Tektronix®

AFG31000 Series Arbitrary Function Generator Programmer's Manual

077-148-803

Tektronix®

AFG31000 Series Arbitrary Function Generator

Programmer's Manual

tek.com

077148803

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 tek.com to find contacts in your area.

Table of Contents

Important safety information xi

Important safety information xi

General safety summary xi

To avoid fire or personal injury xi

Terms in this manual xiii

Service safety summary xiii

Symbols and terms on the product xiv

Introduction 1

Getting started 1

Extended warranty 1

General model information 1

Manual overview 2

Contact information 2

Using the GPIB port 2

GPIB requirements 3

Setting the GPIB address 4

Using Virtual Instrument Software Architecture 4

Installation 5

Syntax and commands 7

Syntax and commands 7

Command syntax 7

Backus-Naur form definition 7

Command and query structure 8

Command message elements 9

SCPI commands and queries 10

IEEE 488.2 common commands 15

Command groups 16

Calibration and diagnostic commands 16

Display commands 16

Mass memory commands 17

Memory commands 18

Output commands 18

Source commands 19

Status commands 21

System commands 22

Synchronization commands 22

Trace commands 22

Trigger commands 23

AFG control command 23
Screen copy command 23
License commands 24
Sequence commands 25
Waveform list commands 25
Sequence control group commands 26
InstaView commands 26

Command descriptions 27

Command descriptions 27
Manual conventions 27
ABORt (no query form) 27
AFGControl:CSCopy (no query form) 28
*CAL? (query only) 29
CALibration[:ALL] 30
*CLS (no query form) 31
DIAGnostic[:ALL] 31
DISPlay:BRIGhtness 32
DISPlay:SAVer:IMMediate (no query form) 32
DISPlay:SAVer[:STATe] 33
DISPlay[:WINDow]:TEXT[:DATA] 33
DISPlay[:WINDow]:TEXT:CLEar (no query form) 34
*ESE 35
*ESR? (query only) 36
HCOPy:SDUMp[:IMMediate] (no query form) 36
*IDN? (query only) 37
INSTaview[1|2]:MEASUrement:DELay 37
INSTaview[1|2]:MEASUrement:MAXimum (query only) 38
INSTaview[1|2]:MEASUrement:MINimum (query only) 38
INSTaview[1|2]:STATe 39
LICense:HID? (query only) 39
LICense: INSTall (no query form) 40
LICense: UNINstall? 40
LICense:LIST? (query only) 41
LICense:ERRor? (query only) 41
LICense: GMT? (query only) 42
LICense: IDN? (query only) 42
LICense: COUNT? (query only) 43
LICense: ITEM? (query only) 44
LICense: VALidate? (query only) 45
MEMory:STATe:DELete (no query form) 45
MEMory:STATe:LOCK 46
MEMory:STATe:RECall:AUTo 47
MEMory:STATe:VALid? (query only) 48
MMEMory:CATalog? (query only) 49
MMEMory:CDIRectory 50
MMEMory:DELete (no query form) 50
MMEMory:LOAD:STATe (no query form) 51
MMEMory:LOAD:TRACe (no query form) 52
MMEMory:LOCK[:STATe] 53
MMEMory:MDIRectory (no query form) 53
MMEMory:STORe:STATe (no query form) 54
MMEMory:STORe:TRACe (no query form) 55
MMEMory:OPEN:SEQuence (no query form) 56
MMEMory:SAVE:SEQuence (no query form) 57
*OPC 58
*OPT? (query only) 58
OUTPut[1|2]:IMPedance 59
OUTPut[1|2]:POLarity 60
OUTPut[1|2][:STATe] 61

OUTPut:TRIGger:MODE....62
*PSC....63
*RCL (no query form)....64
RECALL:SETUp (no query form)....64
*RST (no query form)....65
*SAV (no query form)....65
SAVe:SETUp (no query form)....66
SEQControl:RMODe....67
SEQControl:RUN[:IMMediate] (no query form)....68
SEQControl:STOP[:IMMediate] (no query form)....68
SEQControl:RESET[:IMMediate] (no query form)....69
SEQControl:STATe....69
SEQControl:RSTATe? (query only)....70
SEQControl:SRATe....70
SEQControl:DELay....71
SEQControl:SOURce[n]:SCALe....72
SEQControl:SOURce[n]:OFFSet....73
SEQControl:TIMer....74
SEQuence:ELEM[n]:GOTO:INDex....75
SEQuence:ELEM[n]:GOTO:STATe....76
SEQuence:ELEM[n]:MARKer:STATe....77
SEQuence:ELEM[n]:JTARget:INDex....78
SEQuence:ELEM[n]:JTARget:TYPE....79
SEQuence:ELEM[n]:JUMP:EVENt....80
SEQuence:ELEM[n]:JUMP:SLOPe....81
SEQuence:ELEM[n]:LOOP:COUNt....82
SEQuence:ELEM[n]:LOOP:INFinite....83
SEQuence:ELEM[n]:TWAit[:STATe]....84
SEQuence:ELEM[n]:TWAit:EVENt....85
SEQuence:ELEM[n]:TWAit:SLOPe....86
SEQuence:ELEM[n]:WAVeform[m]....87
SEQuence:LENGth....88
SEQuence:NEW (No Query Form)....88
[SOURce[1|2]]:AM[:DEPTh]....89
[SOURce[1|2]]:AM:INTernal:FREQuency....90
[SOURce[1|2]]:AM:INTernal:FUNCtion....91
[SOURce[1|2]]:AM:INTernal:FUNCtion:EFILe....92
[SOURce[1|2]]:AM:SOURce....93
[SOURce[1|2]]:AM:STATe....94
[SOURce[1|2]:]BURSt:INFInite:REARm (no query form)....94
[SOURce[1|2]:]BURSt:IDLE....95
[SOURce[1|2]]:BURSt:MODE....96
[SOURce[1|2]]:BURSt:NCYCles....97
[SOURce[1|2]]:BURSt[:STATe]....98
[SOURce[1|2]]:BURSt:TDELay....99
[SOURce[1|2]]:COMBine:FEED....100
[SOURce[1|2]]:FM[:DEViation]....101
[SOURce[1|2]]:FM:INTernal:FREQuency....102
[SOURce[1|2]]:FM:INTernal:FUNCtion....103
[SOURce[1|2]]:FM:INTernal:FUNCtion:EFILe....104
[SOURce[1|2]]:FM:SOURce....104
[SOURce[1|2]]:FM:STATe....105
[SOURce[1|2]]:FREQuency:CENTer....106
[SOURce[1|2]]:FREQuency:CONCurrent[:STATe]....107
[SOURce[1|2]]:FREQuency[:CW|:FIXed]....108
[SOURce[1|2]]:FREQuency:MODE....109
[SOURce[1|2]]:FREQuency:SPAN....110
[SOURce[1|2]]:FREQuency:STARt....111
[SOURce[1|2]]:FREQuency:STOP....112
[SOURce[1|2]]:FSKey[:FREQuency]....113
[SOURce[1|2]]:FSKey:INTernal:RATE....114
[SOURce[1|2]]:FSKey:SOURce....115

[SOURce[1|2]]:FSKey:STATe 116
[SOURce[1|2]]:FUNCtion:EFILe....117
[SOURce[1|2]]:FUNCtion:RAMP:SYMMetry 118
[SOURce[1|2]]:FUNCtion[:SHAPe]....119
[SOURce[1|2]]:PHASe[:ADJust]....120
[SOURce[1|2]]: PHASe:CONCurrent[:STATe] 121
[SOURce[1|2]]:PHASe:INITiate (no query form) 121
[SOURce[1|2]]:PM[:DEViation]....122
[SOURce[1|2]]:PM:INTernal:FREQuency....123
[SOURce[1|2]]:PM:INTernal:FUNCtion 124
[SOURce[1|2]]:PM:INTernal:FUNCtion:EFILe....125
[SOURce[1|2]]:PM:SOURce....125
[SOURce[1|2]]:PM:STATe....126
[SOURce[1|2]]:PULSe:DCYCle....127
[SOURce[1|2]]:PULSe:DELay 128
[SOURce[1|2]]:PULSe:HOLD....129
[SOURce[1|2]]:PULSe:PERiod....130
[SOURce[1|2]]:PULSe:TRANsition[:LEADing]....131
[SOURce[1|2]]:PULSe:TRANsition:TRAiling 132
[SOURce[1|2]]:PULSe:WIDTh....133
[SOURce[1|2]]:PWM:INTernal:FREQuency 134
[SOURce[1|2]]:PWM:INTernal:FUNCtion....135
[SOURce[1|2]]:PWM:INTernal:FUNCtion:EFILe 136
[SOURce[1|2]]:PWM:SOURce 136
[SOURce[1|2]]:PWM:STATe 137
[SOURce[1|2]]:RMODe? 137
[SOURce[1|2]]:PWM[:DEViation]:DCYCle....138
[SOURce]:ROSCillator:SOURce 139
[SOURce[1|2]]:SWEep:HTIMe 140
[SOURce[1|2]]:SWEep:MODE 141
[SOURce[1|2]]:SWEep:RTIMe 142
[SOURce[1|2]]:SWEep:SPACing 143
[SOURce[1|2]]:SWEep:TIME 144
[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]....145
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:HIGH....146
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:LOW 147
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:OFFSet 148
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] 149
[SOURce[1|2]]:VOLTage:LIMit:HIGH 150
[SOURce[1|2]]:VOLTage:LIMit:LOW....151
[SOURce[1|2]]:VOLTage:UNIT 152
SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude]....153
*SRE 154
STATus:OPERation:CONDition? (query only) 155
STATus:OPERation:ENABle....155
STATus:OPERation[:EVENt]? (query only)....156
STATus:PRESet (no query form) 156
STATus:QUEStionable:CONDition? (query only)....157
STATus:QUEStionable:ENABle 157
STATus:QUEStionable[:EVENt]? (query only) 158
*STB? (query only)....158
SYSTem:BEEPer:STATe....159
SYSTem:ERRor[:NEXT]? (query only)....159
SYSTem:BEEPer[:IMMediate] (no query form) 160
SYSTem:KCLick[:STATe] 160
SYSTem:KLOCk[:STATe] 161
SYSTem:MACADDress?....162
SYSTem:PASSword:CDISable (no query form)....163
SYSTem:PASSword[:CENable] (no query form)....164
SYSTem:PASSword[:CENable]:STATe? (query only) 165
SYSTem:PASSword:NEW (no query form)....166
SYSTem:RESTart....166

SYSTem:SECurity:IMMediate (no query form)....167
SYSTem:ULANguage....167
SYSTem:VERSion? (query only)....168
TRACe|DATA[:DATA]....168
*TRG (no query form)....169
TRIGger[:SEQuence]:SLOPe....169
TRIGger[:SEQuence]:SOURce....170
TRIGger[:SEQuence]:TIMer....170
TRIGger[:SEQuence][:IMMediate] (no query form)....171
*TST? (query only)....171
*WAI (no query form)....172
WLISt:NAME? (query only)....172
WLISt:SIZE? (query only)....173
WLISt:WAVeform:DELete (no query form)....174
WLISt:WAVeform:IMPort (no query form)....175
WLISt:WAVeform:LENGth? (query only)....176

Status and events....177

Status and events....177
Status reporting structure....177
Registers....178
Queues....184
Messages and codes....185

Programming examples....193

Programming examples....193

Example 1....194

Example 2....195

Default settings....196

Important safety information

Important safety information

This manual contains information and warnings that must be followed by the user for safe operation and to keep the product in a safe condition.

To safely perform service on this product, additional information is provided at the end of this section. See Service safety summary (on page xiii).

General safety summary

Use the product only as specified. Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. Carefully read all instructions. Retain these instructions for future reference.

Comply with local and national safety codes.

For correct and safe operation of the product, it is essential that you follow generally accepted safety procedures in addition to the safety precautions specified in this manual.

The product is designed to be used by trained personnel only.

Only qualified personnel who are aware of the hazards involved should remove the cover for repair, maintenance, or adjustment.

Before use, always check the product with a known source to be sure it is operating correctly.

This product is not intended for detection of hazardous voltages.

Use personal protective equipment to prevent shock and arc blast injury where hazardous live conductors are exposed.

While using this product, you may need to access other parts of a larger system. Read the safety sections of the other component manuals for warnings and cautions related to operating the system.

When incorporating this equipment into a system, the safety of that system is the responsibility of the assembler of the system.

To avoid fire or personal injury

Use proper power cord. Use only the power cord specified for this product and certified for the country of use.

Do not use the provided power cord for other products.

Use proper voltage setting. Before applying power, make sure that the line selector is in the proper position for the source being used or make sure the line voltage is corrected based on the published specifications.

Ground the product. This product is grounded through the grounding conductor of the power cord. To avoid electric shock, the grounding conductor must be connected to earth ground. Before making connections to the input or output terminals of the product, make sure that the product is properly grounded.

Do not disable the power cord grounding connection.

Power disconnect. The power cord disconnects the product from the power source. See instructions for the location. Do not position the equipment so that it is difficult to operate the power cord; it must remain accessible to the user at all times to allow for quick disconnection if needed.

Connect and disconnect properly. Do not connect or disconnect instruments while they are connected to a voltage source.

Use only connectors and adapters supplied with the product, or indicated by Tektronix to be suitable for the product.

Observe all terminal ratings. To avoid fire or shock hazard, observe all ratings and markings on the product. Consult the product manual for further ratings information before making connections to the product. Do not exceed the Measurement Category (CAT) rating and voltage or current rating of the lowest rated individual component.

Do not apply a potential to any terminal, including the common terminal, that exceeds the maximum rating of that terminal.

Do not float the common terminal above the rated voltage for that terminal.

Do not operate without covers. Do not operate this product with covers or panels removed, or with the case open. Hazardous voltage exposure is possible.

Avoid exposed circuitry. Do not touch exposed connections and components when power is present.

Do not operate with suspected failures. If you suspect that there is damage to this product, have it inspected by qualified service personnel.

Disable the product if it is damaged. Do not use the product if it is damaged or operates incorrectly. If in doubt about safety of the product, turn it off and disconnect the power cord. Clearly mark the product to prevent its further operation.

Examine the exterior of the product before you use it. Look for cracks or missing pieces.

Use only specified replacement parts.

Wear eye protection. Wear eye protection if exposure to high-intensity rays or laser radiation exists.

Do not operate in wet/damp conditions. Be aware that condensation may occur if a unit is moved from a cold to a warm environment.

Do not operate in an explosive atmosphere.

Provide proper ventilation. Refer to the installation instructions in the manual for details on installing the product so it has proper ventilation.

Slots and openings are provided for ventilation and should never be covered or otherwise obstructed. Do not push objects into any of the openings.

Provide a safe working environment. Always place the product in a location convenient for viewing the display and indicators.

Avoid improper or prolonged use of keyboards, pointers, and button pads. Improper or prolonged keyboard or pointer use may result in serious injury.

Be sure your work area meets applicable ergonomic standards. Consult with an ergonomics professional to avoid stress injuries.

Use care when lifting and carrying the product. This product is provided with handles for lifting and carrying.

Use only the Tektronix rack-mount hardware specified for this product.

Keep product surfaces clean and dry. Remove the input signals before you clean the product. Inspect the instrument as often as operating conditions require. To clean the exterior surface, perform the following steps:

1. Remove loose dust on the outside of the instrument with a lint-free cloth. Use care to avoid scratching the clear glass display filter.
2. Use a soft cloth dampened with water to clean the instrument. Use an aqueous solution of 75% isopropyl alcohol for more efficient cleaning.

CAUTION. Avoid getting moisture inside the unit during external cleaning. Use only enough cleaning solution to dampen the cloth or swab. To avoid damage to the instrument, do not expose it to sprays, liquids, or solvents, and do not use any abrasive or chemical cleaning agents.

Terms in this manual

These terms may appear in this manual:

WARNING. Warning statements identify conditions or practices that could result in injury or loss of life.

CAUTION. Caution statements identify conditions or practices that could result in damage to this product or other property.

Service safety summary

This section contains additional information required to safely perform service on the product. Only qualified personnel should perform service procedures. Read this *Service safety summary* and the General safety summary before performing any service procedures.

To avoid electric shock. Do not touch exposed connections.

Do not service alone. Do not perform internal service or adjustments of this product unless another person capable of rendering first aid and resuscitation is present.

Disconnect power. To avoid electric shock, switch off the product power and disconnect the power cord from the mains power before removing any covers or panels, or opening the case for servicing.

Use care when servicing with power on. Dangerous voltages or currents may exist in this product. Disconnect power, remove battery (if applicable), and disconnect test leads before removing protective panels, soldering, or replacing components.

Verify safety after repair. Always recheck ground continuity and mains dielectric strength after performing a repair.

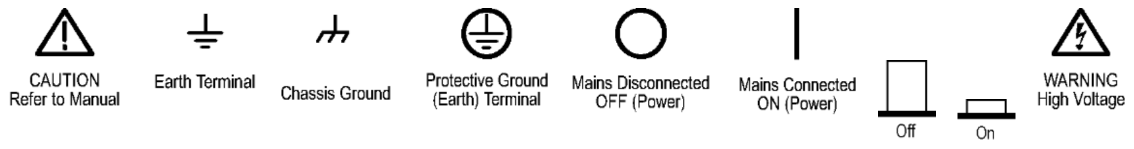
Symbols and terms on the product

These terms may appear on the product:

- DANGER indicates an injury hazard immediately accessible as you read the marking.
- WARNING indicates an injury hazard not immediately accessible as you read the marking.
- CAUTION indicates a hazard to property including the product.

When this symbol is marked on the product, be sure to consult the manual to find out the nature of the potential hazards and any actions which have to be taken to avoid them. (This symbol may also be used to refer the user to ratings in the manual.)

The following symbol(s) may appear on the product:


CAUTION Refer to Manual
Earth Terminal
Chassis Ground
Protective Ground (Earth) Terminal
Mains Disconnected OFF (Power)
Mains Connected ON (Power)
Off
On
WARNING High Voltage

Introduction

Getting started

Thank you for choosing a Tektronix product. The AFG31000 Series Arbitrary Function Generator (AFG) instruments are high-performance instruments with built-in waveform generation applications, real-time waveform monitoring called InstaView^{TM}, and an improved user interface for higher test efficiency.

Extended warranty

Additional years of warranty coverage are available on many products. These valuable contracts protect you from unbudgeted service expenses and provide additional years of protection at a fraction of the price of a repair. Extended warranties are available on new and existing products. Contact your local Tektronix office, sales partner, or distributor for details (refer to Contact information (on page 2) for details).

General model information

This manual provides operation information for the following products. Unless otherwise noted, "AFG31000 Series" refers to the models in the following table.

Table 1: AFG31000 models

Model	Bandwidth	Sample rate	Channel	Waveform memory size	Optional
AFG31021	25 MHz	250 MS/s	1	16 MS/CH	128 MS/CH
AFG31022	25 MHz	250 MS/s	2	16 MS/CH	128 MS/CH
AFG31051	50 MHz	1 GS/s	1	16 MS/CH	128 MS/CH
AFG31052	50 MHz	1 GS/s	2	16 MS/CH	128 MS/CH
AFG31101	100 MHz	1 GS/s	1	16 MS/CH	128 MS/CH
AFG31102	100 MHz	1 GS/s	2	16 MS/CH	128 MS/CH
AFG31151	150 MHz	2 GS/s	1	16 MS/CH	128 MS/CH
AFG31152	150 MHz	2 GS/s	2	16 MS/CH	128 MS/CH
AFG31251	250 MHz	2 GS/s	1	16 MS/CH	128 MS/CH
AFG31252	250 MHz	2 GS/s	2	16 MS/CH	128 MS/CH

Each AFG31000 Series provides a:

- 25 MHz to 250 MHz function signal generator
- 20 MHz to 160 MHz pulse generator

The AFG31000 Series also provides 14-bit vertical resolution.

Manual overview

This manual consists of the following sections:

- **Syntax and Commands:** defines the command syntax and processing conventions, and describes command notation.
- **Status and Events:** explains the status information and event messages reported by the instrument.
- **Programming examples:** contains remote interface application programs to help you develop programs for your application.
- **Supported SCPI commands:** contains a list of commands and SCPI information.

Contact information

If you have any questions after you review the information in this documentation, please contact your local Tektronix office, sales partner, or distributor. You can also call the corporate headquarters of Tektronix in North America at 1-800-833-9200. Visit tek.com to find contacts in your area.

Using the GPIB port

The arbitrary function generator has Talk and Listen functions through which it can communicate with other devices, as well as the external controller, using a GPIB cable.

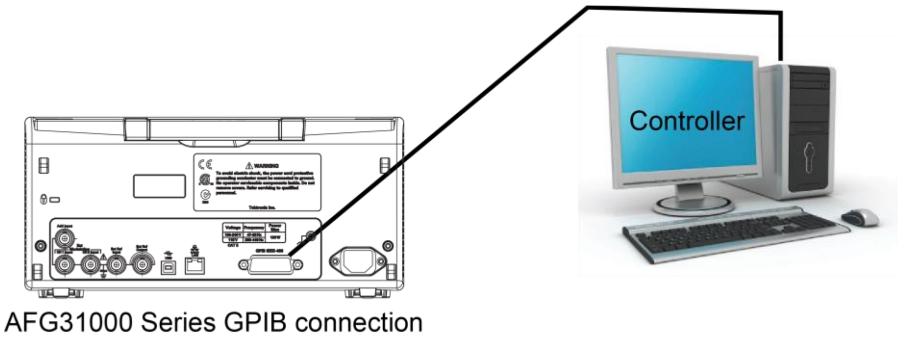


Figure 1: AFG31000 GPIB connection

GPIB requirements

Observe the following rules when you use your arbitrary function generator with a GPIB cable:

- Assign a unique device address to each device that uses the GPIB cable (this is also known as the bus). No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices to a bus.
- Turn on at least two-thirds, or 67%, of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration, as shown in the next figure. Do not use parallel or loop configurations.

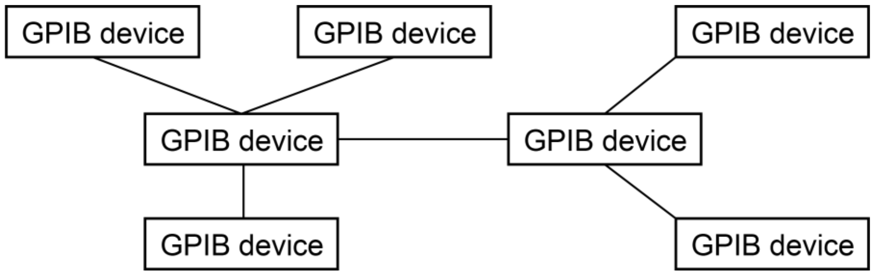


Figure 2: Typical GPIB network configuration

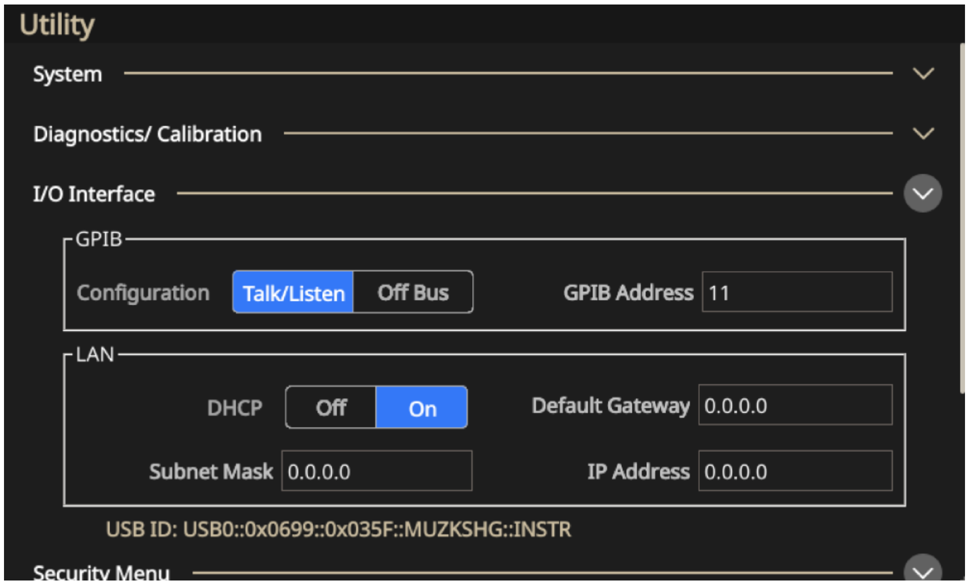
Setting the GPIB address

When you use the GPIB port to communicate with an external controller, follow these steps to set the address of the arbitrary function generator.

