W

*WAI, 32