# Programmer Manual

Tektronix

## AWG710 & AWG710B
## 4GS/s / 4.2GS/s Arbitrary Waveform Generator

## 071-1414-00

This document supports firmware version 4.00 and above.

**www.tektronix.com**

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

## WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

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

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

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

# Table of Contents



## Getting Started



## Syntax and Commands

## Status and Events



## Examples



## Appendices



## Glossary and Index

# List of Figures

# List of Tables

# Preface

This is the programmer manual for the AWG710 and AWG710B Arbitrary Waveform Generators. This manual provides information necessary for operating the instrument over both the General Purpose Interface Bus (GPIB) and Ethernet interfaces.

This manual provides the following information:

- The *Getting Started* section describes how to connect and set up the waveform generator for remote operation.
- The *Syntax and Commands* section defines the command syntax and processing conventions and describes each command in the waveform generator command set.
- The *Status and Events* section explains the status information and event messages reported by the waveform generator.
- The *Programming Examples* section describes how to use the Sample Program floppy disk supplied with the waveform generator.
- The *Appendices* section contains various tables of reference information.
- The *Glossary and Index* section contains a glossary of common terms and an index to this manual.

## Related Manuals

Other documentation for the waveform generator includes:

- The *AWG710 & AWG710B Arbitrary Waveform Generator User Manual* (Tektronix part number 071–1413–00) describes the operation of the instrument.

# Getting Started

# Getting Started

The AWG710 Arbitrary Waveform Generator has GPIB and 10Base-T/100Base-TX Ethernet interface capability. You can write computer programs that remotely set the front panel controls or that transfer waveform data.

To help you get started with programming the waveform generator, this section includes the following subsections:

- *Manual Overview* - summarizes the type of programming information contained in each major section in this manual.
- *Setting Up Remote Communications Using GPIB* - describes how to connect the waveform generator to a controller through the GPIB interface, and how to set the appropriate front panel controls.
- *Setting Up Remote Communications Using Ethernet* - describes how to connect the waveform generator to a controller using the Ethernet interface and how to set the appropriate front panel controls.

## Manual Overview

A summary of the information provided in each major section of this manual follows:

### Syntax and Commands

The *Command Syntax* subsection, which begins on page 2-1, describes the structure and content of the messages your program sends to the waveform generator. You can use the Standard Commands for Programmable Instruments (SCPI) and IEEE 488.2 Common Commands. Figure 1-1 is an example of the syntax and command parts diagrams used in the *Command Syntax* subsection.

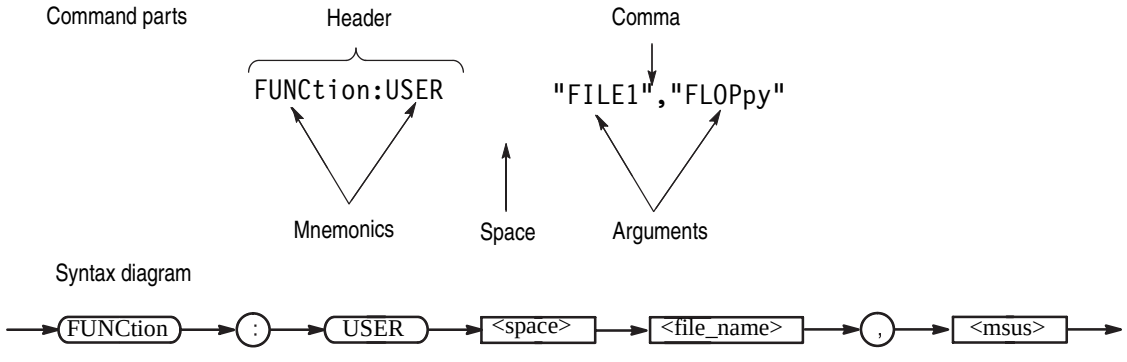



**Figure 1-1: Common message elements**

The *Command Syntax* subsection also describes the result of each command, and provides examples of how you might use it. The *Command Groups* subsection, which begins on page 2-13, provides a command list by functional area. The *Command Descriptions* subsection, which begins on page 2-25, arranges commands alphabetically. Figure 1-2 illustrates the two kinds of command lists.

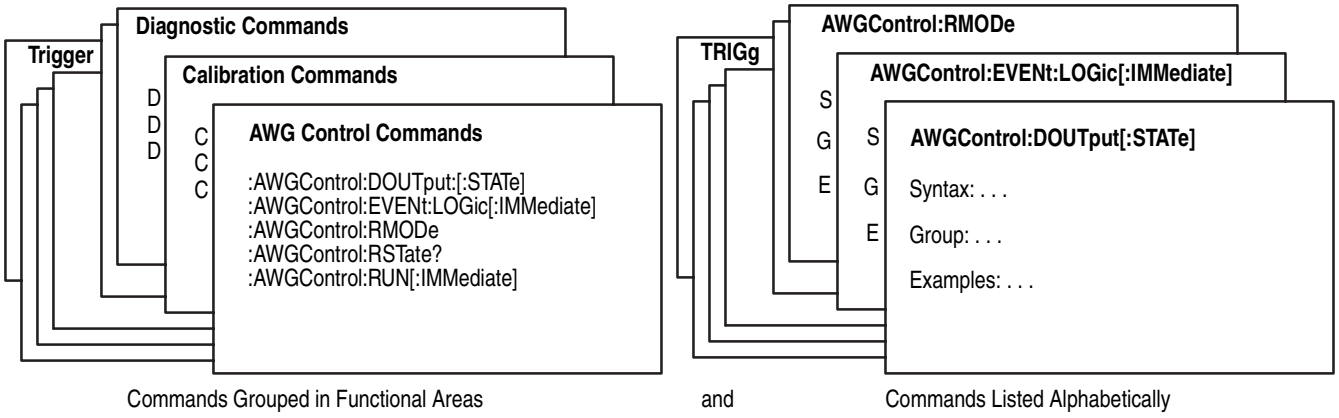



**Figure 1-2: Functional groupings and alphabetical list of commands**

## Status and Events Reporting

The program may request information from the waveform generator. The waveform generator provides information in the form of status and error messages. Figure 1-3 on page 1-3 illustrates the basic operation of this system.

The *Status and Events Reporting* subsection, which begins on page 3-1, describes how to use the status reporting functions that conform to SCPI and IEEE-488.2 in your programs.

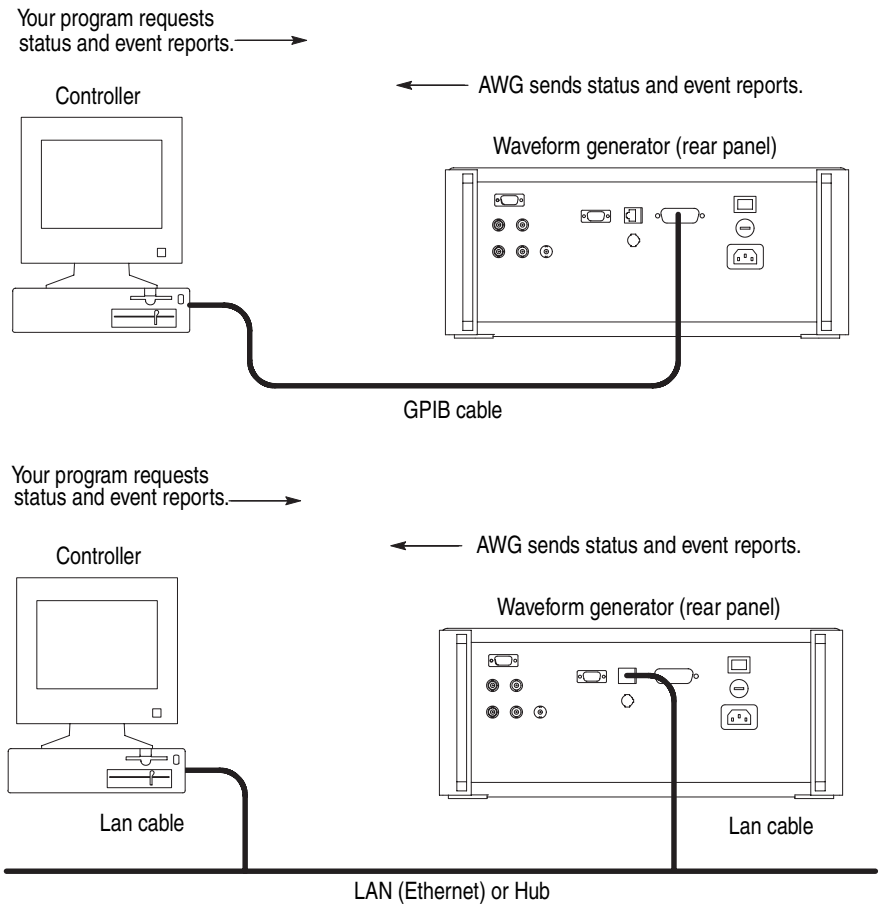



**Figure 1-3: Basic operation of status and events reporting**

## Programming Examples

The *Programming Examples* section, which begins on page 4-1, provides some sample waveform generator programs. A floppy disk (see Figure 1-4) is supplied with this manual. The disk contains a Microsoft Visual C++ and Visual BASIC source–code version of each program.

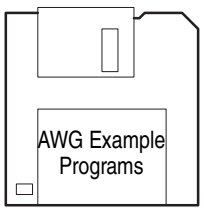



**Figure 1-4: The floppy disk**

## Setting Up Remote Communications Using GPIB

For remote operations, the instrument must be connected to the controller.

The waveform generator has a 24–pin GPIB connector on its rear panel, as shown in Figure 1-5. This connector has a D–type shell and conforms to IEEE Std 488.1–1987.

Attach an IEEE Std 488.1–1987 GPIB cable (Tektronix Part Number 012–0991–XX) to the GPIB connector.

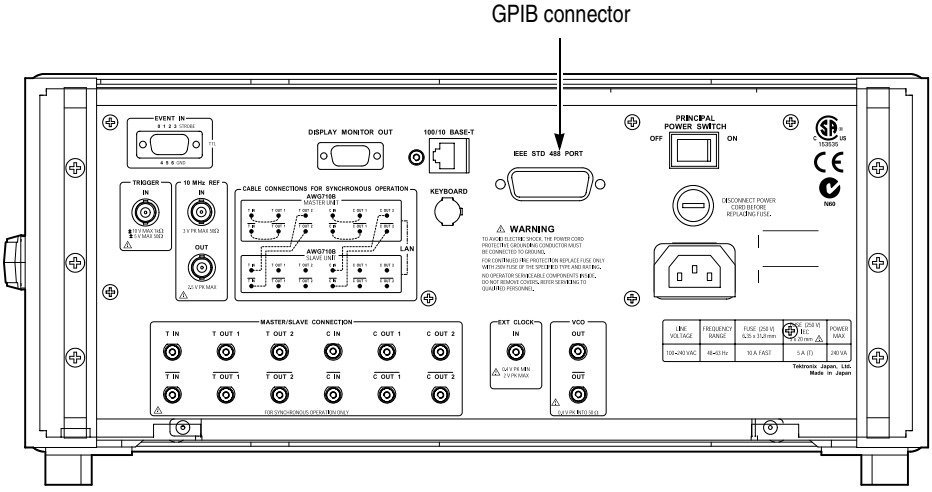



**Figure 1-5: GPIB connector location**

Stack GPIB connectors, if needed, as shown in Figure 1-6.