To set the GPIB address:

1. Select **Utility**.
2. Select **I/O Interface**.
3. Select the **GPIB Address text box**.
4. Input the correct address and select **Enter** on the touchscreen keyboard.

NOTE. The GPIB address cannot be initialized by an *RST command.



Configuration: Talk/Listen | Off Bus (Talk/Listen selected)
GPIB Address: 11
DHCP: Off | On (On selected)
Default Gateway: 0.0.0.0
Subnet Mask: 0.0.0.0
IP Address: 0.0.0.0
USB ID: USB0::0x0699::0x035F::MUZKSHG::INSTR

Figure 3: Set the GPIB address

Using Virtual Instrument Software Architecture

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 GPIB and serial (RS-232) instrument interfaces locally or remotely via an Ethernet LAN connection.

Installation

Use an internet browser to access the Tektronix Web site (www.tektronix.com/downloads) 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 about TekVISA concepts and operations are explained in the **TekVISA Programmer Manual** that can also be found on the Tektronix website. This manual describes TekVISA, the Tektronix implementation of the VISA Application Programming Interface (API). TekVISA provides the interface-independent functionality needed to control and access the embedded software of Tektronix test and measurement equipment in following ways: Using virtual GPIB software running locally on Windows-based instruments using physical GPIB controller hardware using asynchronous serial controller hardware.

Syntax and commands

Syntax and commands

This section provides the following information:

- Command Syntax (on page 7) defines the command syntax and processing conventions.
- Command Groups (on page 16) describes command groups which lists the commands by function.
- Command Descriptions (on page 27) describes the notation of each of the commands in alphabetical order.

Command syntax

You can control the operations and functions of the AFG through the GPIB 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 16)** for a list of the commands by command group.

Backus-Naur form definition

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

Table 2: Table: BNF symbols and meanings

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

Command and query structure

Commands consist of set commands and query commands. 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. If the set form parameter has a minimum and maximum, it will also support a query.

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 `DISPlay:BRIGhtness` has a query form `DISPlay:BRIGhtness?`. Not all commands have both a set and a query form; some commands are set only and some are query only.

A few commands do 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. 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.

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 3: Commands, queries, and query response definitions

Commands:	Commands cause the instrument to perform a specific function or change one of its settings. Commands have the following structure:
	`[:]<Header>[<Space><Argument>[<Comma><Argument>]...]`
Queries:	Queries cause the AFG to return information about its status or settings. Queries have the following 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.
Query Responses:	When a query is sent to the AFG, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format.

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 (:); 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 (:).
<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 <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.

The following figure shows the five command message elements.

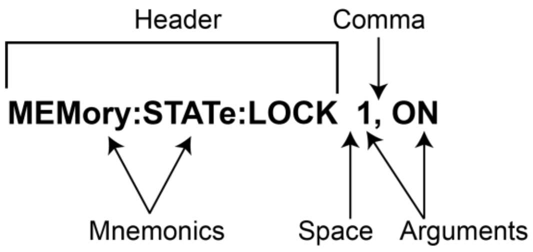


Figure 4: Command message elements

Command entry

Follow these general rules when entering commands:

- Enter commands in upper or lower case.
- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 decimal).
- The instrument ignores commands that consists of just a combination of white space characters and line feeds.

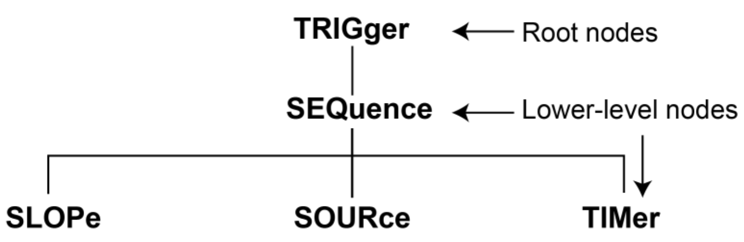
SCPI commands and queries

The AFG 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 next 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.

Example of SCPI subsystem hierarchy tree

Figure 5: SCPI command structure



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 previous figure, `TRIGger` is the root node and `EVENt`, `GATed`, `INPut`, and `SOURce` 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:EVENt:SOURce EXTernal` is a valid SCPI command created from the hierarchy tree (see previous figure).

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:EVENt:SOURce?` is an example of a valid SCPI query using the hierarchy tree in the next figure.

Query responses

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

Table 4: Query response examples

Symbol	Meaning
`SOURce:PULSe:DCYcle?`	50.0
`OUTPut:POLarity?`	NORM

Parameter types

Every parameter in the command and query descriptions is of a specified type. Parameters are indicated by angle brackets, such as <file_name>. There are several different types of parameters. 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 or function generator command set and some are defined by SCPI.

Parameter types used in syntax descriptions

Parameter type	Description	Example
arbitrary block*	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><&EOI>
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**	Hexadecimal numbers (0-9, A, B, C, D, E, F)	#HAA, #H1
NR1** numeric	Integers	0, 1, 15, -1
NR2***, numeric	Decimal numbers	1.2, 3.141516, -6.5
NR3** numeric	Floating point numbers	3.1415E-9, -16.1E5
NRf** numeric	Flexible decimal number 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"

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

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

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

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

Special characters

The Line Feed (LF) 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 next figure, you can 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.

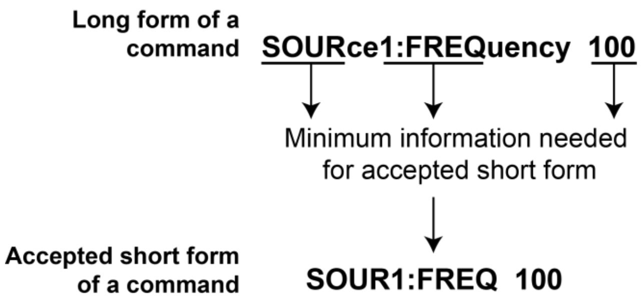

Long form of a command
SOURce1:FREQuency 100
Minimum information needed for accepted short form
Accepted short form of a command
SOUR1:FREQ 100

Figure 6: 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 AFG 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 next figure illustrates a chained message consisting of several commands and queries. 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.

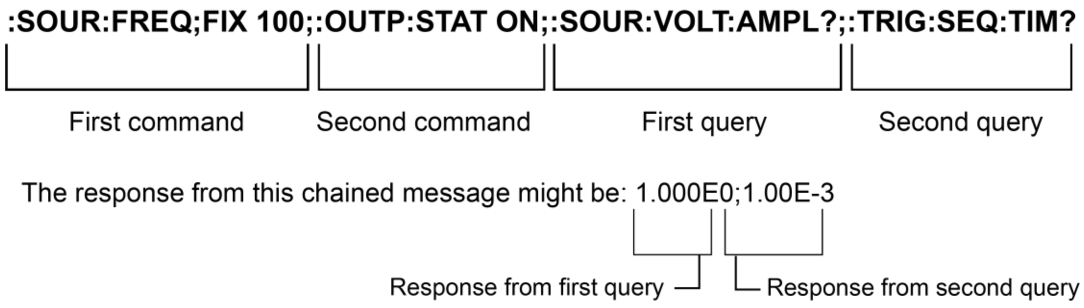

:SOUR:FREQ;FIX 100;:OUTP:STAT ON;:SOUR:VOLT:AMPL?;:TRIG:SEQ:TIM?
First command
Second command
First query
Second query
The response from this chained message might be: 1.000E0;1.00E-3
Response from first query
Response from second query

Figure 7: 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 next figure, the second command has the same root node (`TRIGger:SEQuence`) as the first command, so these nodes can be omitted.

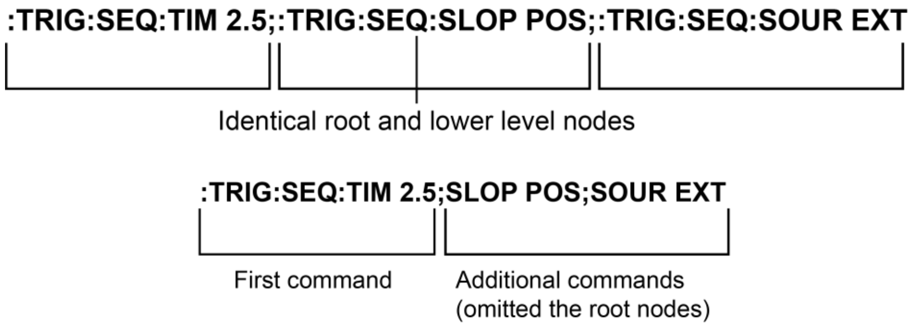

:TRIG:SEQ:TIM 2.5;:TRIG:SEQ:SLOP POS;:TRIG:SEQ:SOUR EXT
Identical root and lower level nodes
:TRIG:SEQ:TIM 2.5;SLOP POS;SOUR EXT
First command
Additional commands (omitted the root nodes)

Figure 8: Example of omitting root and lower-level nodes in a chained message

Unit and SI prefix

If the decimal numeric argument refers to amplitude, frequency, or time, you can express it using the International System of Units (SI) 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, 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.

Table 5: Available units

Symbol	Meaning
dB	decibel (relative amplitude)
dBm	decibel (absolute amplitude)
DEG	degree (phase)
Hz	hertz (frequency)
PCT	percent (%)
s	second (time)
V	volt

Table 6: Available SI prefixes

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

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 the SI prefix. For example, frequency of 15 MHz can be described as follows:

- 15.0E6, 1.5E7 Hz, 15000000, 15000000 Hz, 15MHz, etc. ("15 M" is not allowed.)

You can use either lower or upper case units and prefixes. The next examples have the same result, respectively.

- 170 mHz, 170 mHz, 170 MHz, etc.
- 250 mv, 250 mV, 250 MV, etc.

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.

```
: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.

No embedded spaces are allowed between or within nodes.

- correct `:OUTPUT:FILTER:LPASS:FREQUENCY 200MHZ`
- incorrect `:OUTPUT: FILTER: LPASS:FREQUENCY 200MHZ`

IEEE 488.2 common commands

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 AFG complies with this standard.

Command and query structure

The syntax for IEEE 488.2 common commands (on page 15) 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 Command groups (on page 16) and Command descriptions (on page 27). The following are examples of common commands:

- *ESE 16
- *CLS

The following are examples of common queries:

- *ESR?
- *IDN?

Command groups

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

Calibration and diagnostic commands

Calibration and Diagnostic commands let you initiate the instrument self-calibration routines and examine the results of diagnostic tests. The following table describes the calibration and diagnostic commands.

Header	Description
`*CAL?` (on page 29)	Perform self-calibration and return result status
`CALibration[:ALL]` (on page 30)	Perform self-calibration
`DIAGnostic[:ALL]` (on page 31)	Perform self-test
`*TST?` (on page 171)	Perform self-test and return result status

Display commands

Display commands let you change the graticule style, displayed brightness, and other display attributes. The next table lists and describes Display commands.

Header	Description
`Display:BRIGhtness` (on page 32)	Set or query the LCD display brightness
`DISPlay:SAVer:IMMediate` (on page 32)	Set screen saver
`DISPlay:SAVer[:STATe]` (on page 33)	Set or query the screen saver settings
`DISPlay[:WINDow]:TEXT[:DATA`(on page 33)	Set or query the text message display
`DISPlay[:WINDow]:TEXT:CLEar` (on page 34)	Delete text message

Mass memory commands

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

The instrument stores waveforms and setups to the USB drive or internal flash memory. That means the commands must identify the full path including the drive letter and the folder path; a virtual disk is used.

Virtual Disk:

- U: USB mass memory
- M: Internal flash mass memory
- P: Internal flash mass memory where predefined waveforms are stores, This is read-only storage.

For example, `MMEMORY:DELETE “U:/TEK001.TFWX`, deletes TEK001.TFWX in the USB memory root folder.

And relative folder is supported, for example:

```
MMEMory:CDIRectory “U:/”
MMEMORY:DELETE “TEK001.TFWX”
```

The above two commands produce the same results as

```
MMEMORY:DELETE “U:/TEK001.TFWX.”
```

Header	Description
MMEMory:CATalog? (on page 49)	Query the status of mass memory
MMEMory:CDIRectory (on page 50)	Set/query current directory
MMEMory:DELete (on page 50)	Delete file or directory in mass memory
MMEMory:LOAD:STATe (on page 52)	Copy instrument setting in mass memory to setup memory
MMEMory:LOAD:TRACe (on page 52)	Copy waveform data file in mass memory to edit memory
MMEMory:LOCK[:STATe] (on page 53)	Set/query the lock of mass memory overwrite and deletion
MMEMory:MDIRectory (on page 53)	Create directory in mass memory
MMEMory:STORe:STATe (on page 54)	Save the setup memory status to mass memory
MMEMory:STORe:TRACe (on page 55)	Save waveform data file in edit memory to mass memory
MMEMory:OPEN:SEQuence (on page 56)	Loads a single desired sequence from a file
MMEMory:SAVE:SEQuence (on page 57)	Exports a sequence given a unique name to an eligible storage location as a .SEQ file type.
RECALL:SETUp (on page 64)	Load a settings file in mass memory
SAVe:SETUp (on page 66)	Save a settings file to msss memory

Memory commands

Memory commands let you change setup memory attributes. The next table lists and describes Memory commands.

Header	Description
MEMory:STATe:DELete (on page 45)	Delete the setup memory
MEMory:STATe:LOCK (on page 46)	Set or query the lock of setup memory overwrite and deletion
MEMory:STATe:RECall:AUTo (on page 47)	Set or query the recall of last set memory
MEMory:STATe:VALid? (on page 48)	Query the availability of setup memory
*RCL (on page 64)	Recall instrument setting from setup memory
*SAV (on page 65)	Save instrument setting to setup memory

Output commands

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

Header	Description
OUTPut[1|2]:IMPedance (on page 59)	Set or query impedance
OUTPut[1|2]:POLarity (on page 60)	Set or query polarity
OUTPut[1|2][:STATe] (on page 61)	Set or query output on or off
OUTPut:TRIGger:MODE (on page 62)	Set or query the mode of Trigger Output

Source commands

Source commands let you set waveform output parameters. The next table lists and describes Source commands.

Header	Description
[SOURce]:ROSCillator:SOURce (on page 139)	Set or query clock reference input
[SOURce[1|2]]:AM[:DEPTh] (on page 89)	Set or query amplitude modulation depth
[SOURce[1|2]]:AM:INTernal:FREQuency (on page 90)	Set or query internal modulation frequency
[SOURce[1|2]]:AM:INTernal:FUNCtion (on page 91)	Set or query modulation waveform setting
[SOURce[1|2]]:AM:INTernal:FUNCtion: EFILe (on page 92)	Set or query EFILe setting
[SOURce[1|2]]:AM:SOURce (on page 93)	Set or query amplitude modulation source
[SOURce[1|2]]:AM:STATe (on page 94)	Set or query amplitude modulation status
[SOURce[1|2]]:BURSt:MODE (on page 96)	Set or query burst mode
[SOURce[1|2]]:BURSt:NCYCles (on page 97)	Set or query burst mode waveform output cycle
[SOURce[1|2]]:BURSt[:STATe] (on page 98)	Set or query burst mode status
[SOURce[1|2]]:BURSt:TDELay (on page 99)	Set or query burst mode trigger delay time
[SOURce[1|2]:]BURSt:INFInite:REARm (on page 94)	Set burst infinite to rearm
[SOURce[1|2]:]BURSt:IDLE (on page 95)	Set burst idle state
[SOURce[1|2]]:COMBine:FEED (on page 100)	Set or query internal noise or external signal
[SOURce[1|2]]:FM[:DEViation] (on page 101)	Set or query frequency deviation
[SOURce[1|2]]:FM:INTernal:FREQuency (on page 102)	Set or query internal modulation frequency
[SOURce[1|2]]:FM:INTernal:FUNCtion (on page 103)	Set or query internal modulation waveform
[SOURce[1|2]]:FM:INTernal:FUNCtion:EFILe (on page 104)	Set or query EFILe setting
[SOURce[1|2]]:FM:SOURce (on page 104)	Set or query frequency modulation source
[SOURce[1|2]]:FM:STATe (on page 105)	Set or query frequency modulation status
[SOURce[1|2]]:FREQuency:CENTer (on page 106)	Set or query center frequency
[SOURce[1|2]]:FREQuency:CONCurrent[:STATe] (on page 107)	Set or query concurrent change of frequency
[SOURce[1|2]]:FREQuency[:CW|:FIXed] (on page 108)	Set or query output waveform frequency
[SOURce[1|2]]:FREQuency:MODE (on page 109)	Set or query sweep status
[SOURce[1|2]]:FREQuency:SPAN (on page 110)	Set or query sweep frequency span
[SOURce[1|2]]:FREQuency:STARt (on page 111)	Set or query sweep start frequency
[SOURce[1|2]]:FREQuency:STOP (on page 112)	Set or query sweep stop frequency
[SOURce[1|2]]:FSKey[:FREQuency] (on page 113)	Set or query FSK hop frequency
[SOURce[1|2]]:FSKey:INTernal:RATE (on page 114)	Set or query FSK internal modulation rate
[SOURce[1|2]]:FSKey:SOURce (on page 115)	Set or query FSK source
[SOURce[1|2]]:FSKey:STATe (on page 116)	Set or query FSK status
[SOURce[1|2]]:FUNCtion:EFILe (on page 117)	Set or query EFILe name

Header	Description
[SOURce[1|2]]:FUNCtion:RAMP:SYMMetry (on page 118)	Set or query ramp waveform symmetry
[SOURce[1|2]]:FUNCtion[:SHAPe] (on page 119)	Set or query output waveform
[SOURce[1|2]]:PHASe[:ADJust] (on page 120)	Set or query output waveform phase
[SOURce[1|2]:PHASe:CONCurrent[:STATe] (on page 121)	Set or query instrument values
[SOURce[1|2]]:PHASe:INITiate (on page 121)	Initiate output waveform phase synchronization
[SOURce[1|2]]:PM[:DEViation] (on page 122)	Set or query phase modulation deviation
[SOURce[1|2]]:PM:INTernal:FREQuency (on page 123)	Set or query internal modulation frequency
(SOURce[1|2]]:PM:INTernal:FUNCtion (on page 124)	Set or query internal modulation waveform
[SOURce[1|2]]:PM:INTernal:FUNCtion:EFILe (on page 125)	Set or query EFILe name
[SOURce[1|2]]:PM:SOURce (on page 125)	Set or query phase modulation source
[SOURce[1|2]]:PM:STATe (on page 126)	Set or query phase modulation status
[SOURce[1|2]]:PULSe:DCYCle (on page 127)	Set or query pulse waveform duty cycle
[SOURce[1|2]]:PULSe:DELay (on page 128)	Set or query pulse waveform lead delay
[SOURce[1|2]]:PULSe:HOLD (on page 129)	Set or query pulse waveform parameter
[SOURce[1|2]]:PULSe:PERiod (on page 130)	Set or query pulse waveform period
[SOURce[1|2]]:PULSe:TRANsition[:LEADing] (on page 131)	Set or query pulse waveform leading edge time
[SOURce[1|2]]:PULSe:TRANsition:TRAiling (on page 132)	Set or query pulse waveform trailing edge time
[SOURce[1|2]]:PULSe:WIDTh (on page 133)	Set or query pulse waveform width
[SOURce[1|2]]:PWM:INTernal:FREQuency (on page 134)	Set or query pulse width modulation frequency
[SOURce[1|2]]:PWM:INTernal:FUNCtion (on page 135)	Set or query pulse width modulation waveform
[SOURce[1|2]]:PWM:INTernal:FUNCtion:EFILe (on page 136)	Set or query EFILe name
[SOURce[1|2]]:PWM:SOURce (on page 136)	Set or query pulse width modulation source
[SOURce[1|2]]:PWM:STATe (on page 137)	Set or query pulse width modulation status
[SOURce[1|2]]:PWM[:DEViation]:DCYCle (on page 138)	Set or query pulse width modulation deviation
[SOURce[1|2]]:RMODe? (on page 137)	Queries the run mode
[SOURce]:ROSCillator:SOURce (on page 139)	Sets the reference clock to either internal or external
[SOURce[1|2]]:SWEep:HTIMe (on page 140)	Set or query sweep hold time
[SOURce[1|2]]:SWEep:MODE (on page 141)	Set or query sweep mode
[SOURce[1|2]]:SWEep:RTIMe (on page 142)	Set or query sweep return time
[SOURce[1|2]]:SWEep:SPACing (on page 143)	Set or query sweep spacing
[SOURce[1|2]]:SWEep:TIME (on page 144)	Set or query sweep time
[SOURce[1|2]]:VOLTage:CONCurrent[:STATe] (on page 145)	Set or query concurrent change of amplitude level
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:HIGH (on page 146)	Set or query output amplitude high level
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:LOW (on page 147)	Set or query output amplitude low level

Header	Description
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:OFFSet (on page 148)	Set or query output offset voltage
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] (on page 149)	Set or query output amplitude
[SOURce[1|2]]:VOLTage:LIMit:HIGH (on page 150)	Set or query output amplitude upper limit
[SOURce[1|2]]:VOLTage:LIMit:LOW (on page 151)	Set or query output amplitude lower limit
[SOURce[1|2]]:VOLTage:UNIT (on page 152)	Set or query output amplitude units
SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude] (on page 153)	Set or query internal noise level

Status commands

Status commands give you the ability to determine the status of the instrument. The next table lists and describes Status commands.

Header	Description
*CLS (on page 31)	Clear all event registers and queues
*ESE (on page 35)	Set or query standard event status enable register
*ESR? (query only) (on page 36)	Return standard event status register
*PSC (on page 63)	Set or query power-on status clear
*SRE (on page 154)	Set or query service request enable register
*STB? (on page 158)	Read status byte
STATus:OPERation:CONDition? (on page 155)	Return operation condition register
STATus:OPERation:ENABle (on page 155)	Set or query operation enable register
STATus:OPERation[:EVENt]? (on page 156)	Return operation event register
STATus:PRESet (on page 156)	Preset SCPI enable register
STATus:QUEStionable:CONDition? (on page 157)	Return questionable condition register
STATus:QUEStionable:ENABle (on page 157)	Set or query questionable enable register
STATus:QUEStionable[:EVENt]? (on page 158)	Return questionable event register

System commands

System commands let you control miscellaneous instrument functions. The next table lists and describes System commands.

Header	Description
`*IDN?` (on page 37)	Return identification information
`*OPT?` (on page 58)	Return option information
`*RST` (on page 65)	Reset
`SYSTem:BEEPer[:IMMediate]` (on page 160)	Generate an audible tone
`SYSTem:BEEPer:STATe` (on page 159)	Set or query beeper state
`SYSTem:ERRor[:NEXT]?` (on page 159)	Return error event queue
`SYSTem:KCLick[:STATe]` (on page 160)	Set or query click sound
`SYSTem:KLOCk[:STATe]` (on page 161)	Set or query front panel lock/unlock
`SYSTem:MACADDress?` (on page 161)	Return the MAC address of an AFG 31000
`SYSTem:PASSword:CDISable (no query form)` (on page 162)	Disable protected commands
`SYSTem:PASSword[:CENable]` (on page 164)	Enable protected commands to function
`SYSTem:PASSword[:CENable]:STATe?` (on page 165)	Return security protection state
`SYSTem:PASSword:NEW` (on page 166)	Change current password
`SYSTem:RESTart` (on page 166)	Restart the unit
`SYSTem:SECurity:IMMediate` (on page 167)	Reset to factory default
`SYSTem:ULANguage` (on page 167)	Set or query language for display screen
`SYSTem:VERSion?` (on page 168)	Return version information

Synchronization commands

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

Header	Description
`*OPC` (on page 58)	Set or query operation complete
`*WAI` (on page 172)	Wait to continue

Trace commands

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

Header	Description
`TRACe|DATA[:DATA]` (on page 168)	Set or query waveform data to edit memory

Trigger commands

Trigger commands let you control all aspects of AFG triggering. The next table lists and describes Trigger commands.

Header	Description
ABORt (on page 27)	Initialize trigger system
*TRG (on page 169)	Force trigger event
TRIGger[:SEQuence][:IMMediate] (on page 171)	Generate a trigger event
TRIGger[:SEQuence]:SLOPe (on page 169)	Set/query the slope of trigger signal
TRIGger[:SEQuence]:SOURce (on page 170)	Set/query the source of trigger signal
TRIGger[:SEQuence]:TIMer (on page 170)	Set/query the period of internal clock

AFG control command

AFG Control command copies setups between two channels. The next table lists and describes the AFG control command.

Header	Description
AFGControl:CSCopy (on page 28)	Copy CH1 (or CH2) setup parameters to CH2 (or CH1)

Screen copy command

Screen copy command copies the screen image. The next table lists and describes the Screen copy command.

Header	Description
HCOPy:SDUMp[:IMMediate] (on page 36)	Copy screen image and save the file to USB memory.

License commands

License commands let you install optional features. The next table lists and describes License commands.

Header	Description
`LICense:HID?` (on page 39)	Returns the instrument HostID unique identifier.
`LICense:INSTall` (on page 40)	Accepts a <block data> license and installs it on the instrument.
`LICense:LIST?` (on page 41)	Returns the active license nomenclatures as a comma-separated list of strings.
`LICense:ERRor?` (on page 41)	This query-only command prompts the instrument to return all events and their messages (delimited by commas) relative to license.
`LICense:GMT?` (on page 42)	This query-only command returns the local system date and time.
`LICense:COUNt?` (on page 43)	This query-only command returns the number of active licenses.
`LICense:ITEM?` (on page 44)	This query-only command is used to query license details about an item in the strongbox.
`LICense:VALidate?` (on page 45)	This query-only command is used to validate whether the instrument is capable of realizing the features of the license nomenclature and that the feature is active.
`LICense:INSTall` (on page 40)	Used to install an instrument license.
`LICense:UNINstall?` (on page 40)	Used to uninstall an instrument license.
`LICense:IDN?` (on page 42)	Used to identify the active host.

Sequence commands

Sequence commands let you to define and edit a waveform sequence.

Header	Description
SEQuence:ELEM[n]:GOTO:INDex (on page 75)	Sets or retrieves the target index for the GOTO command of the sequencer
SEQuence:ELEM[n]:GOTO:STATe (on page 76)	Sets or retrieves the GOTO state of the sequencer
SEQuence:ELEM[n]:MARKer:STATe (on page 77)	Sets or retrieves the MARKer state of the sequencer
SEQuence:ELEM[n]:JTARget:INDex (on page 78)	Sets or retrieves the target index for the sequencer’s event jump operation
SEQuence:ELEM[n]:JTARget:TYPE (on page 79)	Sets or queries the target type for the jump
SEQuence:ELEM[n]:JUMP:EVENt (on page 80)	Set or returns the JUMP trigger event type
SEQuence:ELEM[n]:JUMP:SLOpe (on page 81)	Sets or returns the slope of external trigger for jump
SEQuence:ELEM[n]:LOOP:COUNt (on page 82)	Sets or queries the loop count
SEQuence:ELEM[n]:LOOP:INFinite (on page 83)	Sets or returns the infinite looping state for a sequence element
SEQuence:ELEM[n]:TWAit:EVENt (on page 85)	Set or returns the wait trigger event type
SEQuence:ELEM[n]:TWAit:SLOpe (on page 86)	Sets or returns the slope of external trigger for wait
SEQuence:ELEM[n]:WAVeform[m] (on page 87)	Sets or returns the waveform for a sequence element
SEQuence:LENGth (on page 88)	Sets or returns the sequence length
SEQuence:NEW (on page 88)	Creates a new sequence.

Waveform list commands

Waveform List commands let you to add or remove waveform from waveform list in advance mode.

Header	Description
WLISt:NAME? (on page 172)	Returns the waveform name of an element in the waveform list.
WLISt:SIZE? (on page 173)	Returns the size of the waveform list.
WLISt:WAVeform:DELete (on page 174)	Deletes the waveform from the currently loaded setup.
WLISt:WAVeform:IMPort (on page 175)	Imports the waveform to the currently loaded setup.
WLISt:WAVeform:LENGth? (on page 176)	Returns the size of the waveform.

Sequence control group commands

Sequence control commands let you run a sequence according to the mode you want.

Header	Description
SEQControl:RUN[:IMMediate] (on page 68)	Initiates the output of a waveform or sequence.
SEQControl:STOP[:IMMediate] (on page 68)	Stops the output of a waveform or sequence.
SEQControl:RESET (on page 69)	Resets the sequence to its default state.
SEQControl:RMODe (on page 67)	Sets or returns the run mode of the AWG.
SEQControl:STATe (on page 69)	Sets or returns the sequence mode ON/OFF
SEQControl:RSTATe (on page 70)	Returns the sequence run state ON/OFF
SEQControl:SRATe (on page 70)	Sets or returns the sampling rate
SEQControl:DELay (on page 71)	Sets or returns the skew time
SEQControl:TIMer (on page 74)	Sets or returns the jump/wait timer
SEQControl:SOURce[n]:SCALe (on page 72)	Sets or returns the sequence output scale
SEQControl:SOURce[n]:OFFSet (on page 73)	Sets or returns the sequence output offset

InstaView commands

InstaView commands let you to get the actual waveform into the DUT.

Header	Description
INSTaview[1|2]:STATe (on page 39)	Sets or returns the InstaView state
INSTaview[1|2]:MEASUrement:DELay (on page 37)	Sets or returns the signal transmission delay in cable
INSTaview[1|2]:MEASUrement:MINimum (on page 38)	Returns the minimum level
INSTaview[1|2]:MEASUrement:MAXimum (on page 38)	Returns the maximum level

Command descriptions

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 indicates query-only commands.
- Fully spelled out headers, mnemonics, and arguments are provided 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 AFG you are using; differences are noted.

ABORt (no query form)

Disconnect this PI Interface

This command:

Group

AFG Control

Syntax

```
ABORt
```

Examples

```
ABORt
```

AFGControl:CSCopy (no query form)

This command copies setup parameters for one channel to another channel. If your AFG is a single-channel model, this command is not supported.

This command:

Group

AFG Control

Syntax

```
AFGControl:CSCopy {CH1|CH2},{CH1|CH2}
```

Arguments

```
CH1|CH2
```

Examples

```
AFGControl:CSCopy CH1,CH2
```

Copies the CH1 setup parameters into CH2

*CAL? (query only)

This command performs an internal calibration and returns 0 (pass) or a calibration error code.

This command:

NOTE. The self-calibration can take several minutes to complete. Please unplug all connected cables. During this time, the AFG does not execute any commands. Do not power off the instrument during the self-calibration.

Group

Calibration and Diagnostic

Syntax

```
*CAL?
```

Arguments

None

Returns

`<NR1>`

Where:

<NR1>=0 indicates that the internal calibration completed without errors.

<NR1>≠0 indicates that the AFG detected an error.

Examples

```
*CAL?
```

Performs an internal calibration and returns results; for example, it might return 0, which indicates that the calibration completed without any errors.

Related Commands

`CALibration[:ALL]` (on page 30)

CALibration[:ALL]

The `CALibration[:ALL]` command performs an internal calibration.

The `CALibration[:ALL]?` command performs an internal calibration and returns 0 (Pass) or a calibration error code.

NOTE. The self-calibration can take several minutes to complete. Please unplug all connected cables. During this time, the AFG does not execute any commands. Do not power off the instrument during the self-calibration.

This command:

Group

Calibration and Diagnostic

Syntax

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

Returns

`<NR1>`

Where:

`<NR1>=0` indicates that the internal calibration completed without errors.

`<NR1>≠0` indicates that the AFG detected an error.

Examples

`CALibration`

Performs an internal calibration.

`CALibration?`

Performs an internal calibration and returns results; for example, it might return 0, which indicates that the calibration completed without any errors.

Related Commands

`*CAL?` (on page 29)

*CLS (no query form)

This command clears all the event registers and queues used in the AFG status and event reporting system.

This command:

Group

Status

Syntax

```
*CLS
```

Examples

```
*CLS
```

Clears all of the event registers and queues.

DIAGnostic[:ALL]

The `DIAGnostic[:ALL]` command performs a self-test. The `DIAGnostic[:ALL]?` command returns the results after executing the test.

NOTE. The self-test can take several minutes to complete. During this time, the AFG does not execute any commands. Do not power off the instrument during the self-test.

This command:

Group

Calibration and Diagnostic

Syntax

```
DIAGnostic[:ALL]
DIAGnostic[:ALL]?
```

Returns

`<NR1>`

Where:

`<NR1>=0` indicates that the self-test completed without errors.

`<NR1>≠0` indicates that the AFG detected an error.

Examples

```
DIAGnostic
```

Performs a self-test.

```
DIAGnostic?
```

Performs a self-test and returns a number indicating the outcome of the self-test.

Related Commands

`*TST?` (on page 171)

DISPlay:BRIGhtness

This command sets or queries the brightness of the LCD display.

This command:

Group

Display

Syntax

```
DISPlay:BRIGhtness {<brightness>|MINimum|MAXimum}
DISPlay:BRIGhtness? [MINimum|MAXimum]
```

Arguments

`<brightness>::=<NR2>`

Where:

`<NR2>` is a range of display brightness from 0.00 through 1.00 (resolution: 3 digits); the larger the value, the greater the screen brightness.

MINimum sets the display to the lowest brightness level (0.00).

MAXimum sets the display to the highest brightness level (1.00).

Returns

`<NR2>`

Examples

`DISPlay:BRIGhtness MAX`

Sets the display brightness to the highest brightness level.

DISPlay:SAVer:IMMediate (no query form)

This command sets the screen saver state to `ON`, regardless of the `DISPlay:SAVer[:STATe]?` command setting. The screen saver is enabled immediately (without waiting five minutes).

This command:

Group

Display

Syntax

`DISPlay:SAVer:IMMediate`

Examples

`DISPlay:SAVer:IMMediate`

Related Commands

`DISPlay:SAVer[:STATe]` (on page 33)

DISPlay:SAVer[:STATe]

This command sets or queries the screen saver setting of the LCD display. When enabled, the screen saver starts automatically if no operations are applied to the instrument front panel for five minutes.

This command:

Group

Display

Syntax

```
DISPlay:SAVer[:STATe] {ON|OFF|<NR1>}
DISPlay:SAVer[:STATe]?
```

Arguments

`ON or <NR1>≠0`, enables the screen saver function.

`OFF or <NR1>=0`, disables the screen saver function.

Returns

`<NR1>`

Examples

`DISPlay:SAVer:STATe OFF`

Disables the screen saver function.

DISPlay[:WINDow]:TEXT[:DATA]

The `DISPlay[:WINDow]:TEXT[:DATA]` command displays a text message on the instrument screen.

The `DISPlay[:WINDow]:TEXT[:DATA]?` query returns the text string currently displayed on the instrument screen.

The displayable characters are ASCII codes 32 through 126, and the instrument can display approximately 64 characters.

This command:

Group

Display

Syntax

```
DISPlay[:WINDow]:TEXT[:DATA] <string>
DISPlay[:WINDow]:TEXT[:DATA]?
```

Arguments

`<string>`

Returns

`<string>`

Examples

`DISPlay:TEXT?`

Returns the currently displayed text message.

DISPlay[:WINDow]:TEXT:CLEar (no query form)

This command clears the text message from the display screen.

This command:

Group

Display

Syntax

```
DISPlay[:WINDow]:TEXT:CLEar
```

Examples

```
DISPlay:TEXT:CLEar
```

Clears the text message from the screen.

*ESE

This command sets or queries the bits in the Event Status Enable Register (ESER) used in the status and events reporting system of the arbitrary/function generator.

The query command returns the contents of the ESER.

This command:

Group

Status

Syntax

```
*ESE <bit_value>
*ESE?
```

Arguments

<bit_value>::=`<NR1>`

Where:

`<NR1>` is a value in the range of 0 through 255; the binary bits of the ESER are set according to this value.

Returns

```
<bit_value>
```

Examples

```
*ESE 177
```

Sets the ESER to 177 (binary 10110001), which sets the PON, CME, EXE, and OPC bits.

```
*ESE?
```

Might return 186, indicating that the ESER contains the binary value 10111010

Related Commands

```
*CLS (on page 31)
*ESR? (on page 36)
*PSC (on page 63)
*SRE (on page 154)
*STB? (on page 158)
```

*ESR? (query only)

This query-only command returns the contents of the Standard Event Status Register (SESR) used in the status events reporting system in the AFG. `*ESR` also clears the SESR (since reading the SESR clears the register).

This command:

Group

Status

Syntax

```
*ESR?
```

Returns

```
<NR1>
```

Indicates the contents of the SESR as a decimal integer.

Examples

```
*ESR?
```

Might return 181, which indicates that the SESR contains the binary number 10110101.

Related Commands

```
*CLS (on page 31)
*ESE (on page 35)
*SRE (on page 154)
*STB? (on page 158)
```

HCOPy:SDUMp[:IMMediate] (no query form)

This command copies a screen image and saves the image file to the USB flash drive. The image files are saved in a folder named TEK on the USB flash drive. The image file will have a name, for example, something similar to *AFG31K_2018-06-21_10-10-41.png*.

This command:

Group

Screen copy

Syntax

```
HCOPy:SDUMp[:IMMediate]
```

Examples

```
HCOPy:SDUMp:IMMediate
```

Copies the screen image and creates a graphic file on the USB memory.

*IDN? (query only)

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

This command:

Group

System

Syntax

```
*IDN?
```

Returns

<Manufacturer>,<Model>,<Serial Number>,<Firmware Level>

Where:

```
<Manufacturer>::=TEKTRONIX
<Model>::= {AFG31021 | AFG31022 | AFG31051 | AFG31052 | AFG31101 | AFG31102 | AFG31151
| AFG31152 | AFG31251 | AFG31252}
<Serial Number>
<Firmware Level> ::=SCPI:99.0 FV:2.0
```

Examples

```
*IDN?
```

Might return the following response: `TEKTRONIX,AFG31021,C100101,SCPI:99.0 FV:1.0`

INSTaview[1|2]:MEASUrement:DELay

This command sets or returns the signal transmission delay in cable

This command:

Group

InstaView

Syntax

```
INSTaview[1|2]:MEASUrement:DELay {<delay>}
INSTaview[1|2]:MEASUrement:DELay?
```

Arguments

```
<delay>::=<NRf>[<units>]
```

Where:

```
<units>::=[s | ms | µs | ns]
```

Returns

```
Delay
```

Examples

```
INSTaview1:MEASUrement:DELay 2e-9
```

Sets the CH1 cable delay time to 2 ns.

INSTaview[1|2]:MEASUrement:MAXimum (query only)

This query-only command returns the maximum level of output signal in DUT.

This command:

Group

InstaView

Syntax

```
INSTaview[1|2]:MEASUrement:MAXimum?
```

Returns

```
<level>
```

Examples

```
INSTaview1:MEASUrement:MAXimum?
```

Get the Ch1 InstaView waveform max level.

INSTaview[1|2]:MEASUrement:MINimum (query only)

This query-only command returns the minimum level of output signal in DUT.

This command:

Group

InstaView

Syntax

```
INSTaview[1|2]:MEASUrement:MINimum?
```

Returns

```
<level>
```

Examples

```
INSTaview1:MEASUrement:MINimum?
```

Get the CH1 InstaView waveform min level.

INSTaview[1|2]:STATe

This command sets or returns the InstaView state.

This command:

Group

InstaView

Syntax

```
INSTaview[1|2]:STATe {ON|OFF|<NR1>}
INSTaview[1|2]:STATe?
```

Arguments

`ON or <NR1>≠0`

Enable InstaView function.

`OFF or <NR1>=0`

Disable InstaView function.

Returns

`<NR1>`

Examples

`INSTaview:STATe ON`

Enable CH1 InstaView function.

LICense:HID? (query only)

This command returns the instrument HostID unique identifier.

This command:

Group

License

Syntax

```
LICense:HID?
```

Returns

`<string>`

The instrument HostID unique identifier.

Examples

`LICense:HID?`

Might return `AFG-9CS4US5SGJN6X`

LICense: INSTall (no query form)

This command accepts a `<block_data>` license and installs it on the instrument. Restarting the instrument may be necessary to activate the capabilities of a licensed instrument.

This command:

Group

License

Syntax

```
LICense:INSTall <block_data>
```

Arguments

`<arbitrary block>`

`<block_data>` is the license in block data format.

Examples

```
LICense:INSTall <block_data>
```

Install license file to unit.

LICense: UNINstall?

This command is used to return or uninstall a specified license. The exit keyfile generated by the uninstall is returned as block data.

This command attempts to first match the specified parameter to a transaction ID and then to nomenclature. If duplicate active nomenclatures exist, the active license with the earliest expiration is removed. If a transaction ID is specified for a previously-uninstalled license, the same exit keyfile is returned as when the license was first uninstalled. A transaction ID is required to return a previously-uninstalled license.

If a command is called with nomenclature and there are no active matching licenses, an error is generated.

Group

License

Syntax

```
LICense:UNINst? <transaction ID or nomenclature>
```

Returns

`<block_data>`

The exit keyfile.

Examples

```
LICense:UNINstall?
```

Returns `<block data>`.

Related Commands

`LiCense:ERRor? (on page 41), *ESR?`

LICense:LIST? (query only)

This query-only command returns the active license feature names as a comma-separated list of strings.

This command:

Group

License

Syntax

```
LICense:LIST?
```

Returns

```
<string>
```

.

Examples

```
LICense:LIST?
```

Might return `AFG3MEM, AFG3SEQ,` to indicate that it supports the large memory and sequence feature.

LICense:ERRor? (query only)

This query-only command prompts the instrument to return all events and their messages (delimited by commas). Also see License command errors.

This command:

Group

License

Syntax

```
LICense:ERRor?
```

Returns

```
<string>
```

All events and their messages (delimited by commas).

Examples

```
LICense:ERRor?
```

-3000, license error, 125, signature verification failed.

LICense: GMT? (query only)

This query-only command prompts the instrument to return the local system date and time as known by the licensing system on the instrument.

This command:

Group

License

Syntax

`LICense:GMT?`

Returns

`<string>`

ISO 8601-compliant format, including time zone offset, in the format `YYYY-MM-DDTHH24:MM:SS+HH:MM`.

ISO 8601 uses negative offsets for items west of Greenwich.

Examples

`LICense:GMT?`

Might return `2016-05-23T18:46:34-08:00`.

LICense: IDN? (query only)

This query-only command returns the manufacturer, model, serial number, and firmware revision of the active host.

This command:

Group

License

Syntax

`LICense:IDN?`

Returns

`<string>`

Returns the manufacturer, model, serial number, and firmware revision in a comma-separated list.

Examples

`LICense:IDN?`

Might return `TEKTRONIX, ABC1234, B010100, FV:1.2.3`

LICense: COUNT? (query only)

This query-only command returns the count of active AFG31000 Series licenses.

This command:

Group

License

Syntax

```
LICense:COUNT?
```

Returns

`<string>`

Indicates the number of active instrument licenses.

Examples

```
LICense:COUNT?
```

Might return `5`.

LICense: ITEM? (query only)

This query-only command prompts the instrument to return license details about a specific item in the strongbox of the instrument.

This command:

Group

License

Syntax

```
LICense:ITEM? <item number>
```

Arguments

`<item number>`

Specifies the order number of the item in the strongbox. The order is not constant due to changes in the strongbox such as the installation, uninstallation, or expiration of licenses.

Returns

`<nomenclature>, <type>, <expiration date>, <transaction ID>, <APPID>, <description>`

Where:

`<nomenclature>` is the license nomenclature.

`<type>` is the license type. Types include `Trial`, `Fixed`, `Floating`, `MFGTEST`, `SVCTEST`, and `ENGINEERING`.

`<expiration date>` is the date on which the license validity ends, given in ISO 8601-compliant format, in the format `YYYY-MM-DDTHH24:MM:SS+HH:MM`.

`<transaction ID>` is the transaction identifier for the item.

`<APPID>` is a comma-separated list of installed application identifiers.

`<description>` contains detailed license information.

Examples

```
LICense:ITEM? 3
```

Might return: `LICENSENOMENCLATURE, Fixed, 2016-05-23T18:46:34-08:00, ABCD9876, "EFG, XYZ", "Full instrument description"`

LICense: VALidate? (query only)

This query-only command is used to validate whether the instrument state is capable of realizing features of the license nomenclature and that the feature is active.

Validation requests that return `False` can be queried with the `LICense:ERRor?` (on page 41) command.

This command:

Group

License

Syntax

```
LICense:VALidate ?
```

Returns

```
<string>
```

The validity response, either `True` or `False`. A `False` response can be due to missing software, dependent licenses, or hardware problems.

Examples

```
LICense:VALidate?
```

Might return `True`.

MEMory:STATe:DELete (no query form)

This command deletes the contents of specified setup memory. If a specified setup memory is not allowed to be overwritten or deleted, this command returns an error.

This command:

Group

Memory

Syntax

```
MEMory:STATe:DELete {0|1|2|3|4}
```

Arguments

```
0, 1, 2, 3, or 4
```

Specifies the location of setup memory.

Examples

```
MEMory:STATe:DELete 1
```

Deletes the contents of specified setup 1 memory.

MEMory:STATe:LOCK

This command sets or queries whether to lock the specified setup memory. If you lock a setup memory, you cannot overwrite or delete the setup file.

You cannot execute this command for the setup memory of location number 0 (last setup memory).

This command:

Group

Memory

Syntax

```
MEMory:STATe:LOCK {1|2|3|4},{ON|OFF|<NR1>}
MEMory:STATe:LOCK? {1|2|3|4}
```

Arguments

`0, 1, 2, 3, or 4` specifies the setup memory to lock or query.

`ON or <NR1>≠0`

Locks the specified location of setup memory.

`OFF or <NR1>=0`

Allows you to overwrite or delete the specified location of setup memory.

Returns

`<NR1>`

Examples

`MEMory:STATe:LOCK 1,ON`

Locks the setup memory of location number 1

MEMory:STATe:RECall:AUTo

This command sets or queries whether to enable the automatic recall of last setup memory when powered-on. The next time you apply the power, the AFG will automatically recall the settings you used when you powered off the instrument.

If you select OFF, the default setups are recalled when you power on the instrument.

This command:

Group

Memory

Syntax

```
MEMory:STATe:RECall:AUTo {ON|OFF|<NR1>}
MEMory:STATe:RECall:AUTo?
```

Arguments

`OFF or <NR1>=0`

Disables the last setup recall function.

`ON or <NR1>≠0`

Enables the recall of the setup memory last used before the instrument was powered off.

Returns

`<NR1>`

Examples

`MEMory:STATe:RECall:AUTo ON`

Sets the instrument to recall the last setup memory when powered-on.

MEMory:STATe:VALid? (query only)

This query-only command returns the availability of a setup memory.

This command:

Group

Memory

Syntax

```
MEMory:STATe:VALid? {0|1|2|3|4}
```

Arguments

`0, 1, 2, 3, or 4`

Specifies the location of setup memory.

Returns

`<NR1>`

1 means that the specified setup memory has been saved.

0 means that the specified setup memory has been deleted.

Examples

```
MEMory:STATe:VALid? 0
```

Might return 1 if the specified setup memory has been saved.

MMEMory:CATalog? (query only)

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

This command:

Group

Mass Memory

Syntax

```
MMEMory:CATalog?
```

Returns

```
<NR1>,<NR1>   [,<file_name>,<file_type>,<file_size>]
```

Where:

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 exact name of a file.

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

`<file_size>` is the size of the file, in bytes.

Examples

```
MMEMory:CATalog?
```

Might return the following response:

```
32751616,27970560,”SAMPLE1.TFS,,5412”
```

MMEMory:CDIRectory

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

This command:

Group

Mass Memory

Syntax

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

Arguments

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

Indicates the current working directory for the mass storage system.

If you do not specify a parameter, the directory is not changed.

Returns

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

Examples

```
MMEMory:CDIRectory  "U:/AFG/WORK0"
```

Changes the current directory to `<USB memory>/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 be overwritten or deleted, this command causes an error. You can delete a directory if it is empty.

This command:

Group

Mass Memory

Syntax

```
MMEMory:DELete <file_name>
```

Arguments

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

Specifies a file to be deleted.

Examples

```
MMEMory:DELete "M:/TEK001.tfwx"
```

Deletes the specified file from the internal mass storage.

MMEMory:LOAD:STATe (no query form)

This command copies a setup file in the mass storage system to an internal setup memory. If a specified internal setup memory is locked, this command causes an error. When you power off the instrument, the setups are automatically overwritten in the setup memory 0 (last setup memory).

This command:

Group

Mass Memory

Syntax