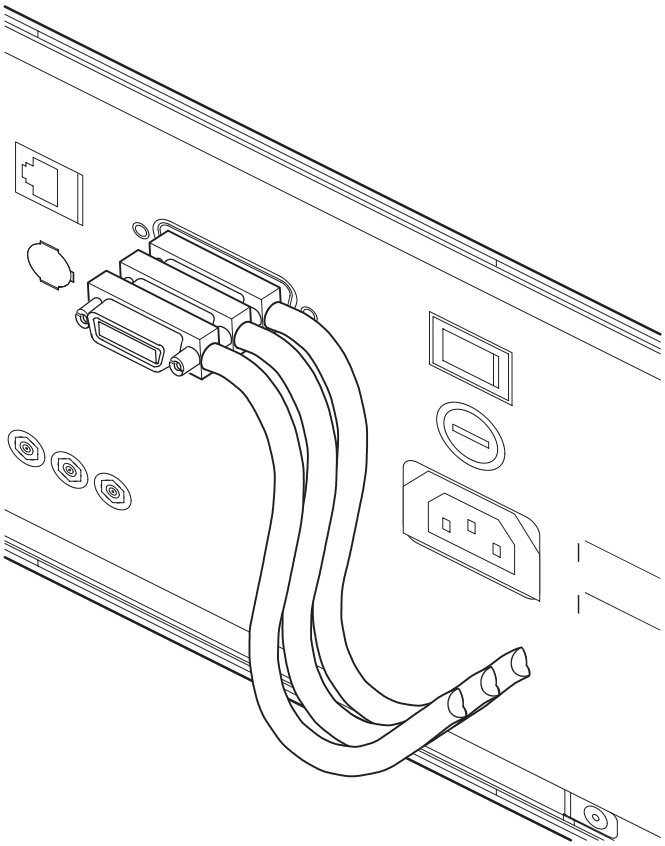


**Figure 1-6: How to stack GPIB connectors**

## GPIB Requirements

Follow these rules when you use your waveform generator with a GPIB network:

- Assign a unique device address to each device on the bus. Two devices can not share the same device address.
- Do not connect more than 15 devices to 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.
- While using the network, turn on at least two–thirds of the devices on the network.
- Connect the devices on the network in a star or linear configuration, as shown in Figure 1-7. Do not use loop or parallel configurations.

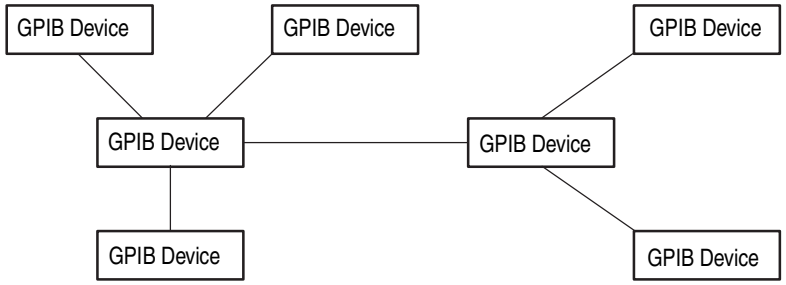



**Figure 1-7: Typical GPIB network configurations**

***NOTE.*** Appendix C: Network Interface Specification *provides more information about the GPIB configuration of the waveform generator.*

## Setting the GPIB Parameters

You must set the GPIB parameters of the waveform generator to match the configuration of the bus. Follow the steps below to set up the waveform generator for the GPIB interface.

1. Press the **UTILITY** button to display the Utility screen.
2. Press the **Comm** menu button at the bottom of the screen.
3. Move the cursor to the **Remote Control** field using the up/down (↑/↓) arrow buttons, then select **GPIB** using the left/right (←/→) arrow buttons.
4. Move the cursor to the **GPIB Configuration** field using the up/down (↑/↓) arrow buttons, then select **Talk/Listen** using either the general purpose knob or the left/right (←/→) arrow buttons. See Figure 1-8 on page 1-7.

5. Move the cursor to the **GPIB Address** field using the down (↓) arrow button. Set the address using either the general purpose knob or the keypad.

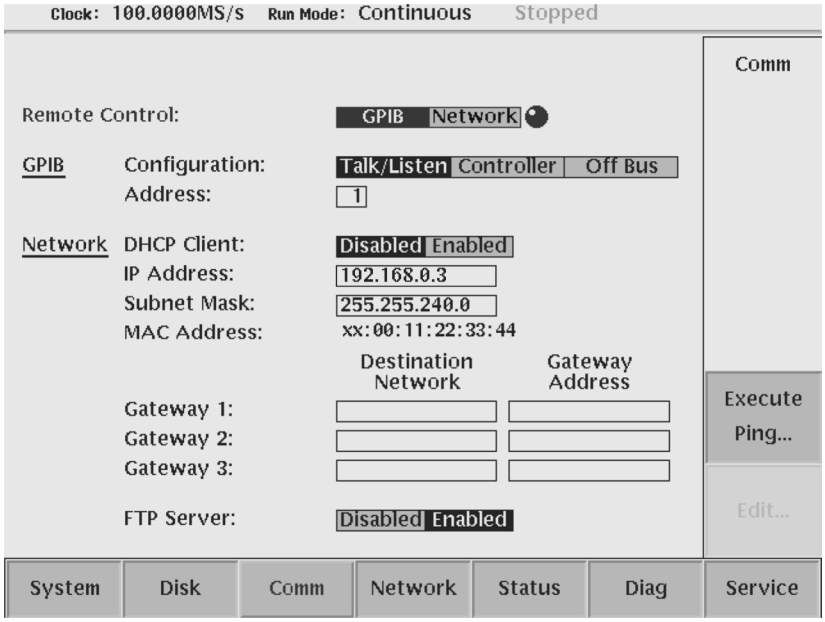



**Figure 1-8: Selecting the GPIB configuration and address**

The waveform generator is set up for bidirectional communication with your controller. Do the following to isolate the waveform generator from the bus:

Select **Off Bus** in the **GPIB Configuration** field.

This selection disables all communication with the controller.

# Setting Up Remote Communications Using Ethernet

***NOTE.*** *For remote operations, the instrument must be connected to the controller.*

The waveform generator has an Ethernet (10Base-T/100Base-Tx) port on the rear panel as shown in Figures 1-9.

Attach an Ethernet cable to the Ethernet port.

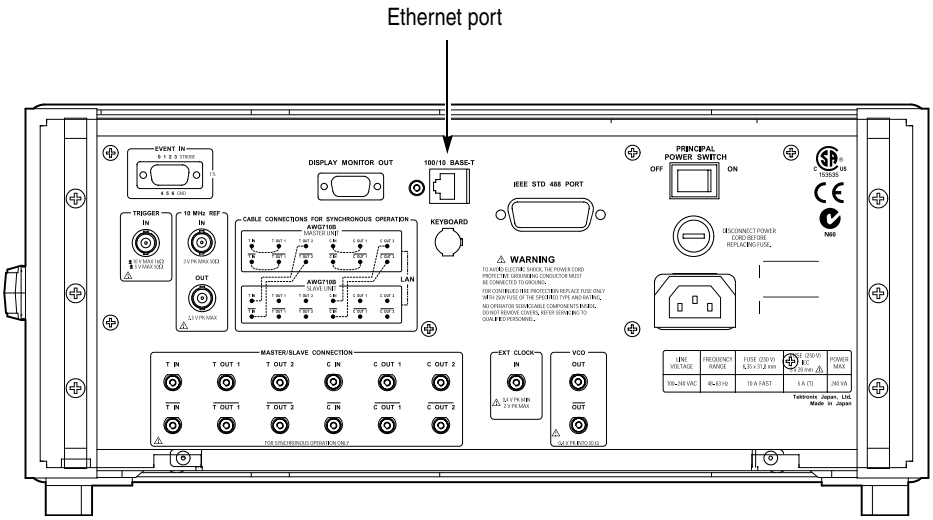



**Figure 1-9: Ethernet port location**

## Setting the Network Parameters

You must set the network parameters of the waveform generator to match the configuration of the network. After you have set these parameters, you can control the waveform generator through the Ethernet interface.

1. Press the **UTILITY** button to display the Utility screen.
2. Press the **Comm** menu button at the bottom of the screen.
3. Move the cursor to the **Remote Control** field using the up/down (↑/↓) arrow buttons, then select **Network** using the left/right (←/→) arrow buttons.
4. Move the cursor to the **Network IP Address** field using the up/down (↑/↓) arrow buttons, then press the **Edit...** button and set the address using the keypad. See Figure 1-10 on page 1-10.

   - Manual operation:

   **a.** Move the cursor to the **DHCP Client** field using the up/down (↑/↓) arrow buttons, then press **Disabled** using the left/right (←/→) arrow buttons.

   **b.** Move the cursor to the **IP Address** field using the up/down (↑/↓) arrow buttons, then press the **Edit...** button.

   **c.** Set the IP Address in IP Address dialog box.

   **d.** If necessary, use the **Subnet Mask** field to set the address.

   - Using DHCP:

   **e.** Move the cursor to the **DHCP Client** field using the up/down (↑/↓) arrow buttons, then press **Enabled** using the left/right (←/→) arrow buttons.

   **f.** AWG710 sends an acquisition request, then the server sends the address. The address is displayed in the **IP Address** field.
5. If necessary, use the **Destination Network** and **Gateway Address** fields to set the destination network and the address.

   You need to set the gateway address when the remote computers are connecting to another network that is connected to the network via a gateway. You can set up to three gateways.

   Set the FTP server to **Enabled** for access to the hard disk system of the instrument from a remote computer.

If you are not familiar with the network setup, consult with your network administrator.