```
MMEMory:LOAD:STATe {0|1|2|3|4},<file_name>
```

Arguments

`0, 1, 2, 3, or 4`

Specifies the location of setup memory.

`<file_name>::=<string>`

Specifies a setup file to be copied.

Examples

```
MMEMory:LOAD:STATe 1,"U:/SETUP1.tfs"
```

Copies a file named SETUP1.TFS in the USB mass storage into the internal memory location 1.

Related Commands

`MEMory:STATe:LOCK (on page 46)`
`MEMory:STATe:RECall:AUTo (on page 47)`
`MMEMory:STORe:STATe (on page 54)`

MMEMory:LOAD:TRACe (no query form)

This command copies a waveform data file in the mass storage system to Edit Memory. If the file format is different, this command causes an error.

This command:

Group

Mass Memory

Syntax

```
MMEMory:LOAD:TRACe EMEMory[1]|EMEMory2,<file_name>
```

Arguments

`EMEMory[1]|EMEMory2`

Refers to the Commands Arguments column in the table in Appendix B.

`<file_name>::=<string>`

Specifies a waveform data file to be copied.

Examples

```
MMEMory:LOAD:TRACe EMEMory1,"U:/TEK001.tfwx"
```

Copies a file named `TEK001.tfwx` in the USB mass storage into Edit Memory 1.

Related Commands

`MMEMory:STORe:TRACe (on page 55)`

MMEMory:LOCK[:STATe]

This command sets or queries whether to lock a file or directory in the mass storage system. If you lock a file or directory, you cannot overwrite or delete it.

This command:

Group

Mass Memory

Syntax

```
MMEMory:LOCK[:STATe] <file_name>,{ON|OFF|<NR1>}
MMEMory:LOCK[:STATe]? <file_name>
```

Arguments

`ON or <NR1>≠0`

Locks a file or directory in the mass storage system.

`OFF or <NR1>=0`

Allows you to overwrite or delete a file or directory in the mass storage system.

Returns

`<NR1>`

Examples

```
MMEMory:LOCK:STATe "M:/setup1.tfs", ON
```

Locks the file "`setup1.tfs`"

MMEMory:MDIRectory (no query form)

This command creates a directory in the mass storage system. If the specified directory is locked in the mass storage system, this command causes an error.

This command:

Group

Mass Memory

Syntax

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

Arguments

`<directory_name>::=<string>`

Specifies a directory name to be created.

Examples

```
MMEMory:MDIRectory "M:/sample"
```

Creates a directory named “SAMPLE1” in the mass storage system.

MMEMory:STORe:STATe (no query form)

This command copies a setup file in the setup memory to a specified file in the mass storage system. If the specified file in the mass storage system is locked, this command causes an error. You cannot create a new file if the directory is locked. If the setup memory is deleted, this command causes an error. The `<file_name>` argument is a quoted string that defines the file name and path.

This command:

Group

Mass Memory

Syntax

```
MMEMory:STORe:STATe {0|1|2|3|4},<file_name>
```

Arguments

`0, 1, 2, 3, or 4`

Specifies the location of setup memory.

`<file_name>::=<string>`

Specifies a file name in the mass storage system; the <file_name> includes path; path separators are forward slashes (/).

Examples

```
MMEMory:STORe:STATe 1,"M:/test1.tfs"
```

Copies the setup file in the setup memory location 1 to a file named “`test1.tfs`” in the internal mass storage system.

Related Commands

`MMEMory:LOAD:STATe (on page 52)`
`MMEMory:LOCK[:STATe] (on page 53)`

MMEMory:STORe:TRACe (no query form)

This command copies a waveform data file in the Edit Memory to a file in the mass storage system. If the file in the mass storage is locked, this command causes an error. You cannot create a new file if the directory is locked.

This command:

Group

Mass Memory

Syntax

```
MMEMory:STORe:TRACe EMEMory[1]|EMEMory2,<file_name>
```

Arguments

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

Specifies a file name in the mass storage system; the <file_name> includes path; path separators are forward slashes (/).

Examples

```
MMEMory:STORe:TRACe EMEMory1,"M:/sample1.tfwx"
```

Copies the content of EMEMory1 to a file named "sample1.tfwx" in the internal mass storage system.

Related Commands

`MMEMory:LOCK[:STATe]` (on page 53)
`MMEMory:LOAD:TRACe` (on page 52)

MMEMory:OPEN:SEQuence (no query form)

This command loads a single sequence if <desired_sequence>is designated.

File formats supported: `.seq – AFG30K Series sequence`.

NOTE. If the sequence, or any associated waveform name already exists, it will be overwritten without warning.

This command:

Group

Mass Memory

Syntax

```
MMEMory:OPEN:SEQuence  <file_name>
```

Arguments

```
< file_path >::=<string>
```

Indicates the relative path to the driver.

Examples

```
MMEMory:OPEN:SEQuence "U:/AFGseq.seq"
```

Opens the AFGseq.seq file and loads the waveforms,sequences, and parameters from this file.

Related Commands

MMEMory:SAVE:SEQuence (on page 57)

MMEMory:SAVE:SEQuence (no query form)

This command exports a sequence given a unique name to an eligible storage location as the .seq file type.his command loads a single sequence if <desired_sequence>is designated.

File formats supported: `.seq - AFG30K Series sequence.`

NOTE. If a file already exists in the selected file path, it is overwritten without warning. If the save fails, the file is deleted.

This command:

Group

Mass Memory

Syntax

```
MMEMory:SAVE:SEQuence  <file_name>
```

Arguments

```
< file_path >::=<string>
```

Indicates the relative path to the driver.

Examples

```
MMEMory:SAVE:SEQuence "U:/mySequence.seq"
```

Saves the sequence to the USB memory `mySequence.seq`.

Related Commands

`MMEMory:OPEN:SEQuence (on page 56)`

*OPC

This command generates the operation complete message by setting bit `0` in the Standard Event Status Register (SESR) when all pending commands that generate an OPC message are complete. The query command places the ASCII character `1` into the output queue when all such OPC commands are complete.

This command:

Group

Synchronization

Syntax

```
*OPC
*OPC?
```

Arguments

None

Returns

```
<execution complete>::=1
```

Where 1 indicates that all pending operations are complete.

Examples

```
*OPC?
```

Return 1 to indicate that all pending OPC operations are finished.

*OPT? (query only)

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

This command:

Group

System

Syntax

```
*OPT?
```

Arguments

None

Returns

```
<OPT>[, <OPT>[, <OPT>[, <OPT>]]]
```

Examples

```
*OPT?
```

Might return empty string, which indicates no option is installed in the instrument.

OUTPut[1|2]:IMPedance

The `OUTPut:IMPedance` command sets the output load impedance for the specified channel. The specified value is used for amplitude, offset, and high/low level settings. You can set the impedance to any value from 1 Ω to 10 kΩ with a resolution of 1 Ω or 3 digits. The default value is 50 Ω. The `OUTPut:IMPedance?` command returns the current load impedance setting in ohms. If the load impedance is set to INFinity, the query command returns 9.9E+37.

This command:

Group

Output

Syntax

```
OUTPut[1|2]:IMPedance {<ohms>|INFinity|MINimum|MAXimum}
OUTPut[1|2]:IMPedance?
```

Arguments

```
<ohms>::=<NR3>[<units>]
```

where: <units>::=OHM

INFinity sets the load impedance to >10 kΩ.

MINimum sets the load impedance to 1 Ω.

MAXimum sets the load impedance to 10 kΩ.

Returns

```
<ohms>::=<NR3>
```

Examples

```
OUTPut1:IMPedance MAXimum
```

Sets the CH 1 load impedance to 10 kΩ.

OUTPut[1|2]:POLarity

This command inverts a specified output waveform relative to the offset level. The query command returns the polarity for the specified channel.

This command:

Group

Output

Syntax

```
OUTPut[1|2]:POLarity {NORMal|INVerted}
OUTPut[1|2]:POLarity?
```

Arguments

`NORMal`

Sets the specified output waveform polarity to Normal.

`INVerted`

Sets the specified output waveform polarity to Inverted.

Returns

`NORM|INV`

Examples

`OUTPut1:POLarity NORMal`

Sets the CH 1 waveform polarity to Normal.

OUTPut[1|2][:STATe]

This command sets or queries whether to enable the AFG output for the specified channel.

This command:

Group

Output

Syntax

```
OUTPut[1|2][:STATe]{ON|OFF|<NR1>}
OUTPut[1|2][:STATe]?
```

Arguments

`ON or <NR1>≠0`

Enables the AFG output.

`OFF or <NR1>=0`

Disables the AFG output.

Returns

`<NR1>`

Examples

`OUTPut1:STATe ON`

Sets the AFG CH 1 output to ON.

OUTPut:TRIGger:MODE

This command sets or queries the mode (trigger or sync) for Trigger Output signal. When the burst count is set to Inf-Cycles in burst mode, TRIGger indicates that the infinite number of cycles of waveform will be output from the Trigger Output connector. When the burst count is set to Inf-Cycles in burst mode, SYNC indicates that one pulse waveform is output from the Trigger Output connector when the Inf-Cycles starts. When Run Mode is specified other than Burst Inf-Cycles, TRIGger, and SYNC have the same effect.

This command:

Group

Output

Syntax

```
OUTPut:TRIGger:MODE {TRIGger|SYNC}
OUTPut:TRIGger:MODE?
```

Arguments

`TRIGger`

Means TRIGger is selected for Trigger Out.

`SYNC`

Means SYNC is selected for Trigger Out.

Returns

`TRIG|SYNC`

Examples

`OUTPut:TRIGger:MODE SYNC`

Outputs one cycle waveform from the Trigger Output connector when Inf-Cycles starts.

*PSC

This command sets and queries the power-on status flag that controls the automatic power-on execution of `SRER` and `ESER`. When `*PSC` is true, `SRER` and `ESER` are set to 0 at power-on. When *PSC is false, the current values in the `SRER` and `ESER` are preserved in nonvolatile memory when power is shut off and are restored at power-on.

This command:

Group

Status

Syntax

```
*PSC <NR1>
*PSC?
```

Arguments

`<NR1>=0`

Sets the power-on status clear flag to false, disables the power-on clear, and allows the instrument to possibly assert `SRQ` after power-on.

`<NR1>≠0`

Sets the power-on status clear flag true; sending *PSC 1 therefore enables the power-on status clear and prevents any `SRQ` assertion after power-on.

Returns

`<NR1>`

Examples

`*PSC 0`

Sets the power-on status clear flag to false.

*RCL (no query form)

This command restores the state of the instrument from a copy of the settings stored in the setup memory. The settings are stored using the `*SAV` command. If the specified setup memory is deleted, this command causes an error.

This command:

Group

Memory

Syntax

```
*RCL {0|1|2|3|4}
```

Arguments

`0, 1, 2, 3, or 4`

Specifies the location of setup memory.

Examples

```
*RCL 3
```

Restores the instrument from a copy of the settings stored in memory location 3.

Related Commands

`*SAV (on page 65)`

RECALL:SETUp (no query form)

This command restores the state of the instrument from a copy of the settings stored in the mess memory. The settings are stored using the `*SAV` command. If the specified file is deleted, this command causes an error.

This command:

Group

Mass Memory

Syntax

```
RECALL:SETUp <file_name>
```

Arguments

`<file_name>`

Specifies the file name.

Examples

```
RECALL:SETUp "U:/file1.tfs"
```

Restores the instrument from the file named file1.tfs that is stored in mass memory.

Related Commands

`SAVe:SETUp (on page 66)`

*RST (no query form)

This command resets the instrument to the factory default settings. This command is equivalent to pushing the Default button on the front panel. The default values are listed in Default Settings.

This command:

Group

System

Syntax

```
*RST
```

Arguments

None

Examples

```
*RST
```

Resets the AFG settings to the factory defaults.

*SAV (no query form)

This command stores the current settings of the AFG to a specified setup memory location. A setup memory location numbered `0` (last setup memory) is automatically overwritten by the setups when you power off the instrument.

NOTE: If a specified numbered setup memory is locked, this command causes an error.

This command:

Group

Memory

Syntax

```
*SAV {0|1|2|3|4}
```

Arguments

`0, 1, 2, 3, or 4`

Examples

```
*SAV 2
```

Saves the current instrument state in the memory location 2.

Related Commands

`*RCL (on page 64)`

SAVe:SETUp (no query form)

This command stores the current settings of the arbitrary function generator to a specified file in mass memory.

This command:

Group

Mass Memory

Syntax

```
SAVe:SETUp <file_name>
```

Arguments

```
<file_name>
```

Examples

```
SAVe:SETUp "U:/file1.tfs"
```

Saves the present instrument state to the file1.tfs in the USB memory.

Related Commands

`RECALL:SETUp (on page 64)`

SEQControl:RMODe

This command and query sets or returns the run mode of the advanced mode.

This command:

Group

Control

Syntax

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

Arguments

`CONTinuous`

Sets Run Mode to Continuous.

`TRIGgered`

Sets Run Mode to Triggered.

`GATed`

Sets Run Mode to Gated.

`SEQuence`

Sets Run Mode to Sequence.

Returns

`CONT|TRIG|GAT|SEQ`

Examples

`SEQControl:RMODe TRIGgered`

Sets the instrument Run mode to Triggered.

`SEQControl:RMODe?`

Returns CONT if the instrument is in continuous mode.

Related Commands

`SEQControl:RUN[:IMMediate]` (on page 68)
`SEQControl:STOP[:IMMediate]` (on page 68)

SEQControl:RUN[:IMMediate] (no query form)

This command initiates the output of a waveform or sequence. This is equivalent to pushing the run button on the front-panel. The validation of the sequence will be done first. If there is an error, such as different length waveforms on CH1&CH2 one element, an error will be returned and the run will fail.

This command:

Group

Control

Syntax

```
SEQControl:RUN[:IMMediate]
```

Examples

```
SEQControl:RUN:IMMediate
```

Put the AFG sequence in the run state if valid.

Related Commands

`SEQControl:STOP[:IMMediate]` (on page 68)

SEQControl:STOP[:IMMediate] (no query form)

This command stops the output of a sequence.

This command:

Group

Control

Syntax

```
SEQControl:STOP[:IMMediate]
```

Examples

```
SEQControl:STOP:IMMediate
```

Stop the output of a sequence.

Related Commands

`SEQControl:RUN[:IMMediate]` (on page 68)

SEQControl:RESET[:IMMediate] (no query form)

This command resets sequence to its default state. The waveform list and the sequence table will be cleared. Parameters such as sampling rate, skew time will be set to its default value.

This command:

Group

Control

Syntax

```
SEQControl:RESET
```

Examples

```
SEQControl:RESET
```

Put the AFG sequence in the run state if valid.

Related Commands

`SEQControl:RUN[:IMMediate]` (on page 68)
`SEQControl:STOP[:IMMediate]` (on page 68)

SEQControl:STATe

This command takes the instrument to Sequence mode or returns it to AFG mode.

This command:

Group

Control

Syntax

```
SEQControl:STATe {ON|OFF|<NR1>}
SEQControl:STATe?
```

Arguments

`ON or <NR1>≠ 0`

Sequence mode on

`OFF or <NR1> = 0`

Sequence mode off

Returns

`<NR1>`

0 indicates that the instrument is in AFG mode.

1 indicates that the instrument is in Sequence mode.

Examples

```
SEQControl:STATe ON
```

Enters sequence mode

```
SEQControl:STATe?
```

Returns 0 if the instrument sequence is stopped.

SEQControl:RSTATe? (query only)

This query returns the run state of the sequence.

This command:

Group

Control

Syntax

```
SEQControl:RSTATe?
```

Returns

`<NR1>`

`0` indicates that the instrument has stopped.

`1` indicates that the instrument is running.

Examples

```
SEQControl:RSTATe?
```

Return 0 if the instrument sequence is stopped.

Related Commands

```
SEQControl:RUN[:IMMediate] (on page 68)
SEQControl:STOP[:IMMediate] (on page 68)
```

SEQControl:SRATe

This command sets and query the sampling rate of the instrument.

This command:

Group

Control

Syntax

```
SEQControl:SRATe <NR2>
SEQControl:SRATe?
```

Arguments

`<NR2>`

Returns

`<NR2>`

Examples

```
SEQControl:SRATe  1.5E9
```

Sets sampling rate to 1.5 GHz.

SEQControl:DELay

This command set or returns the time of trigger delay in sequence. This command is only valid in two-channel model instrument.

This command:

Group

Control

Syntax

```
SEQControl:DELay {<skew_time>|MINimum|MAXimum}
SEQControl:DELay?
```

Arguments

`<skew_time>::=<NR3>`

Ranges from -320ns to 320ns

At `*RST`,this returns 0.

Returns

`<NR3>`

Examples

`SEQControl:DELay 100NS`

Sets the channel two skew timer to 100ns.

Related Commands

`[SOURce[1|2]:]PHASe:INITiate (on page 121)`

SEQControl:SOURce[n]:SCALe

This command sets or returns scale of sequence output.

This command:

Group

Control

Syntax

```
SEQControl:SOURce[1|2]:SCALe <scale_value>
SEQControl:SOURce[1|2]:SCALe?
```

Arguments

```
<scale_value>::=<NR3>
```

Ranges from 0.0 to 1000.

At `*RST`,this returns 100.

Returns

```
<NR3>
```

Examples

```
SEQControl:SOURce1:SCALe 50
```

Sets the channel one output scale to 50%.

Related Commands

`SEQControl:SOURce[1|2]:OFFSet` (on page 73)

SEQControl:SOURce[n]:OFFSet

This command sets or returns offset of sequence output.

This command:

Group

Control

Syntax

```
SEQControl:SOURce[1|2]:OFFSet {<offset_value>|MINimum|MAXimum}
SEQControl:SOURce[1|2]:OFFSet?
```

Arguments

`<offset_value>::=<NR3>`

Ranges from -1e6 to 1e6.

At `*RST`,this returns 0.

Returns

`<NR3>`

Examples

`SEQControl:SOURce1:OFFSet 0.1`

Sets the channel one output offset to 100 mV.

Related Commands

`SEQControl:SOURce[1|2]:SCALe (on page 73)`

SEQControl:TIMer

This command sets or returns the time of wait and jump trigger event. When the event is TIMER and not in off state, it takes effect.

This command:

Group

Control

Syntax

```
SEQControl:TIMer {<timer_value>|MINimum|MAXimum}
SEQControl:TIMer?
```

Arguments

```
< timer_value >::=<NR2>
```

Ranges from 2 us to 3600s.

Returns

```
<NR2>
```

Examples

```
SEQControl:TIMer 0.5
```

Sets the trigger timer to 0.5s.

Related Commands

`SEQuence:ELEM[n]:JUMP:EVENt (on page 80)`

SEQuence:ELEM[n]:GOTO:INDex

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

NOTE: The first element of a sequence is taken to be 1 not 0.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:GOTO:INDex {<target>|MINimum|MAXimum}
SEQuence:ELEM[n]:GOTO:INDex?
```

Arguments

`<target>::=<NR1>`

Ranges from 1 to 256, it is related to the current sequence length and run mode. It should be no bigger than current sequence table size. The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`<target>`

Examples

`SEQuence:ELEM1:GOTO:INDEX 6`

Causes the sequencer to jump to sixth element after executing the first element.

`SEQuence:ELEM1:GOTO:INDEX?`

Might return 6.

Related Commands

SEQuence:ELEM[n]:GOTO:STATe (on page 76)
SEQuence:LENGth (on page 88)

SEQuence:ELEM[n]:GOTO:STATe

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

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:GOTO:STATe <goto_state>
SEQuence:ELEM[n]:GOTO:STATe?
```

Arguments

`<goto_state>::=<Boolean>`

`0` indicates OFF.

`1` indicates ON.

At `*RST`, this returns 0.

The value of `<n>` is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

`SEQuence:ELEM1:GOTO:STATe 1`

Sets the GOTO state to ON.

Related Commands

`SEQuence:ELEM[n]:GOTO:INDex (on page 75)`
`SEQuence:LENGth (on page 88)`

SEQuence:ELEM[n]:MARKer:STATe

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:MARKer:STATe <marker_state>
SEQuence:ELEM[n]:MARKer:STATe?
```

Arguments

`<marker_state>::=<Boolean>`

`0` indicates OFF.

`1` indicates ON.

At `*RST`, this returns 0.

The value of `<n>` is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

```
SEQuence:ELEM1:MARKer:STATE 1
```

Sets the marker state to ON.

Related Commands

`SEQuence:LENGth (on page 88)`

SEQuence:ELEM[n]:JTARget:INDex

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

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:JTARget:INDex <target>
SEQuence:ELEM[n]:JTARget:INDex?
```

Arguments

`<target>::=<NR1>`

Ranges from 1 to 256. It is related to the current sequence length and run mode. It should be no bigger than current sequence table size.

The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

`SEQuence:ELEM1:JTARGET:INDEX 10`

Sets the jump target index to 10th element.

Related Commands

`SEQuence:LENGth` (on page 88)
`SEQuence:ELEM` (on page 75)
`SEQuence:ELEM[n]:JTARget:TYPE`

SEQuence:ELEM[n]:JTARget:TYPE

This command and query sets or returns the event jump target type for the jump.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:JTARget:TYPE {INDex|NEXT|OFF}
SEQuence:ELEM[n]:JTARget:TYPE?
```

Arguments

`INDex`

Enables the sequencer to jump to an index set using `SEQuence:ELEM1:JTARget:INDex` command.

`NEXT`

Enables the sequencer to jump to the next sequence element.

`SEQuence:ELEM1:JTARget:INDex`

Setting is ignored.

`OFF`

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

The value of `<n>` is an index number of sequence, ranging from 1 to 256.

Returns

```
{IND|NEXT|OFF}
```

Examples

```
SEQuence:ELEM1:JTARget:TYPE INDex
```

Sets the jump target type to INDex.

Related Commands

`SEQuence:LENGth` (on page 88)
`SEQuence:ELEM[n]:JTARget:INDex` (on page 78)

SEQuence:ELEM[n]:JUMP:EVENt

This command and query Set or returns the jump trigger event type. If `SEQuence:ELEM[n]:JTARget:TYPE` is in the OFF state, the sequencer ignores trigger signals.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:JUMP:EVENt {EXT|BUS|MANual|TIMer}
SEQuence:ELEM[n]:JUMP:EVENt?
```

Arguments

`EXTernal`

Indicates trigger signal is external trigger.

`BUS`

Indicates trigger signal generated by *TRG command.

`MANual`

Indicates trigger signal generated by manual press button.

`TIMer`

Indicates trigger signal generated by internal timer.

At *RST, this returns EXT.

The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`{EXT|BUS|MAN|TIM}`

Examples

`SEQUENCE:ELEM1:JUMP:EVENt EXT`

Set the jump trigger event to EXT.

Related Commands

`SEQuence:LENGth` (on page 88)
`SEQuence:ELEM[n]:JUMP:SLOpe` (on page 81)

SEQuence:ELEM[n]:JUMP:SLOPe

This command sets or returns the slope of external trigger for jump trigger event. If `SEQuence:ELEM[n]:JTARget:TYPE` is in the `OFF` state, the sequencer ignores trigger signals.

If `SEQUENCE:ELEM[n]:JUMP:EVENt` is not the Ext, this command doesn't take effect.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:JUMP:SLOPe {POSitive|NEGative}
SEQuence:ELEM[n]:JUMP:SLOPe?
```

Arguments

`POSitive`

Indicates that the event occurs on the rising edge of the external trigger signal.

`NEGative`

Indicates that the event occurs on the falling edge of the external trigger signal.

At `*RST,` this returns POSitive.

The value of `<n>` is an index number of sequence, ranging from 1 to 256.

Returns

`{POS|NEG}`

Examples

`SEQuence:ELEM1:JUMP:SLOPe POSitive`

Sets the trigger slope to positive, which triggers on the rising edge of the signal.

Related Commands

`SEQuence:LENGth (on page 88)`
`SEQuence:ELEM[n]:JUMP:EVENt (on page 80)`

SEQuence:ELEM[n]:LOOP:COUNt

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

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:LOOP:COUNt {<NR1>|MINimum|MAXimum}
SEQuence:ELEM[n]:LOOP:COUNt?
```

Arguments

<NR1>

The value ranges between 1 and 1E6.

At `*RST`, this returns 1.

The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

```
SEQuence:ELEM1:LOOP:COUNt 100
```

Sets the element loop count to 100.

Related Commands

`SEQuence:LENGth (on page 88)`
`SEQuence:ELEM[n]:LOOP:INFinite (on page 83)`

SEQuence:ELEM[n]:LOOP:INFinite

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

This command:

Group

Sequence

Syntax

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

Arguments

`<loop_state>::=<Boolean>`

`0` indicates OFF

`1` indicates ON

At `*RST`, this returns 0.The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

```
SEQuence:ELEM1:LOOP:INFinite 1
```

Sets the infinite flag to ON.

Related Commands

`SEQuence:LENGth (on page 88)`
`SEQuence:ELEM[n]:LOOP:COUNt (on page 82)`

SEQuence:ELEM[n]:TWAit[:STATe]

This command and query sets or returns the wait trigger state for an element.

Send a trigger signal, using an internal timer, in one of the following ways:

- By using an external trigger signal.
- By pressing the "Force Trigger" button on the front panel.
- By sending the `*TRG` remote command.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:TWAit[:STATe] <wait_trigger_state>
SEQuence:ELEM[n]:TWAit[:STATe]?
```

Arguments

`<wait_trigger_state>::=<Boolean>`

`0` indicates OFF

`1` indicates ON

At `*RST`, this returns 0.The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`<NR1>`

Examples

`SEQuence:ELEM1:TWAit 1`

Sets the wait trigger state to ON.

Related Commands

`SEQuence:LENGth (on page 88)`
`SEQuence:ELEM[n]:TWAit:EVENt (on page 85)`
`SEQuence:ELEM[n]:TWAit:SLOpe (on page 86)`

SEQuence:ELEM[n]:TWAit:EVENt

This command and query Set or returns the wait trigger event type. If `SEQuence:ELEM[n]:TWAit` is in the `OFF` state, the sequencer ignores trigger signals.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:TWAit:EVENt {EXT|BUS|MANual|TIMer}
SEQuence:ELEM[n]:TWAit:EVENt?
```

Arguments

`EXT`

Indicates trigger signal is external trigger.

`BUS`

Indicates trigger signal generated by *TRG command.

`MANual`

Indicates trigger signal generated by manual press button.

`TIMer`

Indicates trigger signal generated by internal timer.

At `*RST`, this returns EXT.

The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`{EXT|BUS|MAN|TIM}`

Examples

`SEQuence:ELEM1:TWAit:EVENt EXT`

Set the wait trigger event of element 1 to EXT.

Related Commands

`SEQuence:LENGth (on page 88)`
`SEQuence:ELEM[n]:TWAit:SLOpe (on page 86)`

SEQuence:ELEM[n]:TWAit:SLOPe

This command sets or returns the slope of external trigger for wait trigger event. If `SEQuence:ELEM[n]:TWAit` is in the `OFF` state, the sequencer ignores trigger signals.

If `SEQuence:ELEM[n]:TWAit:EVENt` is not the Ext, this command don’t take effect.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:TWAit:SLOPe {POSitive|NEGative}
SEQuence:ELEM[n]:TWAit:SLOPe?
```

Arguments

`POSitive`

Indicates that the event occurs on the rising edge of the external trigger signal.

`NEGative`

Indicates that the event occurs on the falling edge of the external trigger signal.

At `*RST,` this returns POS.

The value of <n> is an index number of sequence, ranging from 1 to 256.

Returns

`{POS|NEG}`

Examples

`SEQuence:ELEM1:TWAit:SLOPe POSitive`

Sets the trigger slope of element 1 to positive, which triggers on the rising edge of the signal.

Related Commands

`SEQuence:LENGth` (on page 88)
`SEQuence:ELEM[n]:TWAit:EVENt` (on page 85)

SEQuence:ELEM[n]:WAVeform[m]

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

NOTE. The value of `n` indicates index number of sequence.The value of `m` = 1|2 is based on the model. If the suffix is omitted, 1 is assumed.The value of m indicates the channel that will output the waveform when the sequence is run.The length of all the waveforms specified for a sequence element must be equal.

This command:

Group

Sequence

Syntax

```
SEQuence:ELEM[n]:WAVeform[1|2] <wfm_name>
SEQuence:ELEM[n]:WAVeform[1|2]?
```

Arguments

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

Indicates that the waveform path and name.

M:/ indicates the internal memory.

U:/ indicates the external USB memory.

P:/ indicates the internal predefine waveforms.

The value of `<n>` is an index number of sequence, ranging from 1 to 256.

Returns

```
<string>
```

Examples

```
SEQuence:ELEM1:WAVeform1 "P:/Pulse1000.tfwx"
```

Sets the `Pulse1000.tfwx` waveform from internal driver into the first element of the sequence.

Related Commands

SEQuence:LENGth (on page 88)

SEQuence:LENGth

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

This command:

Group

Sequence

Syntax

```
SEQuence:LENGth <NR1>
SEQuence:LENGth?
```

Arguments

`<NR1>`

Ranges from 0 to 256.

At `*RST`, this returns 0.

Returns

`<NR1>`

Examples

`SEQuence:LENGth 10`

Creates a sequence of 10 elements initializing all sequence parameters to default values.

`SEQuence:LENGth?`

Now returns 10.

SEQuence:NEW (No Query Form)

This command creates a new sequence with no name, empty steps,and no tracks. This command clears the waveform list and sets the instrument settings to default. Its function is the same as `SEQC:RESET.`

This command:

Group

Sequence

Syntax

```
SEQuence:NEW
```

Examples

`SEQuence:NEW`

Creates a new sequence.

[SOURce[1|2]]:AM[:DEPTh]

This command sets or queries the modulation depth of AM modulation for the specified channel. You can set the modulation depth from 0.0% to 120.0% with resolution of 0.1%.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM[:DEPTh]{<depth>|MINimum|MAXimum}
[SOURce[1|2]]:AM[:DEPTh]?
```

Arguments

`<depth>::=<NR2>[<units>]`

where:

`<NR2>` is the depth of modulating frequency.

`<units>::=PCT`

MINimum sets the modulation depth to minimum value.

MAXimum sets the modulation depth to maximum value.

Returns

`<depth>`

Examples

`SOURce1:AM:DEPth MAXimum`

Sets the depth of modulating signal on CH1 to the maximum value.

[SOURce[1|2]]:AM:INTernal:FREQuency

This command sets or queries the internal modulation frequency of AM modulation for the specified channel. You can use this command only when the internal modulation source is selected. You can set the internal modulation frequency from 1 mHz to 1 MHZ with resolution of 1 mHz.

You can select the source of modulating signal by using the `[SOURce[1|2]]:AM:SOURce [INTernal|EXTernal]` command.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM:INTernal:FREQuency {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:AM:INTernal:FREQuency?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the modulation frequency.

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

Returns

`<frequency>`

Examples

`SOURce1:AM:INTernal:FREQuency 10kHz`

Sets the CH1 internal modulation frequency to 10 kHz.

Related Commands

`[SOURce[1|2]]:AM:SOURce (on page 93)`

[SOURce[1|2]]:AM:INTernal:FUNCtion

This command sets or queries the modulating waveform of AM modulation for the specified channel. You can use this command only when the internal modulation source is selected.

If you specify EFILe when there is no EFILe or the EFILe is not yet defined, this command causes an error. You should set EFILe, then run this command again.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM:INTernal:FUNCtion
{SINusoid|SQUare | TRIangl | RAMP | NRAMp | PRNoise | EMEMory[1] | EMEMory2 | EFILe}
[SOURce[1|2]]:AM:INTernal:FUNCtion?
```

Arguments

`SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise`

One of six types of function waveform can be selected as a modulating signal.

`EMEMory[1] | EMEMory2`

A user defined waveform saved in the user waveform memory or the EMEMory can be selected as a modulating signal.

`EFILe`

EFILe is used as a modulating signal.

Returns

`SIN|SQU|TRI|RAMP|NRAM|PRN|EMEM|EFIL`

Examples

`SOURce1:AM:INTernal:FUNCtion SQUare`

Selects Square as the shape of modulating waveform for the CH1 output.

Related Commands

`[SOURce[1|2]]:AM:SOURce (on page 93)`
`[SOURce[1|2]]:AM:INTernal:FUNCtion:EFILe (on page 91)`

[SOURce[1|2]]:AM:INTernal:FUNCtion:EFILe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM:INTernal:FUNCtion:EFILe <file_name>
[SOURce[1|2]]:AM:INTernal: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 (/)

Returns

`<file_name>`

Examples

```
SOURce1:AM:INTernal:FUNCtion:EFILe “U:\SAMPLE1.tfwx”
```

Sets a file named “U:\SAMPLE1.tfwx” in the mass storage as the modulating waveform for AM modulation.

[SOURce[1|2]]:AM:SOURce

This command sets or queries the source of modulating signal of AM modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM:SOURce {INTernal|EXTernal}
[SOURce[1|2]]:AM:SOURce?
```

Arguments

`INTernal`

Means that the carrier waveform is modulated with an internal source.

`EXTernal`

Means that the carrier waveform is modulated with an external source.

Returns

`INT|EXT`

Examples

`SOURce1:AM:SOURce INTernal`

Sets the CH1 source of modulating signal to internal.

[SOURce[1|2]]:AM:STATe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:AM:STATe {ON|OFF|<NR1>}
[SOURce[1|2]]:AM:STATe?
```

Arguments

`ON or <NR1>≠0`

Enables AM modulation.

`OFF or <NR1>=0`

Disables AM modulation.

Returns

`<NR1>`

Examples

`SOURce1:AM:STATe ON`

Enables the CH1 AM modulation.

[SOURce[1|2]:]BURSt:INFInite:REARm (no query form)

This command stops output, then sets burst state to wait trigger.

This command:

Group

Source

Syntax

```
[SOURce[1|2]:]BURSt:INFInite:REARm
```

Examples

`SOURce:BURSt:INFInite:REARm`

Stop CH1 output, waiting next trigger.

[SOURce[1|2]:]BURSt:IDLE

This command sets or queries the idle state of burst. Idle state means the output level between two burst output.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:BURSt:IDLE {START|DC|END}
[SOURce[1|2]]:BURSt:IDLE?
```

Arguments

`START`

The output stays the same as the first point of the burst waveform.

`DC`

The output keep the DC.

`END`

Returns

`START|DC|END`

Examples

`SOURce1:BURSt:IDLE START`

Sets the idle state to the first point of burst waveform.

[SOURce[1|2]]:BURSt:MODE

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:BURSt:MODE {TRIGgered|GATed}
[SOURce[1|2]]:BURSt:MODE?
```

Arguments

`TRIGgered`

Means that triggered mode is selected for burst mode.

`GATed`

Means gated mode is selected for burst mode.

Returns

`TRIG|GAT`

Examples

`SOURce1:BURSt:MODE TRIGgered`

Selects triggered mode.

[SOURce[1|2]]: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.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:BURSt:NCYCles {<cycles>|INFinity|MINimum|MAXimum}
[SOURce[1|2]]:BURSt:NCYCles?
```

Arguments

`<cycles>::=<NRf>`

where:

`<NRf>` is the burst count; the burst count ranges from 1 to 1,000,000.

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 CH1 burst count to 2.

[SOURce[1|2]]:BURSt[:STATe]

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:BURSt[:STATe] {ON|OFF|<NR1>}
[SOURce[1|2]]: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 CH1.

[SOURce[1|2]]:BURSt:TDELay

This command sets or queries delay time in the burst mode for the specified channel. It specifies a time delay between the trigger and the signal output. This command is available only in the Triggered burst mode.

The setting range is 0.0 ns to 85.000 s with resolution of 100 ps or 5 digits.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:BURSt:TDELay {<delay>|MINimum|MAXimum}
[SOURce[1|2]]:BURSt:TDELay?
```

Arguments

`<delay>::=<NRf>[<units>]`

where: `<units>::=[s | ms | us | ns]`

MINimum sets the trig delay time to minimum value.

MAXimum sets the trig delay time to maximum value.

Returns

`<delay>`

Examples

`SOURce1:BURSt:TDELay 20ms`

Sets the CH1 trig delay time to 20 ms.

[SOURce[1|2]]:COMBine:FEED

This command sets or queries whether to add the internal noise or an external signal to an output signal for the specified channel.

When you specify the internal noise, you can set or query the noise level by the `SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude]` command.

To disable the internal noise add or the external signal add function, specify “ ”.

You can add an external signal to the CH1 output signal of the AFG31000 Series.

The CH2 output is not available for adding external signal.

Both the internal noise and an external signal can be added simultaneously to the AFG.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:COMBine:FEED {"NOISe"|"EXTernal"|"BOTH"|""}
[SOURce[1|2]]:COMBine:FEED?
```

Arguments

`NOISe`

Indicates that the internal noise is added to the output signal.

`EXTernal`

Indicates that an external signal is added to the CH1 output signal.

`BOTH`

Indicates that the internal noise and an external signal are added to the CH1 output signal.

“ ” disables the internal noise to add and external signal add function.

Returns

`“NOIS”|“EXT”|“BOTH”|“”`

Examples

`SOURce1:COMBine:FEED "EXTernal"`

Adds an external signal to the CH1 output signal.

[SOURce[1|2]]:FM[:DEViation]

This command sets or queries the peak frequency deviation of FM modulation for the specified channel. The setting range of frequency deviation depends on the waveform selected as the carrier. For more information, refer to the specifications in the AFG31000 Series Specifications and Performance Verification Manual, which can be found on the Tektronix website tek.com.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM[:DEViation] {<deviation>|MINimum|MAXimum}
[SOURce[1|2]]:FM[:DEViation]?  [MINimum|MAXimum]
```

Arguments

`<deviation>::=<NRf>[<units>]`

where:

`<NRf>` is the frequency deviation.

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

Returns

`<deviation>`

Examples

`SOURce1:FM:DEViation 1.0MHz`

Sets the CH1 frequency deviation to 1.0 MHz.

[SOURce[1|2]]:FM:INTernal:FREQuency

This command sets or queries the internal modulation frequency of FM modulation for the specified channel. You can use this command only when the internal modulation source is selected. You can set the internal modulation frequency from 1 mHz to 1 MHz with resolution of 1 mHz. You can select the source of modulating signal by using the `[SOURce[1|2]]:FM:SOURce {INTernal|EXTernal}` command.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM:INTernal:FREQuency {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FM:INTernal:FREQuency?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the modulation frequency.

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

Returns

`<frequency>`

Examples

`SOURce1:FM:INTernal:FREQuency 10kHz`

Sets the CH1 internal modulation frequency to 10 kHz.

Related Commands

`[SOURce[1|2]]:FM:SOURce (on page 104)`

[SOURce[1|2]]:FM:INTernal:FUNCtion

This command sets or queries the modulating waveform of FM modulation for the specified channel. You can use this command only when the internal modulation source is selected.

If you specify EFILe when there is no EFILe or the EFILe is not yet defined, this command causes an error. You should set EFILe, then run this command again.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM:INTernal:FUNCtion
{SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise|EMEMory[1]|EMEMory2|EFILe}
[SOURce[1|2]]:FM:INTernal:FUNCtion?
```

Arguments

```
SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise|EMEMory[1]|EMEMory2|EFILe
```

Returns

```
SIN|SQU|TRI|RAMP|NRAM|PRN|EMEM|EFIL
```

Examples

```
SOURce1:FM:INTernal:FUNCtion SQUare
```

Selects Square as the shape of modulating waveform for the CH1 output.

Related Commands

`[SOURce[1|2]]:FM:SOURce` (on page 104)

[SOURce[1|2]]:FM:INTernal:FUNCtion:EFILe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM:INTernal:FUNCtion:EFILe <file_name>
[SOURce[1|2]]:FM:INTernal: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 (/).

Returns

`<file_name>`

Examples

`SOURce1:FM:INTernal:FUNCtion:EFILe “SAMPLE1”`

Sets a file named `“SAMPLE1”` in the mass storage.

[SOURce[1|2]]:FM:SOURce

This command sets or queries the source of the modulating signal of FM modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM:SOURce {INTernal|EXTernal}
[SOURce[1|2]]:FM:SOURce?
```

Arguments

`INTernal`

Means that the carrier waveform is modulated with the internal source.

`EXTernal`

Means that the carrier waveform is modulated with an external source.

Returns

`INT|EXT`

Examples

`SOURce1:FM:SOURce INTernal`

Sets the CH1 source of modulating signal to internal.

[SOURce[1|2]]:FM:STATe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FM:STATe {ON|OFF|<NR1>}
[SOURce[1|2]]: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 CH1 FM modulation.

[SOURce[1|2]]:FREQuency:CENTer

This command sets or queries the center frequency of sweep for the specified channel. This command is always used with the `[SOURce[1|2]]:FREQuency:SPAN` command. The setting range of center frequency depends on the waveform selected for sweep.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:CENTer {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FREQuency:CENTer?
```

Arguments

```
<frequency>::=<NRf>[<units>]
```

where:

`<NRf>` is the center frequency.

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

Returns

```
<frequency>
```

Examples

```
SOURce1:FREQuency:CENTer 550kHz
```

Sets the CH1 center frequency to 550 kHz.

Related Commands

`[SOURce[1|2]]:FREQuency:SPAN` (on page 110)
`[SOURce[1|2]]:FREQuency:MODE` (on page 109)

[SOURce[1|2]]:FREQuency:CONCurrent[: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:CONCurrent` command copies the frequency (or period) of the channel specified by the header suffix to another channel. If you specify CH1 with the header, the CH1 frequency will be copied to CH2.

The `[SOURce[1|2]]:FREQuency:CONCurrent?` command returns 0 (OFF) or 1 (ON).

If your AFG is single-channel model, this command is not supported.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:CONCurrent[:STATe] {ON|OFF|<NR1>}
[SOURce[1|2]]:FREQuency:CONCurrent[:STATe]?
```

Arguments

`ON or <NR1>≠0`

Enables the concurrent copy function.

`OFF or <NR1>=0`

Disables the concurrent copy function.

Returns

<NR1>

Examples

`SOURce1:FREQuency:CONCurrent ON`

Copies the frequency value of CH1 to CH2.

[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 μHz or 12 digits. For more information on the setting range, refer to the AFG31000 Series Specifications and Performance Verification Technical Reference, which can be found on the Tektronix website tek.com.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency[:CW|:FIXed] {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FREQuency[:CW|:FIXed]?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the output frequency

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

Returns

`<frequency>`

Examples

`SOURce1:FREQuency:FIXed 500kHz`

Sets the CH1 output frequency to 500 kHz when the Run Mode is set to other than sweep.

[SOURce[1|2]]:FREQuency:MODE

This command sets or queries the frequency sweep state. You can select sine, square, ramp, or arbitrary waveform for sweep. The AFG automatically changes to the Continuous mode if any waveform is selected other than sine, square, ramp, or an arbitrary waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:MODE {CW|FIXed|SWEep}
[SOURce[1|2]]:FREQuency:MODE?
```

Arguments

`CW|FIXed`

Means that the frequency is controlled by the [SOURce[1|2]]: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|SWE`

Examples

`SOURce1:FREQuency:MODE SWEep`

Specifies the sweep command set for controlling the CH1 output frequency.

Related Commands

`[SOURce[1|2]]:FREQuency[:CW|:FIXed]` (on page 108)
`[SOURce[1|2]]:FREQuency:CENTer` (on page 106)
`[SOURce[1|2]]:FREQuency:SPAN` (on page 110)
`[SOURce[1|2]]:FREQuency:STARt` (on page 111)
`[SOURce[1|2]]:FREQuency:STOP` (on page 112)

[SOURce[1|2]]:FREQuency:SPAN

This command sets or queries the span of frequency sweep for the specified channel. This command is always used with the `[SOURce[1|2]]:FREQuency:CENTer` command. The setting range of frequency span depends on the waveform selected for sweep.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:SPAN {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FREQuency:SPAN?
```

Arguments

```
<frequency>::=<NRf>[<units>]
```

where:

`<NRf>` is the frequency span

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

Returns

```
<frequency>
```

Examples

```
SOURce1:FREQuency:SPAN 900kHz
```

Sets the CH1 frequency span to 900 kHz.

Related Commands

`[SOURce[1|2]]:FREQuency:CENTer` (on page 106)
`[SOURce[1|2]]:FREQuency:MODE` (on page 109)

[SOURce[1|2]]:FREQuency:STARt

This command sets or queries the start frequency of sweep for the specified channel. This command is always used with the `[SOURce[1|2]]:FREQuency:STOP` command. The setting range of start frequency depends on the waveform selected for sweep. For more information on the setting range, refer to the AFG31000 Series Specifications and Performance Verification Manual, which can be found on the Tektronix website tek.com.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:STARt {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FREQuency:STARt?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the start frequency.

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

Returns

`<frequency>`

Examples

`SOURce1:FREQuency:STARt 10kHz`

Sets the sweep start frequency of CH1 to 10 kHz.

Related Commands

`[SOURce[1|2]]:FREQuency:MODE (on page 109)`
`[SOURce[1|2]]:FREQuency:STOP (on page 112)`

[SOURce[1|2]]:FREQuency:STOP

This command sets or queries the stop frequency of sweep for the specified channel. This command is always used with the `[SOURce[1|2]]:FREQuency:STARt` command. The setting range of stop frequency depends on the waveform selected for sweep. For more information on the setting range, refer to the AFG31000 Series Specifications and Performance Verification Manual, which can be found on the Tektronix website tek.com.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FREQuency:STOP {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FREQuency:STOP?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the stop frequency.

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

Returns

`<frequency>`

Examples

`SOURce1:FREQuency:STOP 100kHz`

Sets the stop frequency of CH1 to 100 kHz.

Related Commands

`[SOURce[1|2]]:FREQuency:MODE (on page 109)`
`[SOURce[1|2]]:FREQuency:STOP (on page 112)`

[SOURce[1|2]]:FSKey[:FREQuency]

This command sets or queries the hop frequency of FSK modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FSKey[:FREQuency] {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:FSKey[:FREQuency]?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the hop frequency

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

Returns

`<frequency>`

Examples

`SOURce1:FSKey:FREQuency 1.0MHz`

Sets the hop frequency of CH1 FSK modulation to 1.0 MHz.

[SOURce[1|2]]:FSKey:INTernal:RATE

This command sets or queries the internal modulation rate of FSK modulation for the specified channel. You can use this command only when the internal modulation source is selected.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FSKey:INTernal:RATE {<rate>|MINimum|MAXimum}
[SOURce[1|2]]:FSKey:INTernal:RATE?
```

Arguments

`<rate>::=<NRf>[<units>]`

where:

`<NRf>` is the modulation rate

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

Returns

`<rate>`

Examples

`SOURce1:FSKey:INTernal:RATE 50Hz`

Sets the CH1 internal modulation rate to 50 Hz.

[SOURce[1|2]]:FSKey:SOURce

This command sets or queries the source of modulation signal of FSK modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FSKey:SOURce {INTernal|EXTernal}
[SOURce[1|2]]:FSKey:SOURce?
```

Arguments

`INTernal`

Means that the carrier waveform is modulated with an internal source.

`EXTernal`

Means that the carrier waveform is modulated with an external source.

Returns

`INT|EXT`

Examples

`SOURce1:FSKey:SOURce INTernal`

Sets the CH1 source of modulating signal to internal.

[SOURce[1|2]]: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 for the carrier waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FSKey:STATe {ON|OFF|<NR1>}
[SOURce[1|2]]: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.

This command:

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 (/).

Returns

`<file_name>`

Examples

`SOURce1:AM:INTernal:FUNCtion:EFILe “U:\SAMPLE1.tfwx”`

Sets a file named `“SAMPLE1.tfwx”` in the mass storage as the modulating waveform for AM modulation.

[SOURce[1|2]]:FUNCtion:RAMP:SYMMetry

This command sets or queries the symmetry of ramp waveform for the specified channel. The setting range is 0.0% to 100.0%.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FUNCtion:RAMP:SYMMetry {<symmetry>|MINimum|MAXimum}
[SOURce[1|2]]:FUNCtion:RAMP:SYMMetry?
```

Arguments

`<symmetry>::=<NR2>[<units>]`

where:

`<NR2>` is the symmetry

`<units>::=PCT`

Returns

`<symmetry>`

Examples

`SOURce1:FUNCtion:RAMP:SYMMetry 80.5`

Sets the symmetry of the CH1 ramp waveform to 80.5%.

[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.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:FUNCtion[:SHAPe]
{SINusoid|SQUare|PULSe|RAMP|PRNoise|DC|SINC|GAUSsian|LORentz|ERISe|EDECay|HAVersine
    |EMEMory[1]|EMEMory2|EFILe}
[SOURce[1|2]]:FUNCtion[:SHAPe]?
```

Arguments

```
SINusoid|SQUare|PULSe|RAMP|PRNoise|DC|SINC|GAUSsian|LORentz|ERISe|EDECay|HAVersine|
    EMEMory[1]|EMEMory2|EFILe
```

The next table shows the shape of the output waveform and the modulation types for each waveform.