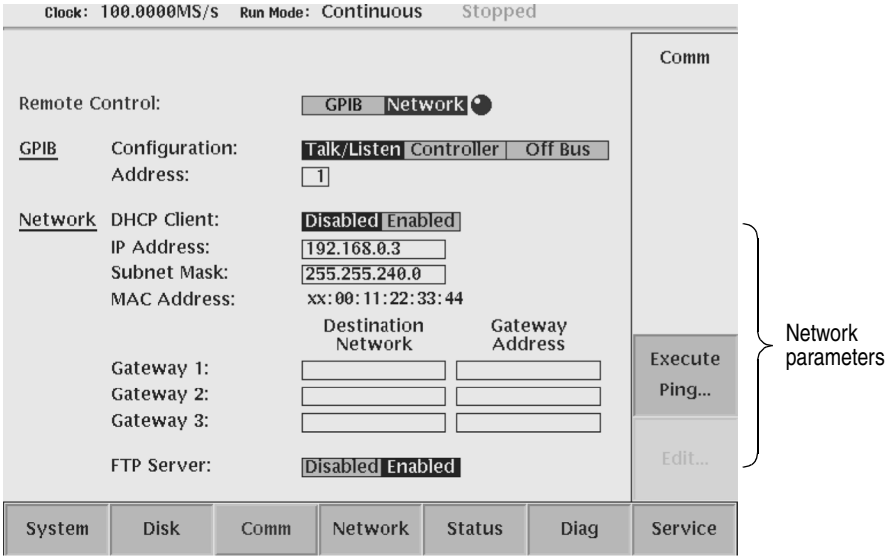



**Figure 1-10: Setting the Network parameters**

## Testing the Network Connection

After completing the connection and settings, verify that the waveform generator can recognize the network and the remote computers, or if the network can recognize the waveform generator. Follow these steps to use the “ping” command to verify that the instrument can communicate with the network:

1. Press the **UTILITY** button to display the Utility screen.
2. Press the **Network** or **Comm** bottom menu button.
3. Press the **Execute Ping** side button to display a dialog box.
4. Enter the IP address of the remote computer in the dialog box, and then push the **OK** side button.

   The ping command sends a packet to the remote computer specified by the IP address. When the computer receives the packet, it sends the packet back to the sender (waveform generator).

   When the waveform generator can communicate with the remote computer through the network the message in Figure 1-11 displays. If communication failed, the message box displays an error message such as `"no response from..."`.
5. Repeat steps 2 and 3 to verify the connection for other remote computers on the network.


**Figure 1-11: Message box to indicate the establishment of communication**

# Syntax and Commands

# Command Syntax

This section contains general information about command structure and syntax usage. You should familiarize yourself with this material before using the waveform generator command descriptions.

This manual describes commands and queries using Backus–Naur Form (BNF) notation. Table 2-1 defines standard BNF symbols.

**Table 2-1: BNF symbols and meanings**

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

## SCPI Commands and Queries

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

The SCPI language is based on a hierarchical or tree structure that represents a subsystem (see Figure 2-1). The top level of the tree is the root node; it is followed by one or more lower–level nodes.

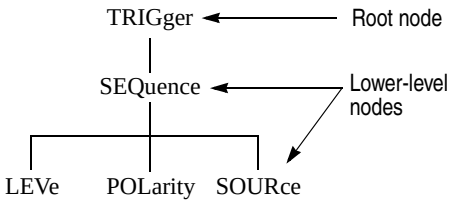



**Figure 2-1: Example of SCPI subsystem hierarchy tree**

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

## Creating Commands

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

In Figure 2-1 on page 2-2, TRIGger is the root node and SEQuence, LEVel, POLarity, and SOURce are lower–level nodes. To create an 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. The command descriptions, which begin on page 2-25, list the valid values for all parameters.

For example, :TRIGger:SEQuence:SOURce EXTernal is a valid SCPI command created from the hierarchy tree in Figure 2-1 on page 2-2.

## Creating Queries

To create a query, start at the root node of a tree structure, move down to the end of a branch, and add a question mark. :TRIGger:SEQuence:SOURce? is an example of a valid SCPI query using the hierarchy tree in Figure 2-1 on page 2-2.

## Query Responses

The query causes the waveform generator to return information about its status or settings. When a query is sent to the waveform generator, only the values are returned. When the returned value is a mnemonic, it is noted in abbreviated format, as shown in Table 2-2.

**Table 2-2: Query response examples**

| Query | Response |
|---|---|
| :SOURce:VOLTage:AMPLitude? | 1.000 |
| :AWGControl:RMODe? | CONT |

A few queries also initiate an operation action before returning information. For example, the *CAL? query runs a calibration.

## Parameter Types

Parameters are indicated by angle brackets, such as <file_name>. There are several different types of parameters, as listed in Table 2-3. The parameter type is listed after the parameter. Some parameter types are defined specifically for the AWG710 command set and some are defined by SCPI.

**Table 2-3: Parameter types used in syntax descriptions**

| Parameter Type | Description | Example |
|---|---|---|
| arbitrary block | A block of data bytes | #512234xxxxx...<br>where 5 indicates that the following 5 digits (12234) specify the length of the data in bytes;<br>xxxxx... indicates the data<br><br>or<br><br>#0xxxxx...<LF><&EOI> |
| boolean | Boolean numbers or values | ON or ≠ 0 : x<= -0.5, 0.5 <= x<br>OFF or 0 : -0.5 < x < 0.5 |
| discrete | A list of specific values | MIN, MAX |
| binary | Binary numbers | #B0110 |
| octal | Octal numbers | #Q75, #Q3 |
| hexadecimal | Hexadecimal numbers (0-9, A- 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, NR3 examples in this table |
| numeric_value | Flexible decimal number that may be type NR1, NR2, NR3, or specific value (MINimum, MAXmum). | See NR1, NR2, NR3 examples in this table |
| string | Alphanumeric characters (must be within quotation marks) | “Testing 1, 2, 3” |

## About MIN, MAX

You can use MINimum and MAXimum keywords in addition to Numeric in the commands with “numeric_value” parameter. You can set the minimum value or the maximum value by the use of this keywords. You can query the minimum value or the maximum value at tha time.

### Special Characters

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

### Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these commands as a combination of upper and lower case letters. The upper case letters indicate the accepted short form of a command, as shown in Figure 2-2. The accepted short form and the long form are equivalent and request the same action of the instrument.

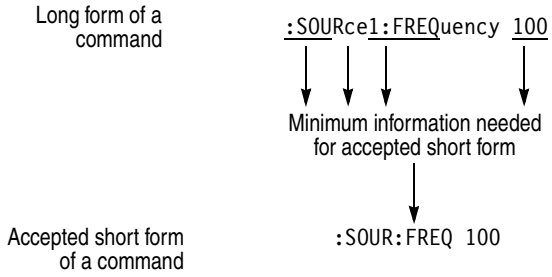



**Figure 2-2: Example of abbreviating a command**

***NOTE.*** *The numeric suffix of a command or query may be included in either the long form or short form; the AWG710 will default to "1" if no suffix is used.*

## Chaining Commands andQueries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, then add a semicolon (;), and finally add more commands or queries and semicolons until you are done. If the command following a semicolon is a root node, precede it with a colon (:). Figure 2-3 illustrates a chained message consisting of several commands and queries. The chained message should end in a command or query, not a semicolon. Responses to any queries in your message are separated by semicolons.

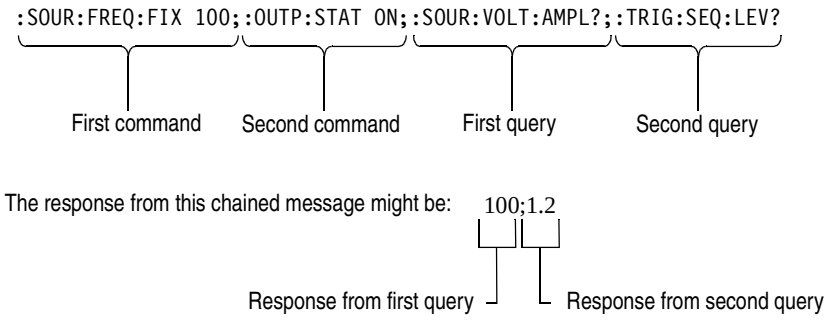



**Figure 2-3: Example of chaining commands and queries**

If a command or query has the same root and lower–level nodes as the previous command or query, you can omit these nodes. In Figure 2-4, the second command has the same root node (SEQuence) as the first command, so these nodes can be omitted.

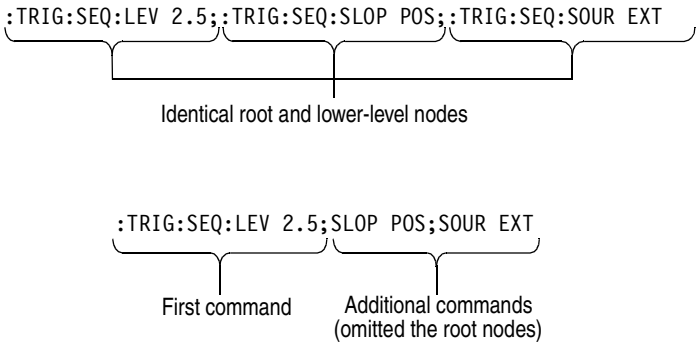



**Figure 2-4: Example of omitting root and lower–level nodes in a chained message**

### Unit and SI Prefix

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

You can omit the unit, but you must include the SI unit prefix. You can use either upper or lowercase units.

- V or v for voltage
- Hz, HZ, or hz for frequency
- ohm, OHM, or Ohm for impedance
- s or S for time

In the case of angle, you can use RADian and DEGree. The default unit is RADian.

The SI prefixes, which must be included, are shown below. Note that either lower or upper case prefixes can be used.

| SI prefix * | p/P | n/N | u/U | m/M | k/K | m/M | g/G |
|---|---|---|---|---|---|---|---|
| Corresponding power | $10^{-12}$ | $10^{-9}$ | $10^{-6}$ | $10^{-3}$ | $10^{3}$ | $10^{6}$ | $10^{9}$ |

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

*** Note that the prefix u/U is used instead of "μ".**

Use **mV** for **V**, and **MHz** for **Hz**.

## General Rules

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.

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

  is the same as

  ```
  :output:filter:lpass:frequency 200MHz
  ```

  and

  ```
  :OUTPUT:filter:LPASS:frequency 200MHz
  ```

***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 waveform generator complies with this standard.

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

- `*ESE 16`
- `*CLS`

The following are examples of common queries:

- `*ESR?`
- `*IDN?`

## Constructed Mnemonics

Some command headers list a range of mnemonics. When constructing the command, you select one mnemonic from the list. You then use the mnemonic in the command just as you do any other mnemonic. Mnemonic ranges can be presented in any of the following formats:

- MNEMonic[a|b|c]. The values a, b, and c represent the actual list of valid selections. You cannot list more than one value.
- For example, for the command :SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRess, the gateway mnemonic could be any of the following:GATeway1, GATeway2, or GATeway3. Therefore, a valid usage of this command would be: SYSTem:COMMunicate:LAN:GATeway1:ADDRess.
- MNEMonic<n>. The value of <n> is the upper range of valid suffixes. If the numeric suffix is omitted, the waveform generator uses the default value of "1".