Output waveform	Modulation types
Sine	AM, FM, PM, FSK, Sweep, Burst
Square	
Ramp	
Arb	
Sin(x)/x	
Gaussian	
Lorentz	
Exponential rise	
Exponential decay	
Haversine	
Pulse	PWM, Burst

If you select a waveform shape that is not allowed with a particular modulation, sweep, or burst, Run mode automatically changes to Continuous.

If you specify *EFILe* when there is no *EFILe* or the *EFILe* is not yet defined, an error occurs. You should set EFILe, then run this command again.

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. `EMEMory[1]|EMEMory2`

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|DC|SINC|GAUS|LOR|ERIS|EDEC|HARV|EMEM|EFIL
```

Examples

```
SOURce1:FUNCtion:SHAPe SQUare
```

Selects the shape of CH1 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 you select a waveform other than DC, Noise, and Pulse.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PHASe[:ADJust] {<phase>|MINimum|MAXimum}
[SOURce[1|2]]:PHASe[:ADJust]?
```

Arguments

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

where:

`<NR3>` is the phase of output frequency.

`<units>::=[RAD | DEG]`

If `<units>` are omitted, RAD is specified automatically; the setting ranges are:

`RAD`: -1 PI to +1 PI, relative to phase value.

`DEG`: -180 to +180, relative to phase value.

Returns

`<phase>`

Examples

`SOURce1:PHASe:ADJust MAXimum`

Sets the maximum value for the phase of CH1 output frequency.

[SOURce[1|2]]: PHASe:CONCurrent[:STATe]

This command enables or disables the function to copy the phase of one channel to another channel. The `[SOURce[1|2]]:PHASe:CONCurrent` command copies the phase of the channel specified by the header suffix to another channel. If you specify CH1 with the header, the CH1 phase will be copied to CH2. The `[SOURce[1|2]]:PHASe:CONCurrent?` command returns `"0" (OFF)` or `"1" (ON)`.

If your arbitrary function generator is a single-channel model, this command is not supported.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PHASe:CONCurrent[:STATe] {ON|OFF|<NR1>}
[SOURce[1|2]]:PHASe:CONCurrent[:STATe]?
```

Arguments

`ON or <NR1>≠0`

Enables the concurrent copy function.

`OFF or <NR1>=0`

Disables the concurrent copy function.

Returns

`<NR1>`

Examples

`SOURce1:PHASe:CONCurrent ON`

Copies the phase value of CH1 to CH2.

[SOURce[1|2]]:PHASe:INITiate (no query form)

This command synchronizes the phase of CH1 and CH2 output waveforms. The AFG performs the same operation if you specify either SOURce1 or SOURce2. If your AFG is single-channel model, this command is not supported.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PHASe:INITiate
```

Examples

`SOURce1:PHASe:INITiate`

Synchronizes the phase of CH1 and CH2 output signals.

[SOURce[1|2]]:PM[:DEViation]

This command sets or queries the peak frequency deviation of PM modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM[:DEViation] {<deviation>|MINimum|MAXimum}
[SOURce[1|2]]:PM[:DEViation]?
```

Arguments

`<deviation>::=<NR3>[<units>]`

where:

`<NR3>` is the phase deviation

`<units>::=[RAD | DEG]`

If `<units>` are omitted, RAD is specified automatically.

The setting ranges are:

`RAD`: 0 PI to +1 PI, relative to phase value.

`DEG`: 0 to +180, in 1 degree steps, relative to phase value.

Returns

`<deviation>`

Examples

`SOURce1:PM:DEViation MAXimum`

Sets the maximum value for the CH1 phase deviation.

[SOURce[1|2]]:PM:INTernal:FREQuency

This command sets or queries the internal modulation frequency of PM modulation for the specified channel. You can use this command only when the internal modulation source is selected. You can set the internal modulation frequency from 1 mHz to 1 MHz with a resolution of 1 mHz. You can select the source of modulating signal by using the `[SOURce[1|2]]:PM:SOURce {INTernal|EXTernal}` command.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM:INTernal:FREQuency {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:PM:INTernal:FREQuency?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the modulation frequency.

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

Returns

`<frequency>`

Examples

`SOURce1:PM:INTernal:FREQuency 10kHz`

Sets the CH1 internal modulation frequency to 10 kHz.

Related Commands

`[SOURce[1|2]]:PM:SOURce (on page 125)`

[SOURce[1|2]]:PM:INTernal:FUNCtion

This command sets or queries the modulating waveform of PM modulation for the specified channel. You can use this command only when the internal modulation source is selected.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM:INTernal:FUNCtion
{SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise|EMEMory[1]|EMEMory2|EFILe}
[SOURce[1|2]]:PM:INTernal:FUNCtion?
```

Arguments

`SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise`

One of six types of function waveform can be selected as a modulating signal.

`EMEMory[1]|EMEMory2`

A user defined waveform saved in the user waveform memory or the EMEMory can be selected as a modulating signal.

`EFILe`

EFILe is used as a modulating signal.

Returns

`SIN|SQU|TRI|RAMP|NRAM|PRN|EMEM|EFIL`

Examples

`SOURce1:PM:INTernal:FUNCtion SQUare`

Selects Square as the shape of modulating waveform for the CH1 output.

Related Commands

`[SOURce[1|2]]:PM:SOURce (on page 125)`

[SOURce[1|2]]:PM:INTernal:FUNCtion:EFILe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM:INTernal:FUNCtion:EFILe <file_name>
[SOURce[1|2]]:PM:INTernal: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 (/).

Returns

`<file_name>`

Examples

`SOURce1:PM:INTernal:FUNCtion:EFILe “M:/SAMPLE1.tfwx"`

Sets a file named “`SAMPLE1.tfwx`" in the internal mass storage.

[SOURce[1|2]]:PM:SOURce

This command sets or queries the source of modulation signal of PM modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM:SOURce {INTernal|EXTernal}
[SOURce[1|2]]:PM:SOURce?
```

Arguments

`INTernal`

Means that the carrier waveform is modulated with an internal source.

`EXTernal`

Means that the carrier waveform is modulated with an external source.

Returns

`INT|EXT`

Examples

`SOURce1:PM:SOURce INTernal`

Sets the CH1 source of modulating signal to internal.

[SOURce[1|2]]: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 for the carrier waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PM:STATe {ON|OFF|<NR1>}
[SOURce[1|2]]: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 CH1 PM modulation.

[SOURce[1|2]]:PULSe:DCYCle

This command sets or queries the duty cycle of the pulse waveform for the specified channel. The setting range is 0.001% to 99.999% in increments of 0.001.

The instrument will hold the settings of leading edge and trailing edge when the duty cycle is varied.

Refer to the `[SOURce[1|2]]:PULSe:WIDTh` command for the setting range.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:DCYCle {<percent>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:DCYCle?
```

Arguments

`<percent>::=<NR2>[<units>]`

where:

`<NR2>` is the duty cycle.

`<units>::=PCT`

Returns

`<percent>`

Examples

`SOURce1:PULSe:DCYCle 80.5`

Sets the duty cycle of the pulse waveform on CH1 to 80.5%.

Related Commands

`[SOURce[1|2]]:PULSe:WIDTh (on page 133)`

[SOURce[1|2]]:PULSe:DELay

This command sets or queries the lead delay of the pulse waveform for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:DELay {<delay>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:DELay?
```

Arguments

`<delay>::=<NR2>[<units>]`

where:

`<NR2>` is the lead delay.

`<units>::=[ns | µs | ms | s]`

Setting range:

```
0 ns to Pulse Period (Continuous mode)
0 ns to Pulse Period - {Pulse Width + 0.8 * (Leading Edge Time + Trailing Edge Time)}
(Triggered/Gated burst mode)
```

Returns

`<delay>`

Examples

`SOURce1:PULSe:DELay 20ms`

Sets the CH1 lead delay to 20 ms.

[SOURce[1|2]]:PULSe:HOLD

The `[SOURce[1|2]]:PULSe:HOLD` command sets the instrument to hold either pulse width or pulse duty.

The `[SOURce[1|2]]:PULSe:HOLD?` query returns WIDTh or DUTY.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:HOLD {WIDTh|DUTY}
[SOURce[1|2]]:PULSe:HOLD?
```

Arguments

`WIDTh`

Means that the AFG holds the pulse width setting.

`DUTY`

Means that the AFG holds the pulse duty setting.

Returns

`WIDT|DUTY`

Examples

`SOURce1:PULSe:HOLD WIDTh`

Holds the CH1 pulse width setting.

[SOURce[1|2]]:PULSe:PERiod

This command sets or queries the period for pulse waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:PERiod {<period>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:PERiod?
```

Arguments

`<period>::=<NRf>[<units>]`

where:

`<NRf>` is the pulse period.

`<units>::=[ns | µs | ms | s]`

Returns

`<period>`

Examples

`SOURce1:PULSe:PERiod 200ns`

Sets the CH1 pulse period to 200 ns.

[SOURce[1|2]]:PULSe:TRANsition[:LEADing]

This command sets or queries the leading edge time of pulse waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:TRANsition[:LEADing] {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:TRANsition[:LEADing]?
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

`<NRf>` is the leading edge time of pulse waveform.

`<units>::=[ns | µs | ms | s]`

Returns

`<seconds>`

Examples

`SOURce1:PULSe:TRANsition:LEADing 200ns`

Sets the CH1 leading edge time to 200 ns.

[SOURce[1|2]]:PULSe:TRANsition:TRAiling

This command sets or queries the trailing edge time of pulse waveform.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:TRANsition:TRAiling {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:TRANsition:TRAiling?
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

`<NRf>` is the trailing edge of pulse waveform

`<units>::=[ns | µs | ms | s]`

Returns

`<seconds>`

Examples

`SOURce1:PULSe:TRANsition:TRAiling 200ns`

Sets the trailing edge time to 200 ns.

[SOURce[1|2]]:PULSe:WIDTh

This command sets or queries the pulse width for the specified channel.

Pulse Width = Period * Duty Cycle / 100

The pulse width must be less than the period. The setting range is 0.001% to 99.999% in terms of duty cycle.

- AFG31021 / 31022: 16 ns to 999.99 s
- AFG31051 / 31052: 10 ns to 999.99 s
- AFG31101 / 31102: 6 ns to 999.99 s
- AFG31151 / 31152: 5 ns to 999.99 s
- AFG31251 / 31252: 4 ns to 999.99 s

Pulse Width ≤ Pulse Period - 0.8 * (Leading Edge Time + Trailing Edge Time).

Pulse Width ≥ 0.625 * (Leading Edge Time + Trailing Edge Time).

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PULSe:WIDTh {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:PULSe:WIDTh?
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

`<NRf>` is the pulse width

`<units>::=[ns | µs | ms | s]`

Returns

`<seconds>`

Examples

`SOURce1:PULSe:WIDTh 200ns`

Sets the CH1 pulse width to 200 ns

Related Commands

`[SOURce[1|2]]:PULSe:DCYCle` (on page 127)

[SOURce[1|2]]:PWM:INTernal:FREQuency

This command sets or queries the internal modulation frequency of PWM modulation for the specified channel. You can use this command only when the internal modulation source is selected.

You can set the internal modulation frequency from 1 mHz to 1 MHz with resolution of 1 mHz.

You can select the source of modulating signal by using the `[SOURce[1|2]]:PWM:SOURce {INTernal|EXTernal}` command.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM:INTernal:FREQuency {<frequency>|MINimum|MAXimum}
[SOURce[1|2]]:PWM:INTernal:FREQuency?
```

Arguments

`<frequency>::=<NRf>[<units>]`

where:

`<NRf>` is the modulation frequency

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

Returns

`<frequency>`

Examples

`SOURce1:PWM:INTernal:FREQuency 10kHz`

Sets the CH1 internal frequency to 10 kHz.

Related Commands

`[SOURce[1|2]]:PWM:SOURce (on page 125)`

[SOURce[1|2]]:PWM:INTernal:FUNCtion

This command sets or queries the modulating waveform of PWM modulation for the specified channel. You can use this command only when the internal modulation source is selected.

If you specify EFILe when there is no EFILe or the EFILe is not yet defined, this command causes an error. You should set EFILe, then run this command again.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM:INTernal:FUNCtion
{SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise|EMEMory[1]|EMEMory2|EFILe}
[SOURce[1|2]]:PWM:INTernal:FUNCtion?
```

Arguments

`SINusoid|SQUare|TRIangle|RAMP|NRAMp|PRNoise`

One of six types of function waveform can be selected as a modulating signal.

`EMEMory[1]|EMEMory2`

A user defined waveform saved in the user waveform memory or the EMEMory can be selected as a modulating signal.

`EFILe`

EFILe is used as a modulating signal.

Returns

`SIN|SQU|TRI|RAMP|NRAM|PRN|EMEM|EFIL`

Examples

`SOURce1:PWM:INTernal:FUNCtion SQUare`

Selects Square as the shape of modulating waveform for the CH1 output.

Related Commands

`[SOURce[1|2]]:PWM:SOURce (on page 125)`

[SOURce[1|2]]:PWM:INTernal:FUNCtion:EFILe

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

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM:INTernal:FUNCtion:EFILe <file_name>
[SOURce[1|2]]:PWM:INTernal:FUNCtion:EFILe?
```

Returns

```
<file_name>
```

Examples

```
SOURce1:PWM:INTernal:FUNCtion:EFILe  "U:/SAMPLE1.tfwx"
```

Set PWM internal function is SAMPLE1.tfwx in the USB memory.

[SOURce[1|2]]:PWM:SOURce

This command sets or queries the source of modulating signal of PWM modulation for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM:SOURce {INTernal|EXTernal}
[SOURce[1|2]]:PWM:SOURce?
```

Arguments

`INTernal`

Means that the carrier waveform is modulated with the internal source.

`EXTernal`

Means that the carrier waveform is modulated with an external source.

Returns

```
INT|EXT
```

Examples

```
SOURce1:PWM:SOURce INTernal
```

Sets the source of modulating signal on CH1 to internal.

[SOURce[1|2]]:PWM:STATe

This command enables or disables PWM modulation. The query command returns the state of PWM modulation. You can select only pulse waveform as a carrier waveform for PWM.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM:STATe {ON|OFF|<NR1>}
[SOURce[1|2]]:PWM:STATe?
```

Arguments

`ON or <NR1>≠0`

Enables PWM modulation.

`OFF or <NR1>=0`

Disables PWM modulation.

Returns

`<NR1>`

Examples

`SOURce1:PWM:STATe ON`

Enables the CH1 PWM modulation.

[SOURce[1|2]]:RMODe?

This query only command queries the run mode.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:RMODe?
```

Arguments

None

Returns

`CONT|MOD|SWE|BURS`

Examples

`SOURce:RMOD?`

Gets the CH1 run mode to MOD.

[SOURce[1|2]]:PWM[:DEViation]:DCYCle

This command sets or queries the PWM deviation in percent for the specified channel.

The setting range must meet the following conditions:

- Deviation ≤ Pulse Width - PWmin
- Deviation ≤ Pulse Period - Pulse Width - PWmin
- Deviation ≤ Pulse Width - 0.8 × (Leading edge time + Trailing edge time)
- Deviation ≤ Pulse Period - Pulse Width - 0.8 × (Leading edge time + Trailing edge time) where PWmin is the minimum pulse width.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PWM[:DEViation]:DCYCle {<percent>|MINimum|MAXimum}
[SOURce[1|2]]:PWM[:DEViation]:DCYCle?
```

Arguments

`<percent>::=<NR2>[<units>]`

where:

`<NR2>` is the PWM deviation.

`<units>::=PCT`

Returns

`<percent>`

Examples

`SOURce1:PWM:DCYCle 5.0`

Sets the CH1 PWM deviation to 5.0%.

[SOURce]:ROSCillator:SOURce

This command sets the reference clock to either internal or external.

This command:

Group

Source

Syntax

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

Arguments

`INTernal`

Means that the reference clock is set to internal.

`EXTernal`

Means that the reference clock is set to external.

Returns

`INT|EXT`

Examples

`SOURce:ROSCillator:SOURce INTernal`

Selects the internal clock reference.

[SOURce[1|2]]:SWEep:HTIMe

This command sets or queries the sweep hold time. Hold time represents the amount of time that the frequency must remain stable after reaching the stop frequency.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:SWEep:HTIMe {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:SWEep:HTIMe?
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

`<NRf>` is the hold time in seconds.

`<units>::=[ns | µs | ms | s]`

Returns

`<seconds>`

Examples

`SOURce1:SWEep:HTIMe 1ms`

Sets the CH1 hold time to 1 ms.

[SOURce[1|2]]:SWEep:MODE

The `[SOURce[1|2]]:SWEep:MODE` command selects auto or manual for the sweep mode for the specified channel.

The query command returns the sweep mode for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:SWEep:MODE {AUTO|MANual}
[SOURce[1|2]]:SWEep:MODE?
```

Arguments

`AUTO`

Sets the sweep mode to auto; the instrument outputs a continuous sweep at a rate specified by Sweep Time, Hold Time, and Return Time.

`MANual`

Sets the sweep mode to manual; the instrument outputs one sweep when a trigger input is received.

Returns

`AUTO|MAN`

Examples

`SOURce1:SWEep:MODE AUTO`

Sets the CH1 sweep mode to auto; the instrument outputs a continuous sweep.

Related Commands

`[SOURce[1|2]]:SWEep:HTIMe` (on page 140)
`[SOURce[1|2]]:SWEep:RTIMe` (on page 142)
`[SOURce[1|2]]:SWEep:TIME` (on page 144)
`TRIGger[:SEQuence]:SOURce` (on page 170)
`TRIGger[:SEQuence]:TIMer` (on page 170)

[SOURce[1|2]]:SWEep:RTIMe

This command sets or queries the sweep return time. Return time represents the amount of time from stop frequency through start frequency. Return time does not include hold time.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:SWEep:RTIMe {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:SWEep:RTIMe?
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

`<NRf>` is the return time in seconds.

`<units>::=[ns | µs | ms | s]`

Returns

`<seconds>`

Examples

`SOURce1:SWEep:RTIMe 1ms`

Sets the CH1 return time to 1 ms.

[SOURce[1|2]]:SWEep:SPACing

The `[SOURce[1|2]]:SWEep:SPACing` command selects linear or logarithmic spacing for the sweep for the specified channel.

The query command returns the type for the sweep spacing for the specified channel.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:PHASe:INITiate  {LINear|LOGarithmic}
[SOURce[1|2]]:SWEep:SPACing?
```

Arguments

`LINear`

Sets the sweep spacing to linear.

`LOGarithmic`

Sets the sweep spacing to logarithmic.

Returns

`LIN|LOG`

Examples

`SOURce1:SWEep:SPACing LINear`

Sets the CH1 sweep spacing to linear.

[SOURce[1|2]]:SWEep:TIME

This command sets or queries the sweep time for the sweep for the specified channel. The sweep time does not include hold time and return time. The setting range is 1 ms to 500 s.

This command:

Group

```
Source
```

Syntax

```
[SOURce[1|2]]:SWEep:TIME {<seconds>|MINimum|MAXimum}
[SOURce[1|2]]:SWEep:TIME?
```

Arguments

```
<seconds>::=<NRf>[<units>]
```

where:

`<NRf>` is the sweep time in seconds.

```
<units>::=[ns | µs | ms | s]
```

Returns

```
<seconds>
```

Examples

```
SOURce1:SWEep:TIME 100ms
```

Sets the CH1 sweep time to 100 ms.

[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]

This command enables or disables the function to copy the voltage level of one channel to another channel.

The `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command copies the voltage level of the channel specified by the header suffix to another channel. If you specify CH1 with the header, the CH1 voltage level will be copied to CH2.

The query command returns “0” (OFF) or “1” (ON). If your instrument is a single-channel model, this command is not supported.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage:CONCurrent[:STATe] {ON|OFF|<NR1>}
[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]?
```

Arguments

`ON or <NR1>≠0`

Enables the concurrent copy function.

`OFF or <NR1>=0`

Disables the concurrent copy function.

Returns

`<NR1>`

Examples

`SOURce1:VOLTage:CONCurrent:STATe ON`

Copies the voltage value of CH1 to CH2.

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

This command sets or queries the high level of output amplitude for the specified channel. If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the high level of other channel is also the same value.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:HIGH {<voltage>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:HIGH?
```

Arguments

`<voltage>::=<NRf>[<units>]`

where:

`<NRf>` is the high level of output amplitude.

`<units>::=[mV | V]`

Returns

`<voltage>`

Examples

`SOURce1:VOLTage:LEVel:IMMediate:HIGH 1V`

Sets the high level of CH1 output amplitude to 1 V.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

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

This command sets or queries the low level of output amplitude for the specified channel. If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the low level of other channel is also the same value.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTag[:LEVel][:IMMediate]:LOW   {<voltage>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTag[:LEVel][:IMMediate]:LOW?
```

Arguments

`<voltage>::=<NRf>[<units>]`

where:

`<NRf>` is the low level of output amplitude.

`<units>::=[mV | V]`

Returns

`<voltage>`

Examples

`SOURce1:VOLTage:LEVel:IMMediate:LOW -1V`

Sets the low level of CH1 output amplitude to -1 V.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

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

This command sets or queries the offset level for the specified channel. If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the offset level of the other channel is also the same value.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:OFFSet {<voltage>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate]:OFFSet?
```

Arguments

`<voltage>::=<NRf>[<units>]`

where:

`<NRf>` is the offset voltage level.

`<units>::=[mV | V]`

Returns

`<voltage>`

Examples

`SOURce1:VOLTage:LEVel:IMMediate:OFFSet 500mV`

Sets the CH1 offset level to 500 mV.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

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

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

Units	Amplitude resolution
V_{PP}	0.1 mVp-p or four digits
V_{RMS}	0.1 mVrms or four digits
DBM	0.1 dBm

You can set the units of output amplitude by using either the unit key selection or the `[SOURce[1|2]]:VOLTage:UNIT` command. The selection by unit key has priority over the remote command.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] {<amplitude>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]
```

Arguments

`<amplitude>::=<NRf>[<units>]`

where:

`<NRf>` is the output amplitude

`<units>::=[VPP | VRMS | DBM]`

Returns

`<amplitude>`

Examples

`SOURce1:VOLTage:LEVel:IMMediate:AMPLitude 1VPP`

Sets the CH1 output amplitude to 1 V_{PP}.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

[SOURce[1|2]]:VOLTage:LIMit:HIGH

This command sets or queries the higher limit of the output amplitude high level for the specified channel. If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the higher level limit of the other channel is the same value.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage:LIMit:HIGH {<voltage>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTage:LIMit:HIGH?
```

Arguments

`<voltage>::=<NRf>[<units>]`

where:

`<NRf>` is the higher limit of output amplitude.