### Source Channel Mnemonics

These commands specify the source channel to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| SOURce1 | CH1 signal of waveform generator |

### Output Channel Mnemonics

These commands specify the output channel to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| OUTPut1 | CH1 analog signal output |

### Direct D/A Output Mnemonics (Except option02)

These commands specify the direct D/A converter output to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| DOUTput1 | Direct output from CH1 D/A converter |

### Gateway Mnemonics

These commands specify the gateway to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| GATeway1 | Gateway 1 |
| GATeway2 | Gateway 2 |
| GATeway3 | Gateway 3 |

### Marker Mnemonics

These commands specify the marker to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| MARKer1 | The signal for the marker 1 |
| MARKer2 | The signal for the marker 2 |

### Remote Device Mnemonics

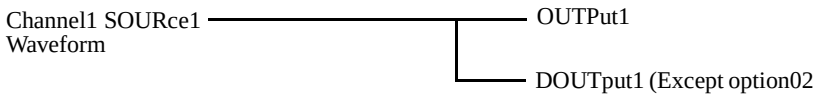
These commands specify the remote device to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| RDEVice1 | Network drive 1 |
| RDEVice2 | Network drive 2 |
| RDEVice3 | Network drive 3 |

### Source to Output Connections

The following illustrations shows the source to output connections for the AWG710/AWG710B instruments.

#### AWG710/AWG710B



## Syntax Diagrams

The syntax of each command and query is explained by both syntax diagrams and BNF notation. Figure2-5 shows some typical syntax diagram structures. The syntax diagrams are described by the following symbols and notation:

- Oval symbols contain literal elements, such as a command or query header and a nonquoted string argument.
- Circle symbols contain separators or special symbols, such as (:), (,), and (?).
- Box symbols contain the defined element, such as <NR1>.
- Arrow symbols connect elements to show the paths that can be taken through the diagram and, thereby, the order in which the elements can be sent in a command structure.
- Parallel paths show that only one of the paths can be taken in the command. See diagram 1 in Figure 2-5.
- A loop around an element(s) shows the element can be repeated. See diagram 2 in Figure 2-5.
- A path around a group of elements shows that those elements are optional. See diagram 3 in Figure 2-5.

***NOTE.*** *The unit and SI prefix that can be added to decimal numeric arguments are not described in the syntax diagram. See* Unit and SI Prefix *on page 2-7.*

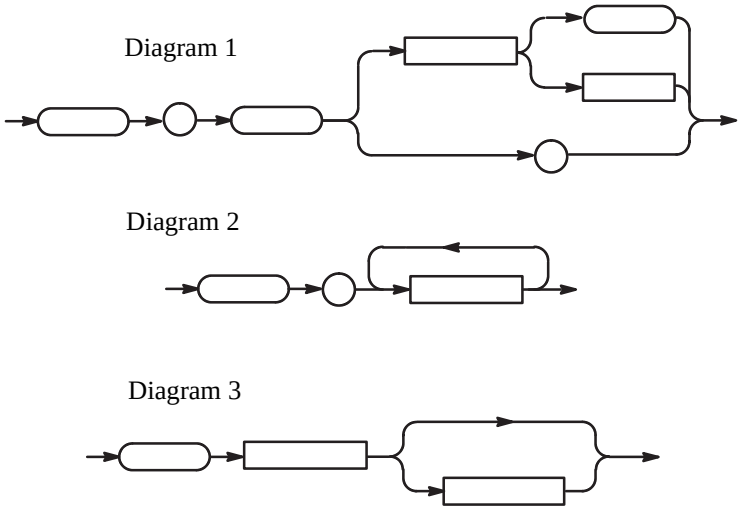



**Figure 2-5: Typical syntax diagrams**

# Command Groups

This section lists commands in two ways, by functional groups and alphabetically. The functional group list starts below. The alphabetical list provides more detail on each command and starts on page 2-25.

The GPIB interface conforms to SCPI (Standard Commands for Programmable Instruments) 1999.0 and IEEE Std 488.2-1987, except where noted.

## Functional Groups

Table 2-4 lists the functional groups into which the AWG710 and AWG710B Arbitrary Waveform Generator (AWG) commands are classified.

**Table 2-4: Functional groups in the AWG command set**

| Group | Function |
|---|---|
| AWG Control | Control operating mode |
| Calibration | Perform calibration |
| Diagnostic | Control self–test routines |
| Display | Control the presentation of information on the front panel display |
| Hardcopy | Dump the whole display into the file on the mass storage |
| Mass Memory | Control file operations on the mass storage |
| Output | Control the characteristics of the waveform output port |
| Source | Set waveform and marker output parameters, such as frequency and level |
| Status | Set and query the registers and queues of the reporting system |
| Synchronization | Control operation complete and pending command execution |
| System | Control miscellaneous instrument functions such as LAN, security, and time |
| Trigger | Synchronize the waveform generator actions with events |

# Command Quick Reference

The next page lists all the commands in each functional group and can be copied for use as a quick reference. The minimum accepted character string for each command is shown in uppercase characters.

**AWG Control commands**

:AWGControl:CLOCk:SOURce (?)
:AWGControl:DOUTput1:STATe (?)
:AWGControl:ENHanced:SEQuence:JMODe (?)
:AWGControl:EVENt:LOGic:IMMediate
:AWGControl:EVENt:SOFTware:IMMediate
:AWGControl:EVENt:TABLe:IMMediate
:AWGControl:FG:FREQuency:CW|FIXed (?)
:AWGControl:FG1:FUNCtion:SHAPe (?)
:AWGControl:FG1:POLarity (?)
:AWGControl:FG1:PULSe:DCYCle (?)
:AWGControl:FG:STATe (?)
:AWGControl:FG1:VOLTage:LEVel:IMMediate:AMPLitude (?)
:AWGControl:FG1:VOLTage:LEVel:IMMediate:OFFSet (?)
:AWGControl:MIX[:STATe] (?)
:AWGControl:RMODe (?)
:AWGControl:RSTate?
:AWGControl:RUN:IMMediate
:AWGControl:SREStore
:AWGControl:SSAVe
:AWGControl:STOP:IMMediate
:AWGControl:SYNChronous:ADDRess (?)
:AWGControl:SYNChronous:CALibration
:AWGControl:SYNChronous:CONNec (?)
:AWGControl:SYNChronous:MASTer[:STATe] (?)
:AWGControl:SYNChronous:SLAVe[:STATe] (?)

**Calibration commands**

*CAL?
:CALibration:ALL (?)

**Diagnostic commands**

:DIAGnostic:DATA?
:DIAGnostic:IMMediate (?)
:DIAGnostic:SELect (?)
*TST?

**Display commands**

:ABSTouch
:DISPlay:ENABle (?)
:DISPlay:HILight:COLor (?)

**Hardcopy commands**

:HCOPy:DESTination
:HCOPy:DEVice:COLor (?)
:HCOPy:DEVice:LANGuage (?)
:HCOPy:IMMediate
:HCOPy:SDUMp:IMMediate

**Mass memory commands**

:MMEMory:CATalog?
:MMEMory:CDIRectory (?)
:MMEMory:CLOSe
:MMEMory:COPY
:MMEMory:DATA (?)
:MMEMory:DELete
:MMEMory:FEED (?)
:MMEMory:INITialize
:MMEMory:MDIRectory
:MMEMory:MOVE
:MMEMory:MSIS (?)
:MMEMory:NAME (?)
:MMEMory:OPEN

**Output commands**

:OUTPut[1]:FILTer:LPASs:FREQuency (?)
:OUTPut[1]:ISTATe (?)
:OUTPut[1]:MARKer[:STATe] (?)
:OUTPut[1]:STATe] (?)

**Source commands**

:SOURce1:FREQuency:CW|:FIXed (?)
:SOURce1:FUNCtion:USER (?)
:SOURce1:MARKer<y>:VOLTage:LEVel:IMMediate:HIGH (?)
:SOURce1:MARKer<y>:VOLTage:LEVel:IMMediate:LOW (?)
:SOURce1:ROSCillator:SOURce (?)
:SOURce1:VOLTage:LEVel:IMMediate:AMPLitude (?)
:SOURce1:VOLTage:LEVel:IMMediate:OFFSet (?)(?)

**Status commands**

*CLS
*ESE (?)
*ESR?
*PSC (?)
*SRE (?)
:STATus:OPERation:CONDition?
:STATus:OPERation:ENABle (?)
:STATus:OPERation:EVENt?
:STATus:PRESet
:STATus:QUEStionable:CONDition?
:STATus:QUEStionable:ENABle (?)
:STATus:QUEStionable:EVENt?
*STB?

**Synchronization commands**

*OPC (?)
*WA

**System commands**

*IDN?
*OPT?
*RST
:SYSTem:BEEPer:IMMediate
:SYSTem:COMMunicate:LAN:DHCP:CLIent:LEASe:TIME (?)
:SYSTem:COMMunicate:LAN:DHCP:CLIent:STATe (?)
:SYSTem:COMMunicate:LAN:FTP:SERVer:STATe (?)
:SYSTem:COMMunicate:LAN:FTP:SERVer:VERSion (?)
:SYSTem:COMMunicate:LAN:GATeway<x>:ADDRess (?)
:SYSTem:COMMunicate:LAN:NFS:TLIMit (?)
:SYSTem:COMMunicate:LAN:PING?
:SYSTem:COMMunicate:LAN:RDEVice<x>:ADDRess (?)
:SYSTem:COMMunicate:LAN:RDEVice<x>:FSYStem (?)
:SYSTem:COMMunicate:LAN:RDEVice<x>:NAME (?)
:SYSTem:COMMunicate:LAN:RDEVice<x>:PROTocol (?)
:SYSTem:COMMunicate:LAN:RDEVice<x>:STATe (?)
:SYSTem:COMMunicate:LAN:SELF:ADDRess (?)
:SYSTem:COMMunicate:LAN:SELF:MADDress?
:SYSTem:COMMunicate:LAN:SELF:SMASk (?)
:SYSTem:DATE (?)
:SYSTem:ERRor:NEXT?
:SYSTem:KDIRection (?)
:SYSTem:KEYBoard:TYPE (?)
:SYSTem:KLOCk (?)
:SYSTem:SECurity:IMMediate
:SYSTem:TIME (?)
:SYSTem:UPTime?
:SYSTem:VERSion?

**Trigger commands**

:ABORt
*TRG
:TRIGger:SEQuence:IMMediate
:TRIGger:SEQuence:IMPedance (?)
:TRIGger:SEQuence:LEVel (?)
:TRIGger:SEQuence:POLarity (?)
:TRIGger:SEQuence:SLOPe (?)
:TRIGger:SEQuence:SOURce (?)
:TRIGger:SEQuence:TIMer (?)

# Command Summaries

Tables 2-5 through 2-17 describe each command in each of the 12 functional groups.

## AWG Control Commands

The AWG Control commands control operating modes. This command group is not SCPI approved.

**Table 2-5: AWG Control commands**

| Header | Description |
|---|---|
| :AWGControl:CLOCk:SOURce (?) | Select the source for the clock signal |
| :AWGControl:DOUTput[1] [:STATe] (?) | Output the raw D/A converter output |
| :AWGControl:ENHanced:SEQuence [:JMODe](?) | Select the jump mode. |
| :AWGControl:EVENt[:LOGic] [:IMMediate] | Generate the event signal for logic jump |
| :AWGControl:EVENt:SOFTware [:IMMediate] <line> | Jump to the specified line in the sequence file |
| :AWGControl:EVENt:TABLe [:IMMediate] | Generate the event signal for table jump |
| :AWGControl:FG:FREQuency [:CW\|:FIXed] (?) | Set the frequency of the function waveform. |
| :AWGControl:FG[1]:FUNCtion [:SHAPe] (?) | Select the function or type of waveform ( square wave, sine wave, etc. ) |
| :AWGControl:FG[1]:POLarity (?) | Set the polarity of the function waveform |
| :AWGControl:FG[1]:PULSe :DCYCle(?) | Set the the duty cycle of the pulse waveform |
| :AWGControl:FG[:STATe] (?) | Turn the function generator mode on or off |
| :AWGControl:FG[1]:VOLTage [:LEVel][:IMMediate] [:AMPLitude] (?) | Set the peak-to-peak voltage of the function waveform |
| :AWGControl:FG[1]:VOLTage [:LEVel][:IMMediate]:OFFSet(?) | Set the offset voltage of the function waveform |
| :AWGControl:MIX[:STATe] (?) | Set the operation mode to the waveform mixed mode |
| :AWGControl:RMODe (?) | Select the run mode, such as triggered or gated |
| :AWGControl:RSTate? | Query the current running status |
| :AWGControl:RUN[:IMMediate] | Enable the output |
| :AWGControl:SREStore | Restore the settings from the specified file |
| :AWGControl:SSAVe | Store the settings to the specified file |

**Table 2-5: AWG Control commands (cont.)**

| Header | Description |
|---|---|
| :AWGControl:STOP[:IMMediate] | Stop the output |
| :AWGControl:SYNChronous:ADDRess (?) | Sets the IP address of the slave device |
| :AWGControl:SYNChronous: CALibration | Execute the Trigger Timing Calibration for the Synchronous Operation. |
| :AWGControl:SYNChronous:CONNect (?) | Control the connection of the slave machine |
| :AWGControl:SYNChronous:MASTer [:STATe] (?) | Set the AWG to the master machine |
| :AWGControl:SYNChronous:SLAVe [:STATe] (?) | Set the AWG to the slave machine |

## Calibration Commands

The Calibration commands calibrate the waveform generator.

**Table 2-6: Calibration commands**

| Header | Description |
|---|---|
| *CAL? | Perform calibration |
| :CALibration[:ALL] (?)e | Perform calibration |

## Diagnostic Commands

The Diagnostic commands control self–test diagnostic routines.

**Table 2-7: Diagnostic commands**

| Header | Description |
|---|---|
| :DIAGnostic:DATA? | Query results of self–test |
| :DIAGnostic[:IMMediate] (?) | Start the self–test |
| :DIAGnostic:SELect (?) | Select the self–test routine |
| *TST? | Perform self–test |

## Display Commands

The Display commands mimic manipulation of front–panel controls and set the presentation of textual information on the front panel display.

**Table 2-8: Display commands**

| Header | Description |
|---|---|
| :ABSTouch | Perform the function corresponding to the front–panel control selected |
| :DISPlay:ENABle (?) | Control ON/OFF of the display |
| :DISPlay:HILight:COLor (?) | Control hilight of the display |

## Hardcopy Commands

The Hardcopy commands are used to print the entire display to a specified file rather than printing to an external device.

The hardcopy commands used in this application do not conform to the 1999 SCPI hardcopy standard. (The 1999 SCPI standards state that the MMEMory:OPEN and MMEMory:CLOSe commands are used to open and close the file specified by MMEMory:NAME, to accommodate feeding data from the HCOPy subsystem. This state–dependent style of feeding data is not used in the waveform generator.) Instead, the hardcopy commands are implemented in a way that more closely resembles previous waveform generator usage. The waveform generator implements the hardcopy commands as illustrated in the following example:

```
MMEMory:NAME ÓSAMPLE1.BMPÓ
MMEMory:OPEN
HCOPy:DESTination ÓMMEMÓ
HCOPy
MMEM:CLOSe
```

The above command sequence can be written as follows for the waveform generator:

```
MMEMory:NAME ÓSAMPLE1.BMPÓ
HCOPy
```

In this case, the entire display will be written to the SAMPLE1.BMP file.

**Table 2-9: Hardcopy commands**

| Header | Description |
|---|---|
| :HCOPy:DESTination | Set the destination |
| :HCOPy:DEVice:COLor (?) | Select the color, or monochrome |
| :HCOPy:DEVice:LANGuage (?) | Select the data format |
| :HCOPy[:IMMediate] | Initiate the plot, or print immediately |
| :HCOPy:SDUMp[:IMMediate] | Plot or print the whole display |

## Mass Memory Commands

The Mass Memory commands provide mass storage capabilities.

**Selecting Mass Memory Devices.** The waveform generator supports the devices listed below. The network drives can be specified with the SYSTem command group.

**Table 2-10: Mass storage in AWG710 and AWG710B**

| String argument | Description |
|---|---|
| MAIN | Internal hard disk drive |
| FLOP or FLOPPY | Internal floppy disk drive |
| NET1 | Network drive 1 |
| NET2 | Network drive 2 |
| NET3 | Network drive 3 |

**File Names.** The <file_name> parameter is described in some Mass Memory commands with a string. The content of the string depends on the format needs of the mass storage media. In particular, the file name may contain characters for specifying subdirectories (e.g. "/") and the period separator (".). The instrument checks the file format when reading, and processes the file based on its content, regardless of the file extention.

**Table 2-11: Mass Memory commands**

| Header | Description |
|---|---|
| :MMEMory:CATalog? | Query information on the mass storage media |
| :MMEMory:CDIRectory (?) | Change the default directory for a file system |
| :MMEMory:CLOSe | Close the file specified in NAME |
| :MMEMory:COPY | Copy an existing file to a new file |
| :MMEMory:DATA (?) | Load data into the file |
| :MMEMory:DELete | Remove a file |
| :MMEMory:FEED (?) | Feed data into the file specified in NAME |
| :MMEMory:INITialize | Initialize the specified mass storage |
| :MMEMory:MDIRectory | Make a directory |
| :MMEMory:MOVE | Move an existing file to another file |
| :MMEMory:MSIS (?) | Select the current mass storage |
| :MMEMory:NAME (?) | Set the file name to be opened or closed |
| :MMEMory:OPEN | Open the file specified in NAME |

## Output Commands

The Output commands control the characteristics of the waveform output port.

**Table 2-12: Output commands**

| Header | Description |
|---|---|
| :OUTPut[1]:FILTer[:LPASs] :FREQuency (?) | Determine the cutoff frequency of the low pass filter |
| :OUTPut[1]:MARKer[:STATe] (?) | Control whether the all marker output terminal is open or closed |
| :OUTPut[1][:STATe] (?) | Control whether the output terminal is open or closed |
| :OUTPut[1]:ISTate (?) | Set the inverted output on or off |

## Source Commands

The Source commands set waveform and marker output parameters, such as frequency and level.

**Table 2-13: Source commands**

| Header | Description |
|---|---|
| [:SOURce[1]]:FREQuency [:CW\|:FIXed](?) | Set sampling frequency for outputting waveform |
| [:SOURce[1]]:FUNCtion:USER (?) | Specify the user–defined waveform or pattern file |
| [:SOURce[1]]:MARKer[1\|2] [:LEVEL][:IMMediate]:HIGH (?) | Set high level for marker output |
| [:SOURce[1]]:MARKer[1\|2] [:LEVEL][:IMMediate]:LOW (?) | Set low level for marker output |
| [:SOURce[1]]:ROSCillator :SOURce (?) | Select the reference oscillator source |
| [:SOURce[1]]:VOLTage[:LEVel] [:Immediate][:AMPLitude] (?) | Set the actual magnitude of the output signal |
| [:SOURce[1]]:VOLTage[:LEVel] [:Immediate]:OFFSet (?) | Set the offset that is added to the output signal |

## Status Commands

The external controller uses the Status commands to coordinate operation between the waveform generator and other devices on the bus. The Status commands set and query the registers/queues of the waveform generator event/status reporting system. For more information about the registers and queues described in Table 2-14, refer to the *Status and Event Reporting* section on page 3-1.

**Table 2-14: Status commands**

| Header | Description |
|---|---|
| *CLS | Clear all the event registers and queues |
| *ESE (?) | Set and query ESER |
| *ESR? | Query SESR |
| *PSC (?) | Set power–on status clear flag |
| *SRE (?) | Set and query SRER |
| :STATus:OPERation:CONDition? | Query the contents of OCR |
| :STATus:OPERation:ENABle (?) | Set the enable mask of OENR |
| :STATus:OPERation[:EVENt]? | Query the contents of OEVR |
| :STATus:PRESet | Preset OENR and QENR |
| :STATus:QUEStionable:CONDition? | Query the contents of QCR |
| :STATus:QUEStionable:ENABle (?) | Set the enable mask of QENR |
| :STATus:QUEStionable[:EVENt]? | Query the contents of QEVR |
| *STB? | Query SBR |

## Synchronization Commands

The external controller uses the Synchronization commands to prevent external communications from interfering with waveform generator operation.

**Table 2-15: Synchronization commands**

| Header | Description |
|---|---|
| *OPC (?) | Generate or return the operation complete message |
| *WAI | Hold off all commands until all pending operations complete |

## System Commands

The System commands control miscellaneous instrument functions, such as LAN communication, security, and time.