`<units>::=[mV | V]`

Returns

`<voltage>`

Examples

`SOURce1:VOLTage:LIMit:HIGH 1V`

Sets the higher limit of CH1 output amplitude to 1 V.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

[SOURce[1|2]]:VOLTage:LIMit:LOW

This command sets or queries the lower limit of the output amplitude low level for the specified channel. If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the low level lower limit of the other channel is the same value.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage:LIMit:LOW {<voltage>|MINimum|MAXimum}
[SOURce[1|2]]:VOLTage:LIMit:LOW?
```

Arguments

`<voltage>::=<NRf>[<units>]`

where:

`<NRf>` is the lower limit of output amplitude.

`<units>::=[mV | V]`

Returns

`<voltage>`

Examples

`SOURce1:VOLTage:LIMit:LOW 10mV`

Sets the lower limit of CH1 output amplitude to 10 mV.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)

[SOURce[1|2]]:VOLTage:UNIT

This command sets or queries the units of output amplitude for the specified channel. This command does not affect the offset, High level, or Low level of output. The setting of this command is not affected by the units setting of `[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]` command.

$$V_{rms} = \frac{V_{pp}}{2\sqrt{2}} \ x\ (sin)$$

$$dBm=10 \ x \ lg \ \frac{P}{0.001}$$

$$P= \frac{V_{rms}^2}{R_L}$$

$$R_L \text{ load impedance } V_{rms} = \frac{V_{pp}}{2\sqrt{3}} (triangle)$$

If your instrument is a dual-channel model and the `[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` command is set to ON, then the units of the other channel are set the same.

This command:

Group

Source

Syntax

```
[SOURce[1|2]]:VOLTage:UNIT {VPP|VRMS|DBM}
[SOURce[1|2]]:VOLTage:UNIT?
```

Arguments

`VPP`

Sets the units of the output voltage to V_{pp}

`VRMS`

Sets the units of the output voltage to V_{rms}

`DBM`

Sets the units of the output voltage to dBm; you cannot specify DBM if the load impedance is set to infinite

Returns

`VPP|VRMS|DBM`

Examples

`SOURce1:VOLTage:UNIT VPP`

Sets the voltage units to V_{pp}.

Related Commands

`[SOURce[1|2]]:VOLTage:CONCurrent[:STATe]` (on page 145)
`[SOURce[1|2]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]` (on page 149)

SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude]

This command sets or queries the internal noise level which applies to the output signal for the specified channel. The noise level represents the percent against current amplitude level. The setting range is 0 to 50%. This command is available when Run Mode is set to Continuous, Burst, or Sweep. You can set or query whether to add the internal noise to the output signal using the `[SOURce[1|2]]:COMBine:FEED` command.

This command:

Group

Source

Syntax

```
SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude] {<percent>|MINimum|MAXimum}
SOURce<3|4>:POWer[:LEVel][:IMMediate][:AMPLitude]?
```

Arguments

`<percent>::=<NR2>[<units>]`

where:

`<NR2>` is the noise level.

`<units>::=PCT`

Returns

`<percent>`

Examples

`SOURce3:POWer:LEVel:IMMediate:AMPLitude 50PCT`

Sets the internal noise level that is added to the output signal to 50%.

Related Commands

`[SOURce[1|2]]:COMBine:FEED (on page 100)`

*SRE

This command sets and queries the bits in the Service Request Enable Register (SRER).

This command:

Group

Source

Syntax

```
*SRE <bit_value>
*SRE?
```

Arguments

```
<bit_value>::=<NR1>
```

where:

`<NR1>` is a value in the range from 0 through 255; the binary bits of the SRER are set according to this value; using an out-of-range value causes an execution error; the power-on default for SRER is 0 if *PSC is set to 1; if *PSC is set to 0, the SRER maintains the previous power cycle value through the current power cycle.

Returns

```
<bit_value>
```

Examples

```
*SRE 48
```

Sets the bits in the SRER to binary 00110000.

```
*SRE?
```

Might return 32, showing that the bits in the SRER have the binary value of 00100000.

Related Commands

*PSC (on page 63)

STATus:OPERation:CONDition? (query only)

This query-only command returns the contents of the Operation Condition Register.

This command:

Group

Source

Syntax

```
STATus:OPERation:CONDition?
```

Returns

```
<bit_value>::=<NR1>
```

Examples

```
STATus:OPERation:CONDition?
```

Might return 32 which indicates that the OCR contains the binary number 00000000 00100000 and the CH1 of the instrument is waiting for trigger.

STATus:OPERation:ENABle

This command sets or queries the mask for the Operation Enable Register.

This command:

Group

Source

Syntax

```
STATus:OPERation:ENABle <bit_value>
STATus:OPERation:ENABle?
```

Arguments

```
<bit_value>::=<NR1>
```

Returns

```
<bit_value>
```

Examples

```
STATus:OPERation:ENABle 1
```

Sets the CALibrating bit in the OENR to on.

STATus:OPERation[:EVENt]? (query only)

This query-only command returns the value in the Operation Event Register and clears the Operation Event Register.

This command:

Group

Source

Syntax

```
STATus:OPERation[:EVENt]?
```

Returns

```
<NR1>
```

Examples

```
STATus:OPERation?
```

Might return 1 which indicates that the OEVR contains the binary number 00000000 00000001 and the CALibrating bit is set to on.

STATus:PRESet (no query form)

This command presets the SCPI status registers (OENR and QENR).

This command:

Group

Source

Syntax

```
STATus:PRESet
```

Examples

```
STATus:PRESet
```

Presets the SCPI status registers.

STATus:QUEStionable:CONDition? (query only)

This query-only command returns the contents of the Questionable Condition Register.

This command:

Group

Status

Syntax

```
STATus:QUEStionable:CONDition?
```

Returns

```
<bit_value>::=<NR1>
```

Examples

```
STATus:QUEStionable:CONDition?
```

Returns up to 32 which indicates that the QCR contains the binary number 00000000 00100000 and the accuracy of frequency is questionable.

STATus:QUEStionable:ENABle

This command sets or queries the mask for the Questionable Enable Register.

This command:

Group

Status

Syntax

```
STATus:QUEStionable:ENABle <bit_value>
STATus:QUEStionable:ENABle?
```

Arguments

```
<bit_value>::=<NR1>
```

Returns

```
<bit_value>
```

Examples

```
STATus:QUEStionable:ENABle 32
```

Sets the FREQuency bit in the QENR to on.

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

This query-only command returns the value in the Questionable Event Register and clears the Questionable Event Register.

This command:

Group

Status

Syntax

```
STATus:QUEStionable[:EVENt]?
```

Returns

```
<bit_value>::=<NR1>
```

Examples

```
STATus:QUEStionable:EVENt?
```

May return 32 which indicates that the QEVR contains the binary number 00000000 00100000 and the FREQuency bit is set to on.

*STB? (query only)

This query-only command returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit.

This command:

Group

System

Syntax

```
*STB?
```

Examples

```
*STB?
```

Might return 96, showing that the SBR contains the binary value 01100000.

SYSTem:BEEPer:STATe

The `SYSTem:BEEPer:STATe` command sets the beeper ON or OFF.

The `SYSTem:BEEPer:STATe?` command returns “0” (OFF) or “1” (ON).

When the beeper is set to ON, the instrument will beep when an error message or a warning message is displayed on the screen. The instrument does not beep when an error or warning caused by remote command execution.

This command:

Group

System

Syntax

```
SYSTem:BEEPer:STATe {ON|OFF|<NR1>}
SYSTem:BEEPer:STATe?
```

Arguments

`ON or <NR1>≠0`

Enables the beeper.

`OFF or <NR1>=0`

Disables the beeper.

Returns

`<NR1>`

Examples

`SYSTem:BEEPer:STATe ON`

Enables the beeper function

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

This query-only command returns the contents of the Error/Event queue. If the instrument detects an error or an event occurs, the event number and event message will be returned.

This command:

Group

System

Syntax

```
SYSTem:ERRor[:NEXT]?
```

Returns

```
<Error/event number>::=<NR1>
<Error/event description>::=<string>
```

Examples

`SYSTem:ERRor?`

might return the following response:

```
-410,”Query INTERRUPTED”
```

SYSTem:BEEPer[:IMMediate] (no query form)

This command causes the instrument to beep immediately.

This command:

Group

System

Syntax

```
SYSTem:BEEPer[:IMMediate]
```

Arguments

None

Returns

None

Examples

```
SYSTem:BEEPer
```

Causes a beep.

SYSTem:KCLick[:STATe]

This command enables or disables the click sound when you push the front panel buttons or turn the general purpose knob. The query command returns “0” (OFF) or “1” (ON).

This command:

Group

System

Syntax

```
SYSTem:KCLick[:STATe] {ON|OFF|<NR1>}
SYSTem:KCLick[:STATe]?
```

Arguments

`ON or <NR1>≠0`

Enables click sound

`OFF or <NR1>=0`

Disables click sound

Returns

`<NR1>`

Examples

```
SYSTem:KCLick:STATe ON
```

Enables the click sound.

SYSTem:KLOCk[:STATe]

This command locks or unlocks the instrument front panel controls. The query command returns “0” (OFF) or “1” (ON).

This command:

Group

System

Syntax

```
SYSTem:KLOCk[:STATe] {ON|OFF|<NR1>}
SYSTem:KLOCk[:STATe]?
```

Arguments

`ON or <NR1>≠0`

Locks front panel controls.

`OFF or <NR1>=0`

Unlocks front panel controls.

Returns

`<NR1>`

Examples

`SYSTem:KLOCk:STATe ON`

Locks front panel controls.

SYSTem:MACADDress?

This command returns the MAC address of the AFG31000.

Group

System

Syntax

```
SYSTem:MACADDress?
```

Arguments

None.

Returns

```
<MACAddress>
```

Examples

```
SYSTem:MACADDress?
```

Could return a MAC address of `08-00-11-22-d1-60`.

SYSTem:PASSword:CDISable (no query form)

This command disables protected commands. The instrument security protection is activated. In the AFG31000 Series instruments, no remote commands are under the control of SYSTem:PASSword commands.

This command:

Group

System

Syntax

```
SYSTem:PASSword:CDISable <password>
```

Arguments

```
<password>::=<string>
```

Specifies current password; the string is case sensitive.

Returns

None

Examples

```
SYSTem:PASSword:CDISable "password"
```

Activates the security protection.

Related Commands

SYSTem:PASSword[:CENable] (on page 164)
SYSTem:PASSword[:CENable]:STATe? (on page 165)
SYSTem:PASSword:NEW (on page 166)

SYSTem:PASSword[:CENable] (no query form)

This command enables protected commands to function. The instrument security protection is deactivated. No remote commands are under the control of SYSTem:PASSword commands.

This command:

Group

System

Syntax

```
SYSTem:PASSword[:CENable] <password>
```

Arguments

```
<password>::=<string>
```

Specifies current password; the string is case sensitive.

Examples

```
SYSTem:PASSword:CENable "password"
```

Deactivates the security protection.

Related Commands

```
SYSTem:PASSword:CDISable (on page 162)
SYSTem:PASSword[:CENable]:STATe? (on page 165)
SYSTem:PASSword:NEW (on page 166)
```

SYSTem:PASSword[:CENable]:STATe? (query only)

This query-only command returns the security protection state.

This command:

Group

System

Syntax

```
SYSTem:PASSword[:CENable]:STATe?
```

Returns

`<NR1>`

where:

`<NR1>=0` indicates that the security protection is in the on state.

`<NR1>≠0` indicates that the security protection is in the off state.

Examples

```
SYSTem:PASSword:STATe?
```

Might return 0, indicating that the instrument security protection is on.

Related Commands

SYSTem:PASSword:CDISable (on page 162)
SYSTem:PASSword[:CENable]:STATe? (on page 165)
SYSTem:PASSword:NEW (on page 166)

SYSTem:PASSword:NEW (no query form)

This command changes the password.

This command:

Group

System

Syntax

```
SYSTem:PASSword:NEW <current_password>,<new_password>
```

Arguments

`<current_password>::=<string>`

Specifies current password.

`<new_password>::=<string>`

Specifies a new password.

Password strings are case sensitive; a password must have at least four characters, and not more than 12 characters.

Examples

```
SYSTem:PASSword:NEW “DEFAULT”,“abc123”
```

Changes the current password DEFAULT to abc123.

Related Commands

`SYSTem:PASSword:CDISable (on page 162)`
`SYSTem:PASSword[:CENable] (on page 164)`
`SYSTem:PASSword[:CENable]:STATe? (on page 165)`

SYSTem:RESTart

This command restarts the instrument. This command is equivalent to pushing the power switch on the front panel.

This command:

Group

System

Syntax

```
SYSTem:RESTart
```

Arguments

None

Examples

```
SYSTem:RESTart
```

Restarts the AFG31000 Series.

SYSTem:SECurity:IMMediate (no query form)

This command erases all the current instrument setups, setup memory, last setup memory, user waveform memory, and log content, and recalls the factory default settings. Calibration data is not erased.

The communication settings are initialized to the factory default settings. This might cause a remote communication error.

This command:

Group

System

Syntax

```
SYSTem:SECurity:IMMediate
```

Examples

```
SYSTem:SECurity:IMMediate
```

Initializes the instrument.

SYSTem:ULANguage

This command sets or queries the language that the instrument uses to display information on the screen.

This command:

Group

System

Syntax

```
SYSTem:ULANguage
{ENGLish|FRENch|GERMan|JAPanese|KORean|SCHinese|TCHinese|RUSSian}
SYSTem:ULANguage?
```

Arguments

```
ENGLish|FRENch|GERMan|JAPanese|KORean|SCHinese|TCHinese|RUSSian
```

Specifies which language will be used to display instrument information on the screen.

Returns

```
ENGL|FREN|GERM|JAP|KOR|SCH|TCH|RUSS
```

Examples

```
SYSTem:ULANguage FRENch
```

Specifies that the instrument displays information in French.

SYSTem:VERSion? (query only)

This query-only command returns the conformed SCPI version of the instrument.

This command:

Group

System

Syntax

```
SYSTem:VERSion?
```

Returns

```
<SCPI Version>::=YYYY.V
```

where:

`YYYY` – indicates year

`V` – indicates the version number for that year.

Examples

```
SYSTem:VERSion?
```

May return 1999.0.

TRACe|DATA[:DATA]

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

This command:

Group

Trace

Syntax

```
TRACe|DATA[:DATA] {EMEMory[1]|EMEMory2},<binary_block_data>
TRACe|DATA[:DATA]? {EMEMory[1]|EMEMory2}
```

Arguments

```
<binary_block_data> where
<binary_block_data>
```

Is the waveform data in binary format.

Returns

```
<binary_block_data>
```

Examples

```
DATA:DATA EMEMory1,#42000<DAB><DAB>...<DAB>
```

Transmits a waveform to the edit memory1 in the AFG; 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.

*TRG (no query form)

This command generates a trigger event.

This command:

Group

Trigger

Syntax

```
*TRG
```

Examples

```
*TRG
```

Generates a trigger event.

Related Commands

TRIGger[:SEQuence][:IMMediate] (on page 171)

TRIGger[:SEQuence]:SLOPe

This command sets or queries the slope of trigger signal.

This command:

Group

Trigger

Syntax

```
TRIGger[:SEQuence]:SLOPe {POSitive|NEGative}
TRIGger[:SEQuence]:SLOPe?
```

Arguments

`POSitive`

Indicates that the event occurs on the rising edge of the external trigger signal.

`NEGative`

Indicates that the event occurs on the falling edge of the external trigger signal.

Returns

```
POS|NEG
```

Examples

```
TRIGger:SLOPe POSitive
```

Sets the trigger slope to positive, which triggers on the rising edge of the signal.

TRIGger[:SEQuence]:SOURce

This command sets or queries the trigger source for an external trigger signal.

This command:

Group

Trigger

Syntax

```
TRIGger[:SEQuence]:SOURce {TIMer|EXTernal}
TRIGger[:SEQuence]:SOURce?
```

Arguments

`TIMer`

Specifies an internal clock as the trigger source.

Returns

`TIM|EXT`

Examples

`TRIGger:SEQuence:SOURce EXTernal`

Sets an external trigger input as the trigger source.

TRIGger[:SEQuence]:TIMer

This command sets or queries the period of an internal clock when you select the internal clock as the trigger source with the `TRIGger[:SEQuence]:SOURce` command. The setting range is 1 μs to 500.0 s.

This command:

Group

Trigger

Syntax

```
TRIGger[:SEQuence]:TIMer <seconds>
TRIGger[:SEQuence]:TIMer
```

Arguments

`<seconds>::=<NRf>[<units>]`

where:

```
    <units>::=[us | ms | s]
```

Returns

`<seconds>`

Examples

`TRIGger:TIMer 5ms`

Sets the internal trigger rate to 5 ms.

TRIGger[:SEQuence][:IMMediate] (no query form)

This command forces a trigger event to occur.

This command:

Group

Trigger

Syntax

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

Examples

```
TRIGger:SEQuence:IMMediate
```

Generates a trigger event.

*TST? (query only)

This command performs a self-test and returns the results.

NOTE. The self-test can take several minutes to complete. During this time, the AFG does not execute any commands. Do not power off the instrument during the self-test.

This command:

Group

Calibration and Diagnostic

Syntax

```
*TST?
```

Returns

`<NR1>`

where:

`<NR1>`=0 indicates that the self-test completed without errors.

`<NR1>`≠0 indicates that the AFG detected an error.

Examples

```
*TST?
```

Performs a self-test and returns 0 if no error is reported.

Related Commands

`DIAGnostic[:ALL]` (on page 31)

*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.

This command:

Group

Synchronization

Syntax

```
*WAI
```

Examples

```
*WAI
```

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

Related Commands

`*OPC (on page 58)`

WLISt:NAME? (query only)

This command returns a waveform name from the waveform list in the position specified by the index value.

This command:

Group

Waveform

Syntax

```
WLISt:NAME? <index>
```

Arguments

```
<index>::=<NR1>
```

Ranges from 1 to 256.The index should be under the size of waveform list.

Returns

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

Examples

```
WLISt:NAME? 21
```

May return "M:/untitled21.tfwx".

Related Commands

`WLISt:SIZE? (query only) (on page 173)`

WLISt:SIZE? (query only)

This command returns the number of waveforms in the waveform list.

This command:

Group

Waveform

Syntax

```
WLISt:SIZE?
```

Arguments

None

Returns

```
<NR1>
```

Ranges from 0 to 256.

Examples

```
WLISt:SIZE?
```

May return 2.

WLISt:WAVeform:DELete (no query form)

This command deletes the waveform from the waveform list. If the deleted waveform is currently loaded into waveform memory and sequence table, it is unloaded.

NOTE. When ALL is specified, all 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.

This command:

Group

Waveform

Syntax

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

Arguments

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

Indicates that the waveform path and name.

M:/ indicates the internal memory

U:/ indicates the external USB memory

P:/ indicates the internal predefine waveforms

ALL. Indicates all the waveforms in the list

Examples

```
WLISt:WAVeform:DELete ALL
```

Deletes all user-defined waveforms from the currently loaded setup.

```
WLISt:WAVeform:DELete "M:/Test1.tfwx"
```

Deletes a waveform called `"Test1.tfwx"` in internal memory.

Related Commands

`WLISt:SIZE? (query only)` (on page 173)

WLISt:WAVeform:IMPort (no query form)

This command imports the waveform from internal driver or USB driver into the waveform list.

This command:

Group

Waveform

Syntax

```
WLISt:WAVeform:IMPort {<wfm_name>}
```

Arguments

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

Indicates that the waveform path and name.

M:/ indicates the internal memory.

U:/ indicates the external USB memory.

P:/ indicates the internal predefine waveforms.

Examples

```
WLISt:WAVeform:IMPort “M:/Test1.tfwx"
```

Imports a waveform called `"Test1.tfwx"` in internal memory into the waveform list.

Related Commands

`WLISt:SIZE? (query only)` (on page 173)
`WLISt:WAVeform:DELete (no query form)` (on page 174)

WLISt:WAVeform:LENGth? (query only)

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

This command:

Group

Waveform

Syntax

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

<wfm_name> is not optional.

Arguments

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

Indicates that the waveform path and name.

M:/ indicates the internal memory.

U:/ indicates the external USB memory.

P:/ indicates the internal predefine waveforms.

Returns

```
<NR1>
```

Examples

```
WLISt:WAVeform:LENGth? "M:/Sine360.tfwx"
```

Returns the data points of waveform “`Sine360.tfwx`”.

Status and events

Status and events

This section provides details about the status information and events the arbitrary function generator reports.

Status reporting structure

The arbitrary function generator status reporting functions conform to IEEE-488.2 and SCPI standards. Use the status reporting function to check for instrument errors and to identify the types of events that have occurred on the instrument.

The error and event reporting system consists of the following three blocks:

- Standard/Event Status
- Operation Status
- Questionable Status

The operations processed in these blocks are summarized in status bytes, which provide the error and event data (see next figure).

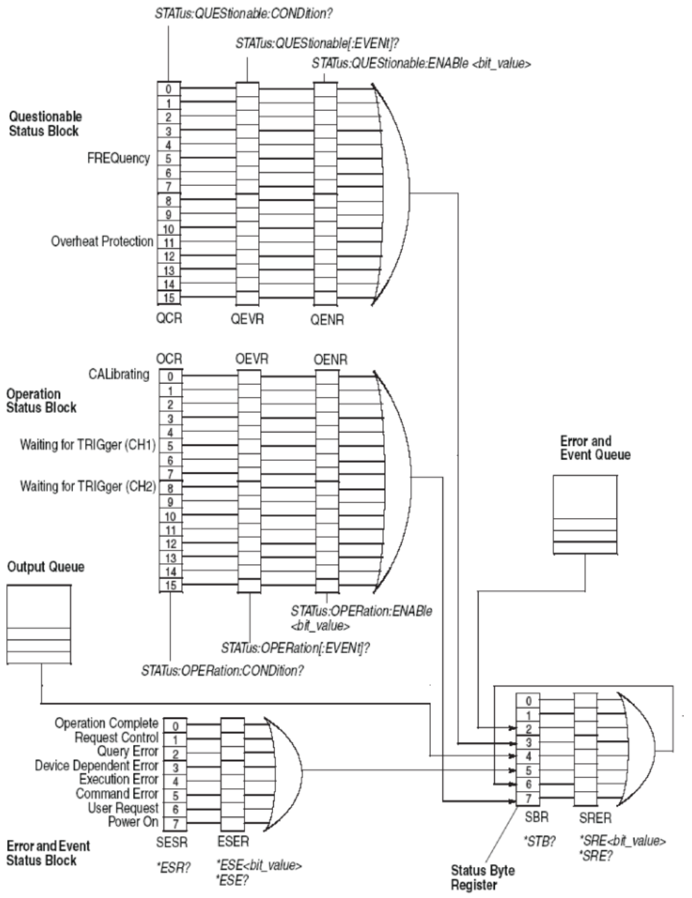


Figure 10: Error and event handling process

Standard and event status block

This block is used to report power on/off, command error, and command execution status.

The block has two registers: the Standard Event Status Register (SESR) and the Event Status Enable Register (ESER).

Standard Event Status Register	The SESR is an eight-bit status register. When an error or other type of event occurs on the instrument, the corresponding bit is set. You cannot write to this register.
Event Status Enable Register	The ESER is an eight-bit enable register that masks the SESR. You can set this mask, and take AND with the SESR to determine whether or not the ESB bit in the Status Byte Register (SBR) should be set.

Operation status block

This block is used to report on the status of several operations being executed by the arbitrary function generator.

The block has three registers: the Operation Condition Register (OCR), the Operation Event Register (OEVR), and the Operation Enable Register (OENR).

Operation Condition Register	When the instrument achieves a certain status, the corresponding bit is set to the OCR. It is not allowed for the user to write to this register.
Operation Event Register	The OCR bits that have changed from false (reset) to true (set) status are set in the OEVR.
Operation Enable Register	The function of the OENR is to mask the OEVR. You can set this mask and take AND with the OEVR to determine whether or not the OSS bit in the Status Byte Register (SBR) should be set.

Questionable status block

This block reports on the status of signals and data, such as the accuracy of entered data and signals generated by the instrument. The register configuration and process flow are the same as the Questionable Status Block.

Registers

The registers in the event reporting system fall into two functional groups:

- The Status Registers contain information about the status of the instrument.
- Enable Registers determine whether selected types of events are reported to the Status Registers and the Event Queue.

Status registers

There are six types of status registers:

- Status Byte Register (SBR)
- Standard Event Status Register (SESR)
- Operation Condition Register (OCR)
- Operation Event Register (OEVR)
- Questionable Condition Register (QCR)
- Questionable Event Register (QEVR)

Status Byte Register (SBR)	The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2-1992 (see). These bits are used to monitor the output queue, SESR, and service requests, respectively.

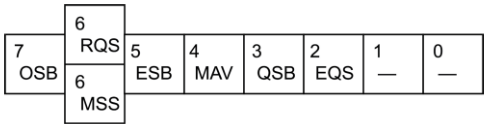


Figure 11: Status Byte Register (SBR)

SBR bit functions

Bit	Function	Description
7 (MSB)	OSB	Operation Status Bit. Indicates that an operation event has occurred.
6	RQS	Request Service. When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (in other words, that the GPIB bus SRQ line is LOW). The RQS bit is cleared when serial poll ends.
6	MSS	Master Status Summary. When the instrument is accessed using the *STB? query, this bit is called the Master Status Summary (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query.
5	ESB	Event Status Bit. This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed.
4	MAV	Message Available Bit. This bit indicates that a message has been placed in the output queue and can be retrieved.
3	QSB	Questionable Status Bit.
2	EQS	Error/Event Queue Summary.
1 - 0	——	Not used

Standard Event Status Register (SESR)	The SESR records eight types of events that can occur within the instrument as shown in the next figure.

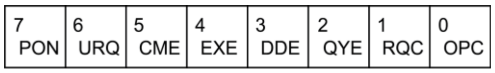


Figure 12: Standard Event Status Register

SESR bit functions

Bit	Function	Description
7 (MSB)	PON	Power On. Indicates that the power to the instrument is on.
6	URQ	User Request. Indicates that an application event has occurred. The arbitrary function generator does not use this bit.
5	CME	Command Error. Indicates that an error occurred while the arbitrary function generator was parsing a command or query.
4	EXE	Execution Error. Indicates that an error occurred while the arbitrary function generator was executing a command or query. Execution errors occur for one of the following reasons:
		- a value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument
		- the command was not executed properly because the conditions for execution is differed from those required.
3	DDE	Device Error. An instrument error has been detected.
2	QYE	Query Error. Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons:
		- an attempt was made to retrieve messages from the output queue when the output queue is empty or in pending status
		- the output queue message was cleared while it was being retrieved from the output queue
1	RQC	Request Control. The arbitrary function generator does not use this bit.
0	OPC	Operation Complete. Indicates that the operation is complete. This bit is set when all pending operations complete following the *OPC command.

Operation Event Register (OEVR)	This register has the same content as the Operation Condition Register.
Operation Condition Register (OCR)	The Operation Condition Register is made up of sixteen bits, which note the occurrence of events as shown in the next figure.

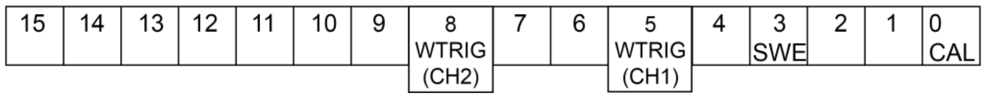


Figure 13: Operation Condition Register (OCR)

OCR bit functions

Bit	Function	Description
15 to 9	——	Not used.
8	WTRIG CH2	Waiting for Trigger. Indicates whether the instrument is waiting for a trigger. This bit is set when CH 2 (in the case of dual-channel model) is waiting for a trigger. Bit is reset when the waiting-for-trigger status is canceled.
5	WTRIG CH1	Waiting for Trigger. Indicates whether the instrument is waiting for a trigger. This bit is set when CH 1 (in the case of dual-channel model) is waiting for a trigger. Bit is reset when the waiting-for-trigger status is canceled.
4	——	Not used.
3	SWE	Sweep. Indicates whether the instrument is executing a frequency sweep. This bit is set when a frequency sweep is being executed on CH 1 or another channel (in the case of dual-channel model). Bit is reset when the execution stops.
2 to 1	——	Not used.
0	CAL	Calibration. Indicates whether the instrument is being calibrated. This bit is set when calibration is in progress and is reset when the calibration is complete.

Questionable Event Register (QEVR).	This register has the same content as the Questionable Condition Register.
Questionable Condition Register (QCR).	The Questionable Condition Register is made up of sixteen bits which note the occurrence of two types of events.

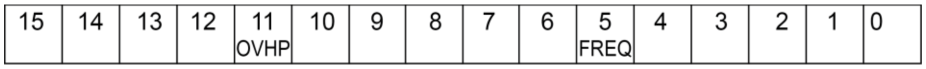


Figure 14: Questionable Enable Register (QENR)

QCR bit functions

Bit	Function	Description
15 to 12	——	Not used.
11	OVHP	Overheat protection. Indicates whether the instrument internal temperature is in questionable condition.
10 to 6	——	Not used.
5	FREQ	Frequency. Indicates whether frequency accuracy of the signal is of questionable quality.
4 to 0	——	Not used.

Enable registers

There are four types of enable registers:

- Event Status Enable Register (ESER),
- Service Request Enable Register (SRER),
- Operation Enable Register (OENR),
- Questionable Enable Register (QENR),

Each bit in the enable registers corresponds to a bit in the controlling status register. By setting and resetting the bits in the enable register, you can determine whether or not events that occur will be registered to the status register and queue.

Event Status Enable Register (ESER)

The ESER consists of bits defined exactly the same as bits 0 through 7 in the SESR register. You can use this register to control whether or not the Event Status Bit (ESB) in the SBR should be set when an event has occurred, and to determine if the corresponding SESR bit is set.

To set the ESB in the SBR (when the SESR bit has been set), set the ESER bit corresponding to that event. To prevent the ESB from being set, reset the ESER bit corresponding to that event.

Use the *ESC command to set the bits in the ESER. Use the *ESR? query to read the contents of the ESER. The next figure shows the ESER functions.

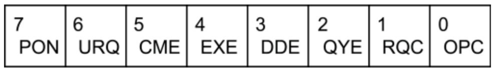


Figure 15: Standard Event Status Register

Service Request Enable Register (SRER).

The SRER consists of bits defined exactly the same as bits 0 through 7 in the SBR. You can use this register to define which events will generate service requests.

The SRER bit 6 cannot be set. Also, the RQS is not maskable.

The generation of a service request with the GPIB interface involves changing the SRQ line to LOW, and making a service request to the controller. The result is that a status byte for which an RQS has been set is returned in response to serial polling by the controller.

Use the *SRE command to set the bits of the SRER. Use the *SRE? query to read the contents of the SRER. Bit 6 must be set to 0. The next figure shows the SRER functions.

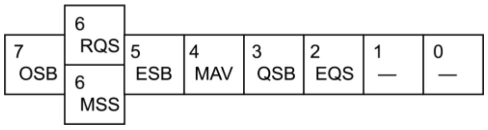


Figure 16: Status Byte Register (SBR)

Operation Enable Register (OENR).

The OENR consists of bits defined exactly the same as bits 0 through 15 in the OEVR (see Operation Event Register (OEVR)). You can use this register to control whether or not the Operation Status Bit (OSB) in the SBR is set when an event occurs and the corresponding OEVR bit is set.

Use the `STATus:OPERation:ENABle` command to set the bits in the OENR. Use the `STATus:OPERation:ENABle?` query to read the contents of the OENR.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							WTRIG (CH2)			WTRIG (CH1)		SWE			CAL

Figure 17: Operation Condition Register (OCR)

Questionable Enable Register (QENR).

The QENR consists of bits defined exactly the same as bits 0 through 15 in the QEVR register. You can use this register to control whether the QSB in the SBR is set when an event occurs and the corresponding QEVR bit is set.

Use the STATus:QUEStionable:ENABle command to set the bits in the QENR.

Use the STATus:QUEStionable:ENABle? query to read the contents of the QENR.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				OVHP						FREQ					

Figure 18: Questionable Enable Register (QENR)

Queues

There are two types of queues in the status reporting system:

- Output
- Error or event

Output queue

The output queue is an FIFO (first-in, first-out) queue that holds response messages to queries awaiting retrieval. When there are messages in the queue, the MAV bit in the SBR is set.

The output queue is emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error occurs and the output queue is emptied; however, the operation proceeds even if an error occurs.

Error or event queue

The event queue is an FIFO queue, which stores events as they occur in the instrument. If more than 64 events are stored, the 64th event is replaced with event code -350 (“Queue Overflow”).

The oldest error code and text are retrieved by using one of the following queries:

```
SYSTem:ERRor[:NEXT]?
```

First, issue the *ESR? query to read the contents of the SESR. The contents of the SESR are cleared after they are read. If an SESR bit is set, events are stacked in the Error/Event Queue. Retrieve the event code with the following command sequence:

```
*ESR?
SYSTem:ERRor[:NEXT]?
```

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

Messages and codes

Error and event codes with negative values are SCPI standard codes. Error and event codes with positive values are unique to the AFG31000 Series Arbitrary Function Generator.

The next table lists event code definitions. When an error occurs, you can find its error class by checking for the code in the following tables. Events in these tables are organized by event class.

Definition of event codes

Event class	Code range	Description
No error	0	No event or status
Command errors	-100 to -199	Command syntax errors
Execution errors	-200 to -299	Command execution errors
Device-specific errors	-300 to -399	Internal device errors
Query errors	-400 to -499	System event and query errors
Power-on events	-500 to -599	Power-on events
User request events	-600 to -699	User request events
Request control events	-700 to -799	Request control events
Operation complete events	-800 to -899	Operation complete events
Extended device-specific errors	1 to 32767	Device dependent device errors
Reserved	other than above	not used

Command errors

The next 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.

Command error messages

Error code	Error message
-100	Command error
-101	Invalid character
-102	Syntax error
-103	Invalid separator
-104	Data type error
-105	GET not allowed
-108	Parameter not allowed
-109	Missing parameter
-110	Command header error
-111	Header separator error
-112	Program mnemonic too long
-113	Undefined header
-114	Header suffix out of range
-115	Unexpected number of parameters
-120	Numeric data error
-121	Invalid character in number
-123	Exponent too large
-124	Too many digits
-128	Numeric data not allowed
-130	Suffix error
-131	Invalid suffix
-134	Suffix too long
-138	Suffix not allowed
-140	Character data error
-141	Invalid character data
-144	Character data too long
-148	Character data not allowed
-150	String data error
-151	Invalid string data
-158	String data not allowed
-160	Block data error
-161	Invalid block data
-168	Block data not allowed
-170	Expression error
-171	Invalid expression
-178	Expression data not allowed
-180	Macro error
-181	Invalid outside macro definition
-183	Invalid inside macro definition
-184	Macro parameter error

Execution error messages

The next table lists the errors that are detected during execution of a command.

Error code	Error message
-200	Execution error
-201	Invalid while in local
-202	Settings lost due to RTL
-203	Command protected
-210	Trigger error
-211	Trigger ignored
-212	Arm ignored
-213	Init ignored
-214	Trigger deadlock
-215	Arm deadlock
-220	Parameter error
-221	Settings conflict
-222	Data out of range
-223	Too much data
-224	Illegal parameter value
-225	Out of memory
-226	Lists not same length
-230	Data corrupt or stale
-231	Data questionable
-232	Invalid format
-233	Invalid version
-240	Hardware error
-241	Hardware missing
-250	Mass storage error
-251	Missing mass storage
-252	Missing media
-253	Corrupt media
-254	Media full
-255	Directory full
-256	File name not found
-257	File name error
-258	Media protected
-260	Expression error
-261	Math error in expression
-270	Macro error
-271	Macro syntax error
-272	Macro execution error
-273	Illegal macro label
-274	Macro parameter error
-275	Macro definition too long
-276	Macro recursion error
-277	Macro redefinition not allowed

Error code	Error message
-278	Macro header not found
-280	Program error
-281	Cannot create program
-282	Illegal program name
-283	Illegal variable name
-284	Program currently running
-285	Program syntax error
-286	Program runtime error
-290	Memory use error
-291	Out of memory
-292	Referenced name does not exist
-293	Referenced name already exists
-294	Incompatible type

Device specific errors

The next table lists the device-specific errors that can occur during arbitrary function generator operation. These errors may indicate that the instrument needs repair.

Device specific error messages

Error code	Message
-300	Device specific error
-310	System error
-311	Memory error
-312	PUD memory lost
-313	Calibration memory lost
-314	Save/recall memory lost
-315	Configuration memory lost
-320	Storage fault
-321	Out of memory
-330	Self-test failed
-340	Calibration failed
-350	Queue overflow
-360	Communication error
-361	Parity error in program message
-362	Framing error in program message
-363	Input buffer overrun
-365	Time out error

Query Errors

The next table lists the error codes that are returned in response to an unanswered query.

Query error messages

Error code	Message
-400	query error
-410	query INTERRUPTED
-420	query UNTERMINATED
-430	query DEADLOCKED
-440	query UNTERMINATED after indefinite response

Power on event

This event occurs when the instrument detects an OFF to ON transition in its power supply.

Power on event message

Event code	Message
-500	Power on

User request event

This event is not used in this instrument.

User request event

Event code	Message
-600	User request

Request control event

This event is not used in this instrument.

Request control event

Event code	Message
-700	Request control

Operation complete events

This event occurs when the instrument’s synchronization protocol, having been enabled by an `*OPC` command, completes all selected pending operations.

Operation complete event

Event code	Message
-800	Operation complete

Device errors

The next table lists the error codes that are unique to the AFG31000 Series Arbitrary Function Generator.

Error code	Error message
1101	Calibration failed; CH1 Internal offset
1102	Calibration failed; CH2 Internal offset
1103	Calibration failed; CH1 Output offset
1104	Calibration failed; CH2 Output offset
1105	Calibration failed; CH1 Output gain
1106	Calibration failed; CH2 Output gain
1201	Calibration failed; CH1 x 12 dB K12 attenuator
1202	Calibration failed; CH2 x 12 dB K22 attenuator
1203	Calibration failed; CH1 x 20 dB K13 attenuator
1204	Calibration failed; CH2 x 20 dB K23 attenuator
1205	Calibration failed; CH1 x 20 dB K14 attenuator
1206	Calibration failed; CH2 x 20 dB K24 attenuator
1207	Calibration failed; CH1 x 14/20 dB K15 attenuator (14db for AFG3125X, AFG3115X; 20dB for AFG3110X, AFG3105X, AFG3102X)
1208	Calibration failed; CH2 x 14/20 dB K25 attenuator (14db for AFG3125X, AFG3115X; 20dB for AFG3110X, AFG3105X, AFG3102X)
1209	Calibration failed; CH1 x 20 dB K18 attenuator
1210	Calibration failed; CH2 x 20 dB K28 attenuator
1211	Calibration failed; CH1 Filter
1212	Calibration failed; CH2 Filter
1215	Calibration failed; U51
1216	Calibration failed; Saving calibration file failure
2101	Self-test failed; Calibration data checksum
2301	Self-test failed; CH1 Internal offset
2302	Self-test failed; CH2 Internal offset
2303	Self-test failed; CH1 Output offset
2304	Self-test failed; CH2 Output offset
2305	Self-test failed; CH1 Output gain
2306	Self-test failed; CH2 Output gain
2401	Self-test failed; CH1 x 12 dB K12 attenuator
2402	Self-test failed; CH2 x 12 dB K22 attenuator
2403	Self-test failed; CH1 x 20 dB K13 attenuator
2404	Self-test failed; CH2 x 20 dB K23 attenuator
2405	Self-test failed; CH1 x 20 dB K14 attenuator
2406	Self-test failed; CH2 x 20 dB K24 attenuator
2407	Self-test failed; CH1 x 14/20 dB K15 attenuator (14db for AFG3125X, AFG3115X; 20dB for AFG3110X, AFG3105X, AFG3102X)
2408	Self-test failed; CH2 x 14/20 dB K25 attenuator (14db for AFG3125X, AFG3115X; 20dB for AFG3110X, AFG3105X, AFG3102X)

Error code	Error message
2409	Self-test failed; CH1 x 20 dB K18 attenuator
2410	Self-test failed; CH2 x 20 dB K28 attenuator
2411	Self-test failed; CH1 Filter
2412	Self-test failed; CH2 Filter
2415	Self-test failed; U51
9112	Waveform error; invalid waveform length
9113	Waveform error; waveform length is too short

InstaView calibration events

This event occurs when the instrument does InstaView calibration.

Error code	Error message
3001	Calibration failed; CH1 x 4 attenuator for offset_DAC
3002	Calibration failed; CH2 x 4 attenuator for offset_DAC
3003	Calibration failed; CH1 x 16 attenuator for offset_DAC
3004	Calibration failed; CH2 x 16 attenuator for offset_DAC
3005	Calibration failed; CH1 x 4 attenuator for variable gain
3006	Calibration failed; CH2 x 4 attenuator for variable gain
3007	Calibration failed; CH1 x 16 attenuator for variable gain
3008	Calibration failed; CH2 x 16 attenuator for variable gain
3009	Calibration failed; CH1 x 4 attenuator for attenuation circuit
3010	Calibration failed; CH2 x 4 attenuator for attenuation circuit
3011	Calibration failed; CH1 x 16 attenuator for attenuation circuit
3012	Calibration failed; CH2 x 16 attenuator for attenuation circuit

License command errors

This event occurs when the instrument encounters an error with a license command.

Error code	Error message
3120	"Error loading keys"
3121	"Error loading option"
3122	"Error loading option key database"
3123	"Error loading license file"
3124	"Error updating license file"
3125	"Signature verification failed"
3126	"Memory allocation failed"
3127	"Parsing failed"
3128	"HostID verification failed"
3129	"AppId verification failed"
3130	"LicenseId verification failed"
3131	"Past expiration date"
3132	"License ID already answered"
3133	"Clock tampering detected"
3134	"Strongbox tampering detected"
3135	"Illegal Parameter error"
3136	"Error loading exit key file"
3137	"Feature does not exist"
3138	"No licenses found"

Programming examples

Programming examples

The following two example programs demonstrate methods that you can use to control the arbitrary function generator through the general purpose interface bus (GPIB).

- Example 1: Set up a Waveform Output
- Example 2: Waveform Transfer and Copy

The example programs are written in Microsoft Visual Basic Version 6.0. The programs run on Windows PC compatible systems equipped with TekVISA and a National Instruments GPIB board with the associated drivers.

TekVISA is the Tektronix implementation of the VISA application programming interface (API). TekVISA is industry-compliant software for writing interoperable instrument drivers in a variety of application development environments (ADEs).

The example programs assume that the GPIB system recognizes the PC (external controller) as GPIB0, and the address number of the instrument as 11.

If you use an interface other than GPIB, change the resource name of source code. Refer to the **TekVISA Programmer Manual** for details about resource on tek.com.

Example 1

This is a sample program for setting the arbitrary function generator outputs.

```
Private Sub Sample1_Click()
'
'Assign resource
'
Tvc1.Descriptor = "GPIB0::11::INSTR"
'
'Initialize of device setting
'
Tvc1.WriteString ("*RST")
'
'Set CH1 output parameters
'
Tvc1.WriteString ("FUNCTION SIN") 'Set output waveform SIN
Tvc1.WriteString ("FREQUENCY 10E3") 'Set frequency 10kHz
Tvc1.WriteString ("VOLTAGE:AMPLITUDE 2.00") 'Set amplitude 2Vpp
Tvc1.WriteString ("VOLTAGE:OFFSET 1.00") 'Set offset 1V
Tvc1.WriteString ("PHASE:ADJUST 0DEG") 'Set phase 0degree
'
'Set CH2 output parameters
'
Tvc1.WriteString ("SOURCE2:FUNCTION SIN") 'Set output
waveform SIN
Tvc1.WriteString ("SOURCE2:FREQUENCY 10E3") 'Set frequency
10kHz \
Tvc1.WriteString ("SOURCE2:VOLTAGE:AMPLITUDE 1.00") 'Set
amplitude 1Vpp
Tvc1.WriteString ("SOURCE2:VOLTAGE:OFFSET 0.00") 'Set offset
0V
Tvc1.WriteString ("SOURCE2:PHASE:ADJUST 90DEG") 'Set phase
90degrees
'
'Save settings and output on
'
Tvc1.WriteString ("*SAV 1") 'Save settings to Setup1
Tvc1.WriteString ("*RCL 1") 'Recall settings from Setup1
'
End Sub
```

Example 2

This is a sample program for sending an arbitrary waveform to the arbitrary function generator's Edit Memory1 and copying the contents of Edit Memory1 to the user waveform memory.

```
    Private Sub SendWave()
        'Assign resource
        Const ResourceName As String = "TCPIP0::172.17.108.175::INSTR"
        Dim rm As GlobalResourceManager
        Dim Tvc1 As TcpipSession
        Tvc1 = rm.Open(ResourceName)

        'Initialize device
        Tvc1.FormattedIO.WriteLine("*IDN?")
        Tvc1.FormattedIO.WriteLine("*RST")



        Dim num_points = 3000
        Dim base = 9000
        Dim offset = 6000
        Dim SinePoint As Integer

        Dim SineWaveData(num_points) As Byte

        For i = 0 To num_points
            'Build sine wave
            SinePoint = Int(base + offset * Math.Sin(i / 120))
            SineWaveData(i) = SinePoint / 256
        Next i


        'Transfer waveform
        'Transfer arbitrary block data to edit memory
        Tvc1.SendEndEnabled = False
        Tvc1.FormattedIO.WriteLine("TRACE:DATA EMEM1, #43000")
        Tvc1.SendEndEnabled = True
        Tvc1.RawIO.Write(SineWaveData)
        'Set CH1 output parameters
        Tvc1.FormattedIO.WriteLine("FUNCTION EMEM1") 'Set output waveform EMEM1
        Tvc1.FormattedIO.WriteLine("FREQUENCY 8K") 'Set frequency 8kHz
        Tvc1.FormattedIO.WriteLine("OUTPUT ON") 'Set CH1 output on
    End Sub
```

Default settings

The following table shows the default settings for the instrument. These settings are restored when the default button on the front panel is pressed.

Output configuration settings	**Default setting**
Function	Sine
Frequency	1.000 000 000 00 MHz
Phase(except Pulse)	0.00°
Amplitude	1.000 Vp-p
Offset	0 mV
Symmetry (Ramp)	0.5
Duty (Pulse)	0.5
Leading edge(Pulse)	AFG31021/31022: 8.0ns AFG31051/31052: 6.0ns AFG31101/31102: 4.0ns AFG31151/31152: 3.0ns AFG31251/31252: 2.0ns
Trailing edge(Pulse)	AFG31021/31022: 8.0ns AFG31051/31052: 6.0ns AFG31101/31102: 4.0ns AFG31151/31152: 3.0ns AFG31251/31252: 2.0ns
Delay (Pulse)	0.00ns
Output units	Vpp

Setting bar configuration settings	**Default setting**
Output invert	Off
External add	Off
High limit	AFG31021/31022: 5.000V AFG31051/AFG31052: 5.000V AFG31101/31102: 5.000V AFG31151/31152: 2.500V AFG31251/31252: 2.500V
Low limit	AFG31021/31022: -5.000V AFG31051/AFG31052: -5.000V AFG31101/31102: -5.000V AFG31151/31152: -2.500V AFG31251/31252: -2.500V
Output impedance	50 Ω
Output noise add	Off
Level CH1=CH2	Off
Frequency CH1=CH2	Off
Phase CH1=CH2	Off

Run mode configuration settings	Default setting
Run Mode	Continuous

Modulation configuration settings	Default setting
Modulation type	AM
Modulation source	Internal
Modulation waveform (except FSK,PSK)	Sine
Modulation waveform (FSK, PSK)	Square
Modulation frequency	10.000 000 0kHz
AM depth	0.5
Modulation type	AM
Modulation source	Internal
Modulation waveform (except FSK,PSK)	Sine
Modulation waveform (FSK, PSK)	Square
Modulation frequency	10.000 000 0kHz
AM depth	0.5
FM deviation	1.000 000 00MHz
PM deviation	90.00°
FSK hop frequency	1.000 000 00MHz
FSK rate	10.000 000 0kHz
PSK hop phase	90.00°
PWM deviation	0.05

Sweep configuration settings	Default setting
Sweep start frequency	100.000 000 000kHz
Sweep stop frequency	1.000 000 000 000MHz
Sweep time	10.000 00ms
Sweep hold time	0.00us
Sweep return time	1.000 00ms
Sweep type	Linear
Sweep mode	Repeat
Sweep source	Internal
Trigger slope	Positive
Trigger interval	1.000 000 000ms

Burst configuration settings	Default setting
Burst mode	Triggered(N-Cycle)
Burst count	5
Burst idle state	START
Trigger source	Internal
Trigger slope	Positive
Trigger interval	1.000 000 000ms
Trigger delay	0.0 ns

System configuration settings	Default setting
Clock reference	Internal
Trigger out	Trigger

Advanced Mode configuration settings	Default setting
Run mode	Sequence
Scale	1
Skew	0.0ns
Sample rate	AFG31021/31022: 250MS/s AFG31051/31052: 500MS/s AFG31101/31102: 1GS/s AFG31151/31152/AFG31251/31252:2GS/s
Timer	1.00ms
Waveform list length	0
Sequence table length	0

The front-panel Default button does not reset the following settings:	
Language option	
Soft Keyboard	
Power-on settings	
Screen saver	
Click tone	
Beeper	
Brightness	
Saved setup files	
Saved waveform files	
License	
Calibration data	
GPIB setup	
Ethernet setup	
Access protection, password	Factory default: 123456

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