**Table 2-16: System commands**

| Header | Description |
|---|---|
| *IDN? | Query ID information about the waveform generator |
| *OPT? | Query installed options |
| *RST | Reset the waveform generator |
| :SYSTem:BEEPer[:IMMediate] | Generate an audible tone |
| :SYSTem:COMMunicate:LAN:DHCP [:CLIent]:LEASe:TIME (?) | Control the lease time of DHCP client function |
| :SYSTem:COMMunicate:LAN:DHCP [:CLIent][:STATe] (?) | Control the DHCP client function |
| :SYSTem:COMMunicate:LAN:FTP [:SERVer][:STATe] (?) | Control the FTP server function |
| :SYSTem:COMMunicate:LAN:FTP [:SERVer]:VERSion (?) | Change the FTP version |
| :SYSTem:COMMunicate:LAN :GATeway:ADDRess (?) | Set IP address of the gateway |
| :SYSTem:COMMunicate:LAN :NFS:TLIMit (?) | Set the timeout of NFS |
| :SYSTem:COMMunicate:LAN:PING? | Execute PING test for the specified IP address |
| :SYSTem:COMMunicate:LAN :RDEVice[1\|2\|3]:ADDRess (?) | Set IP address of the remote host |
| :SYSTem:COMMunicate:LAN :RDEVice[1\|2\|3]:FSYStem (?) | Set the mount directory of the remote host |
| :SYSTem:COMMunicate:LAN :RDEVice[1\|2\|3]:NAME (?) | Set the name of the remote host |
| :SYSTem:COMMunicate:LAN :RDEVice[1\|2\|3]:PROTocol (?) | Set the protocol of the communication between the waveform generator and the remote host |
| :SYSTem:COMMunicate:LAN :RDEVice[1\|2\|3][:STATe] (?) | Control whether the communication with the specified remote host is enabled |
| :SYSTem:COMMunicate:LAN [:SELF]:ADDRess (?) | Set IP address of the waveform generator |
| :SYSTem:COMMunicate:LAN [:SELF]:MADDress? | Query the MAC address of the waveform generator |
| :SYSTem:COMMunicate:LAN [:SELF]:SMASk (?) | Set the subnet mask of the waveform generator |
| :SYSTem:DATE (?) | Set the internal calender |

**Table 2-16: System commands (cont.)**

| Header | Description |
|---|---|
| :SYSTem:ERRor[:NEXT]? | Query the next entry from the waveform generator's error/event queue |
| :SYSTem:KDIRection (?) | Set the direction of cursor movement controlled by the general purpose knob |
| :SYSTem:KEYBoard[:TYPE] (?) | Select the keyboard type |
| :SYSTem:KLOCk (?) | Lock the front panel and keyboard |
| :SYSTem:SECurity:IMMediate | Destroy all data and settings for security |
| :SYSTem:TIME (?) | Set the internal clock |
| :SYSTem:UPTime? | Query elapsed time from the power–on |
| :SYSTem:VERSion? | Query the SCPI version number |

## Trigger Commands

The Trigger commands synchronize the waveform generator actions with events.

**Table 2-17: Trigger commands**

| Header | Description |
|---|---|
| ABORt | Reset the trigger system |
| *TRG | Generate the trigger event |
| :TRIGger[:SEQuence][:IMMediate] | Immediately trigger the sequence operation |
| :TRIGger[:SEQuence]:IMPedance (?) | Select the input impedance of the external trigger |
| :TRIGger[:SEQuence]:LEVel (?) | Set the trigger level |
| :TRIGger[:SEQuence]:POLarity (?) | Select the polarity of the trigger signal |
| :TRIGger[:SEQuence]:SLOPe (?) | Select the slope of the trigger signal |
| :TRIGger[:SEQuence]:SOURce (?) | Select the source for the event detector |
| :TRIGger[:SEQuence]:TIMer (?) | Set the period of the internal clock |

# Command Descriptions

This section lists each command and query in the waveform generator command set in alphabetical order. Each command entry includes a command description and command group, related commands (if any), syntax, and arguments. Each entry also includes one or more usage examples.

This section fully spells out headers, mnemonics, and arguments with the minimum spelling shown in upper case. For example, to use the abbreviated version of the `:SOURce:FREQuency` command, just type `:SOUR:FREQ`.

The symbol "(?)" follows the command header of commands that can be used as either a command or a query; the symbol "?" follows commands that can only be used as a query. Commands that are command–only or query–only are noted as such.

## ABORt (No Query Form)

This command resets the trigger system and places all trigger sequences in the idle state. This command is equivalent to depressing the FORCE TRIGGER button on the front panel in the gated mode.

**Group** Trigger

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

**Syntax** `ABORt`


**Arguments** None

**Examples** `ABORt`
resets the trigger system.

## :ABSTouch (No Query Form)

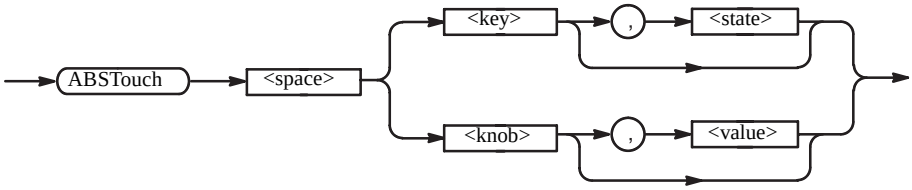
This command performs the functions that are manually set by pressing the corresponding front–panel key and button, or by rotating the corresponding knob. This command works even when the instrument is in the keylock or local lockout states.

**Group** Display

**Related Commands** None

**Syntax**

```
ABSTouch <key>[,<boolean>]
ABSTouch <knob>[,<value>]
```




**Arguments**

```
<key>::= BOTTom[1] | BOTTom2 | BOTTom3 | BOTTom4 | BOTTom5 |
BOTTom6 | BOTTom7 | SIDe[1] | SIDe2 | SIDe3 | SIDe4 | SIDe5 |
CMENu | RUN | DARRow | UARRow | LARRow | RARRow | SETup | APPL |
EDIT | UTILity | HARDcopy | TOGGle | SHIFt | ENTer | VMENu |
QKEDit | HMENu | TMENu | FTRigger | FEVent | SEVen |MEGa | EIGHt
| KILo | NINe | MILLi | FOUR | MICRo | FIVe | NANo | SIX | PICo |
ONE | D | TWO | E | THRee | F | ZERo | A | POINt | B | SIGN | C |
CLR | G | DELete | INF | RETurn | OUTPut[1] | IOUTput[1] | MOUTput

<knob>::= OFFSet | LSCale | HSHift | SSCale | LEVel | GPKNob

<boolean>::= ON | OFF | 1 | 0
```

This argument sets the press and release of the specified front panel key. If you specify ON or nonzero value in this argument, the front panel key is set to press. If you specify OFF or zero value in this argument, the front panel key is set to release. When the argument is not specified, 1 is set.

```
<value>::= <NR1>
```

This argument sets the rotating direction and quantities of the specified front panel knob. If you specify a positive value in this argument, the knob rotates clockwise. If you specify a negative value in this argument, the knob rotates counterclockwise. When the argument is not specified, 1 is set.

Figure 2-6 shows ABSTouch arguments corresponding to the associated controls.

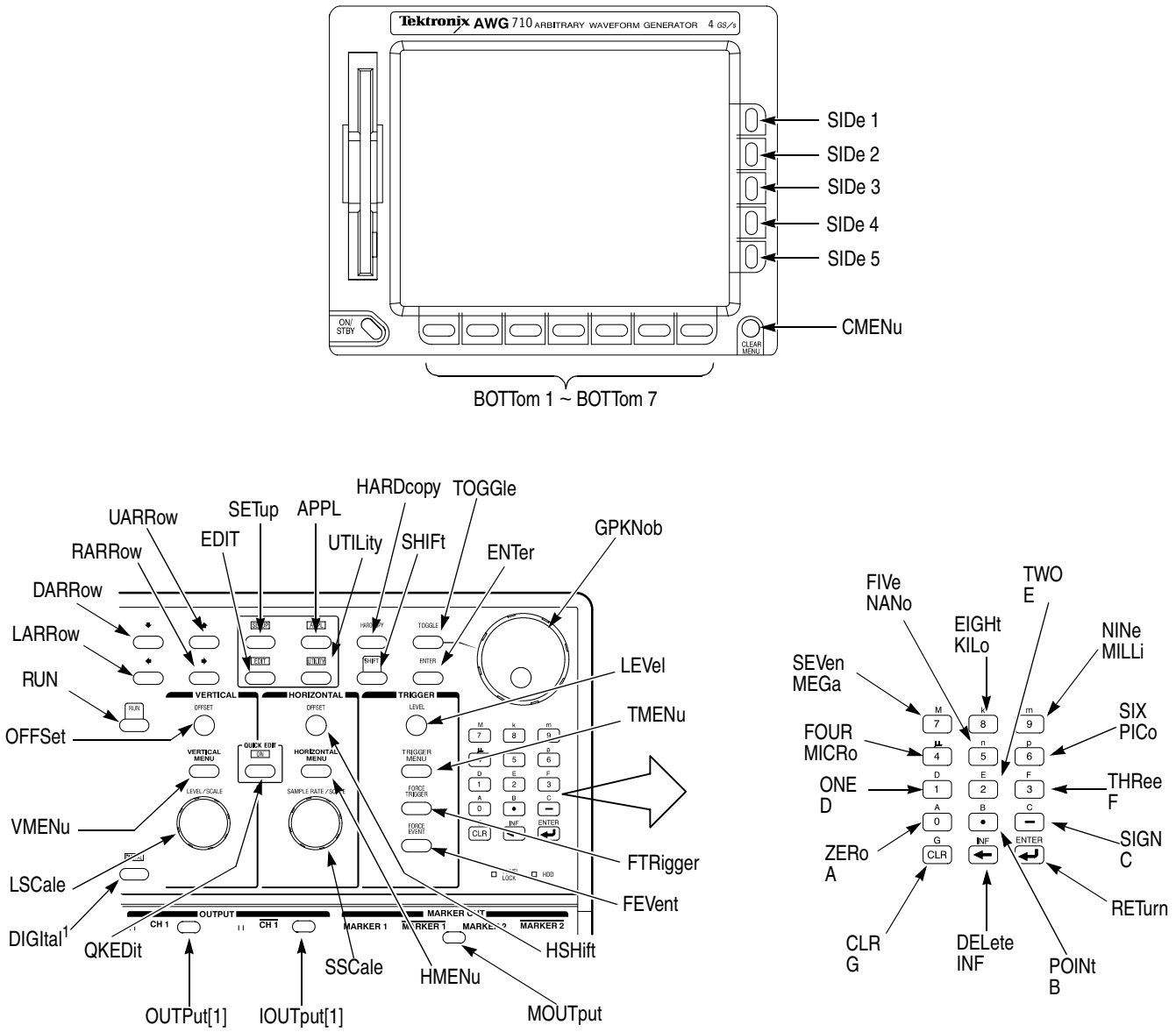



**Figure 2-6: ABSTouch arguments and Front panel**

**Examples** `ABSTOUCH SETUP`

displays the setup menu that is displayed by pressing the SETUP button on the front panel.

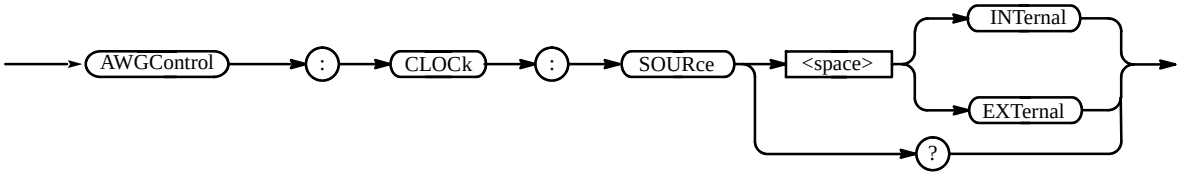
## :AWGControl:CLOCk:SOURce (?) (only AWG710B)

This command selects the clock source as either Internal or External.

**Group** AWG Control

**Syntax** `:AWGControl:CLOCk:SOURce { INTernal | EXTernal }`

`:AWGControl:CLOCk:SOURce?`




**Arguments** `INTernal` select the internal clock derived from the reference clock as a clock source.

`EXTernal` select the external clock signal connected to the EXT CLOCK IN on the rear panel as a clock source.

At `*RST`, this parameter is set to `INTernal`.

**Examples** `:AWGControl:CLOCk:SOURce EXTernal`
selects the external clock source.

## :AWGControl:DOUTput[1][:STATe] (?) (except option02)

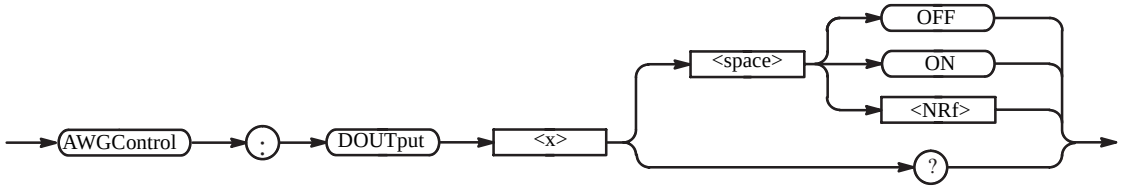
This command supplies raw output of the waveform generator D/A converter for the specified channel. The setting :OUTPut:FILTer command and :SOURce:VOLTage:OFFSet command are ignored.

**Group** AWG Control

**Related Commands** :SOURce:VOLTage command group, :OUTPut:FILTer command group

**Syntax**

```
:AWGControl:DOUTput[1][:STATe] <boolean>
:AWGControl:DOUTput[1][:STATe]?
```




**Arguments**

<boolean> ::={ OFF | ON | 0 | 1 }

OFF or 0 provides the D/A converter output normally.

ON or 1 provides raw output of the D/A converter.

At *RST, this value is set to 0.

**Examples**

```
AWGControl:DOUTput1:STATe ON
```

supplies the D/A converter output directly to CH 1.

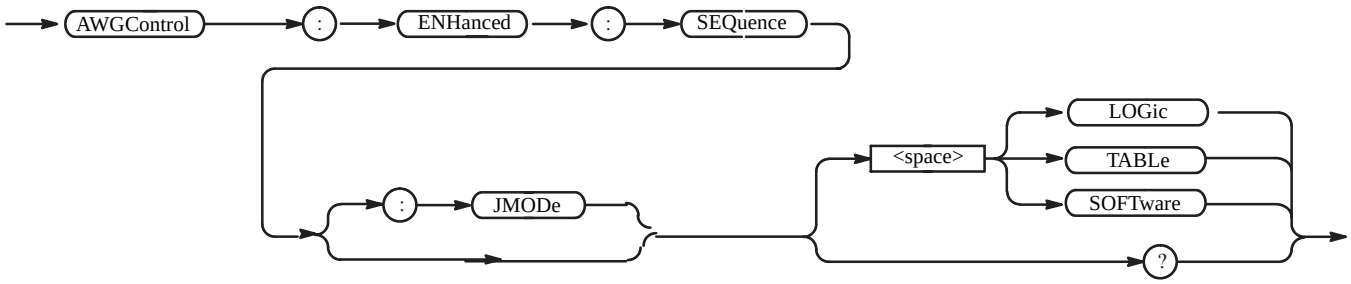
## :AWGControl:ENHanced:SEQuence[:JMODe] (?)

This command selects the jump mode in the sequence of the enhanced mode.

**Group** AWG Control

**Syntax** `:AWGcontrol:ENHanced:SEQuence[:JMODe] {LOGic | TABLe | SOFTware}`

`:AWGControl:ENHanced:SEQuence[:JMODe]?`




**Arguments**

`LOGic` The jump mode is ”logic”.
`TABLe` The jump mode is ”table”.
`SOFTware` The jump mode is ”software”.

At `*RST`, this value is set to TABLe.

**Examples** `:AWGControl:ENHanced:SEQuence SOFTware`
sets the jump mode to software.

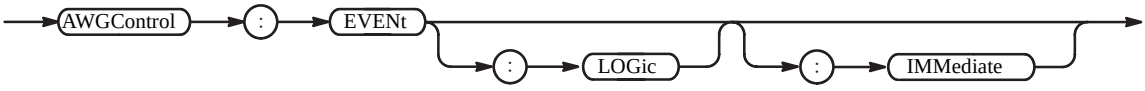
## :AWGControl:EVENt[:LOGic][:IMMediate] (No Query Form)

This command generates a trigger event for the “logic jump” specified in the sequence file. This has the same effect as pressing the FORCE EVENT button on the front panel.

**Group** AWG Control

**Related Commands** `:AWGControl:RUN[:IMMediate], *TRG`

**Syntax** `:AWGControl:EVENt[:LOGic][:IMMediate]`



**Arguments** None

**Examples** `:AWGControl:EVENt:LOGic:IMMediate`
generates a trigger event for the “logic jump”.

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

This command jumps to a specified line in a sequence file. To enable this command, a sequence file must be loaded and software jump mode must be set in the sequence file.

This command will return a “Settings conflict” error (code:-221) when any of these conditions are present:

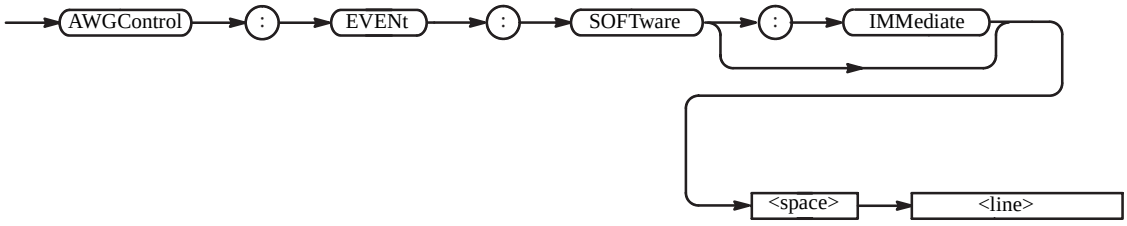
1) The waveform generator is not in Enhanced mode.
2) No sequence file is loaded.
3) The Jump Mode setting of the sequence file is not Software.

It also will return a “Data out of range” error (code:-222) if the <line> argument is less than or equal to zero, or greater than the number of steps of the loaded sequence file.

**Group** AWG Control

**Related Commands** None

**Syntax** `:AWGControl:EVENt:SOFTware[:IMMediate] <line>`




**Arguments** `<line>::=<NR1>` is the line number to be jumped to in the sequence file.

**Examples** `:AWGControl:EVENt:SOFTware:IMMediate 10`
jumps to line 10 in the sequence file.

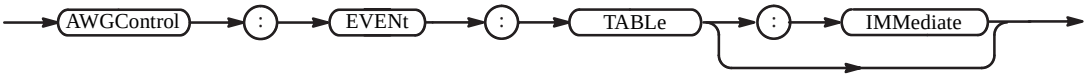
## :AWGControl:EVENt:TABLe[:IMMediate] (No Query Form)

This command generates a trigger event for the “table jump” specified in the sequence file when a jump mode of sequence (ENHanced mode) is a Table.

**Group** AWG Control

**Related Commands** None

**Syntax** `:AWGControl:EVENt:TABLe[:IMMediate]`




**Arguments** None

**Examples** `:AWGControl:EVENt:TABLe:IMMediate`
generates a trigger event for the “table jump”.

## :AWGControl:FG:FREQuency[:CW|:FIXed] (?)

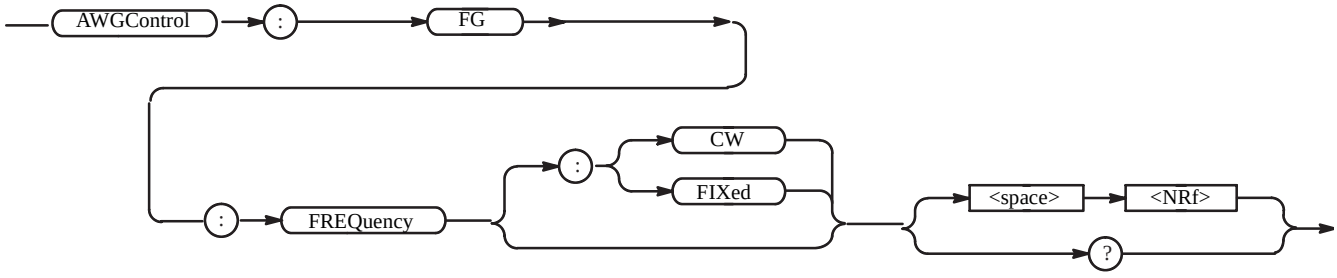
This command adjusts the frequency of the function waveform.
This query returns the frequency currently set.

CW (Continuous Wave) and FIXed are aliases, and have the same effect.

**Group** AWG Control

**Syntax** `:AWGControl:FG:FREQuency[:CW|:FIXed] <numeric_value>`

`:AWGControl:FG:FREQuency[:CW|:FIXed]?`



**Arguments** <numeric_value> is the output waveform frequency. The range is 1Hz to 400 MHz.

At *RST, this value is set to 20 MHz.

**Examples** :AWGControl:FG:FREQuency 10MHz
sets the frequency to 10 MHz.

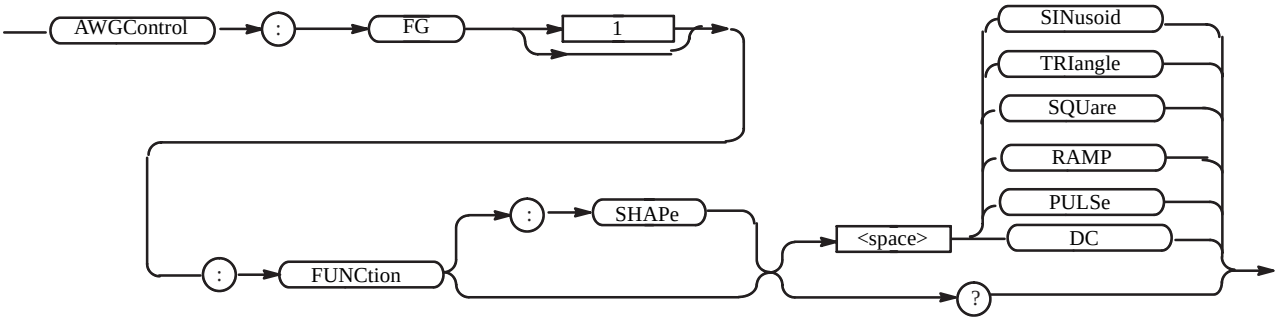
## :AWGControl:FG[1]:FUNCtion[:SHAPe] (?)

This command selects a standard function waveform (as opposed to a waveform file).
This query returns the currently selected standard function waveform.

**Group** AWG Control

**Syntax** :AWGControl:FG[1]:FUNCtion[:SHAPe] <shape>

:AWGControl:FG[1]:FUNCtion[:SHAPe] ?




**Arguments** SINusoid selects a sine wave function waveform.
TRIangle selects a triangle function waveform.
SQUare selects a square wave function waveform.

RAMP selects a ramp function waveform.
PULSe selects a pulse function waveform.
DC selects a DC function waveform.

At *RST, this value is set to SINusoid

**Examples** :AWGControl:FG1:FUNCtion RAMP
sets the standard function waveform to RAMP.

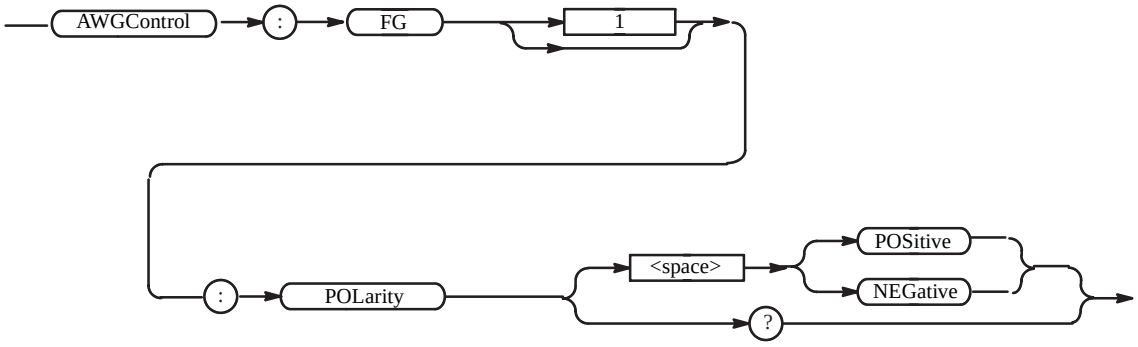
# :AWGControl:FG[1]:POLarity (?)

This command sets polarity of the function waveform.
This query returns polarity currently set.

**Group** AWG Control

**Syntax** `:AWGControl:FG[1]:POLarity {POSitive | NEGative}`

`:AWGControl:FG[1]:POLarity?`




**Arguments** `POSitive` sets waveform to positive polarity.
`NEGative` sets waveform to negative polarity.

At `*RST`, this value is set to POSitive.

**Examples** `:AWGControl:FG1:POLarity POSitive`
sets the polarity to positive.

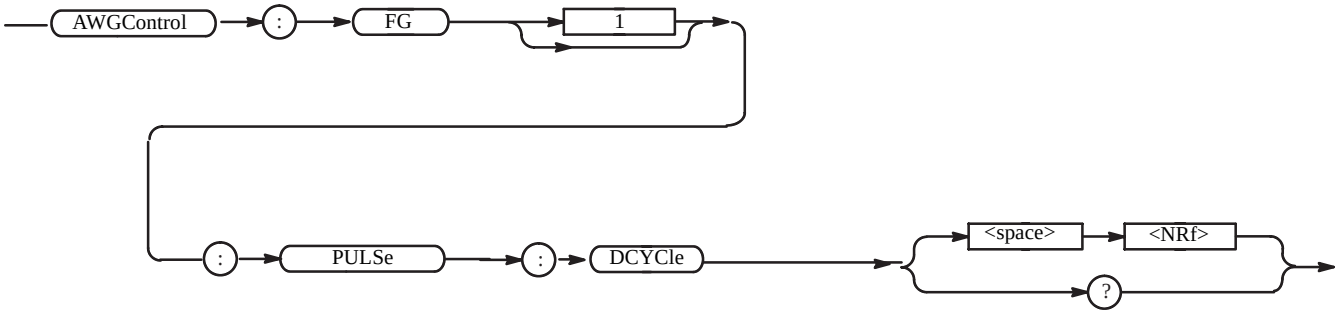
## :AWGControl:FG[1]:PULSe:DCYCle (?)

This command sets the duty cycle of the pulse waveform.
This query returns the duty cycle of the pulse waveform.

**Group** AWG Control

**Syntax** `:AWGControl:FG[1]:PULSe:DCYCle <numeric_value>`

`:AWGControl:FG[1]:PULSe:DCYCle?`




**Arguments** `<numeric_value>` is the duty cycle. The range is 0.1 to 99.9%.
Step:

| Frequency | Step(%) |
|---|---|
| 1.000Hz to 4.000MHz | 0.1 |
| 4.001MHz to 20.00MHz | 0.5 |
| 20.01MHz to 40.00MHz | 1 |
| 40.01MHz to 80.00MHz | 2 |
| 80.01MHz to 100.0MHz | 2.5 |
| 100.1MHz to 160.0MHz | 4 |
| 160.1MHz to 200.0MHz | 5 |
| 200.1MHz to 400.0MHz | 10 |

At `*RST`, this value is set as 10.0.

**Examples** `:AWGControl:FG1:PULSe:DCYCle 20`
sets the duty cycle to 20%.

## :AWGControl:FG[:STATe] (?)

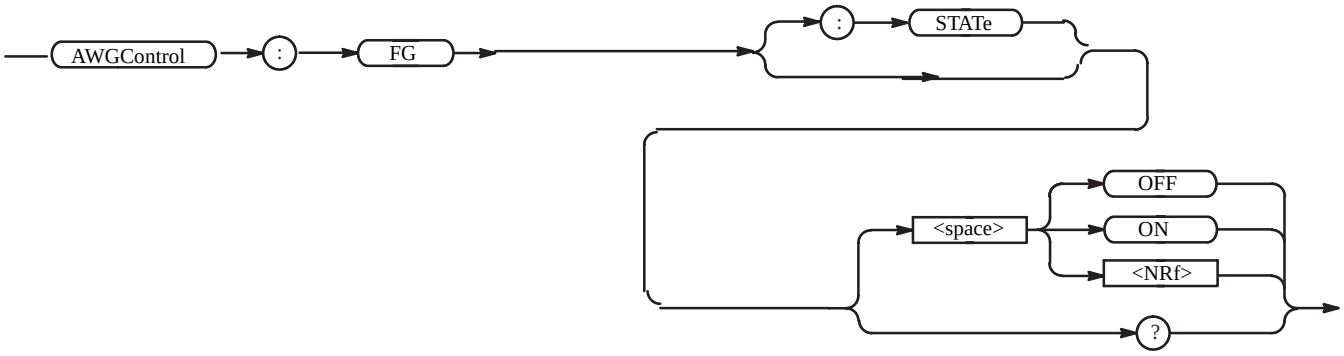
This command turns the FG(Function Generation) mode on or off.
This query returns status indicating whether the waveform generator is set to the function generator mode.

**Group** AWG Control

**Syntax**

```
AWGControl:FG[:STATe] <boolean>
```

```
AWGControl:FG[:STATe]?
```




**Arguments**

```
<boolean> ::={ OFF | ON | <NRf> }
OFF or 0 sets the FG mode to OFF .
ON or 1 sets the FG mode to  ON.
```

At *RST, this value is set to OFF.

**Examples**

```
:AWGControl:FG ON
```

sets the FG mode to ON.

# :AWGControl:FG[1]:VOLTage[:LEVel][:IMMediate][:AMPLitude] (?)

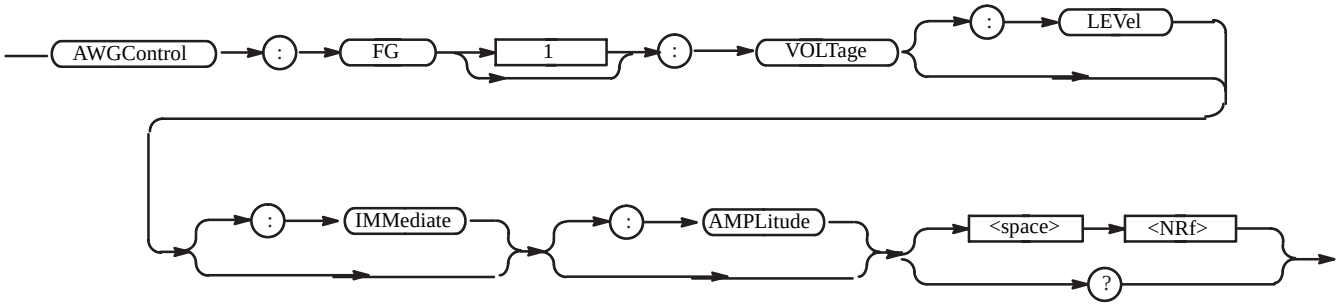
This command adjusts peak-to-peak voltage of the function waveform. This query returns peak-to-peak voltage currently set.

**Group** AWG Control

**Syntax**

```
:AWGControl:FG[1]:VOLTage[:LEVel][:IMMediate][:AMPLitude]
<numeric_value>
```

```
:AWGControl:FG[1]:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
```




**Arguments** <numeric_value> is the amplitude of the waveform. The Step is 1mV. The range is 0.020Vpp to 2.000Vpp
(In the case of Option02: 0.5Vpp to 1.0Vpp)

At *RST, this value is set to 1.0.

**Examples**

```
:AWGControl:FG1:VOLTage 2.0
```

sets the amplitude to 2.000Vpp.

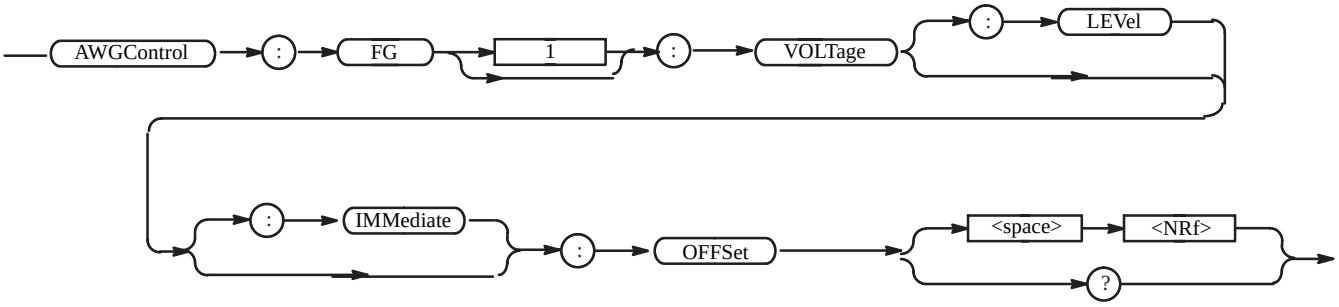
## :AWGControl:FG[1]:VOLTage[:LEVel][:IMMediate]:OFFSet (?) (except option02)

This command adjusts offset voltage of the function waveform.
This query returns offset voltage currently set.

**Group** AWG Control

**Syntax** `A:WGControl:FG[1]:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric_value>`

`:AWGControl:FG[1]:VOLTage[:LEVel][:IMMediate]:OFFSet?`




**Arguments** `<numeric_value>` is the offset of the waveform. The Step is 1mV.
The range is -0.500V to +0.500V

At `*RST`, this value is set to 0.0.

**Examples** `:AWGControl:FG1:VOLTage:OFFSet 0.1`
sets the offset to 0.1V.

## :AWGControl:MIX[:STATe] (?)

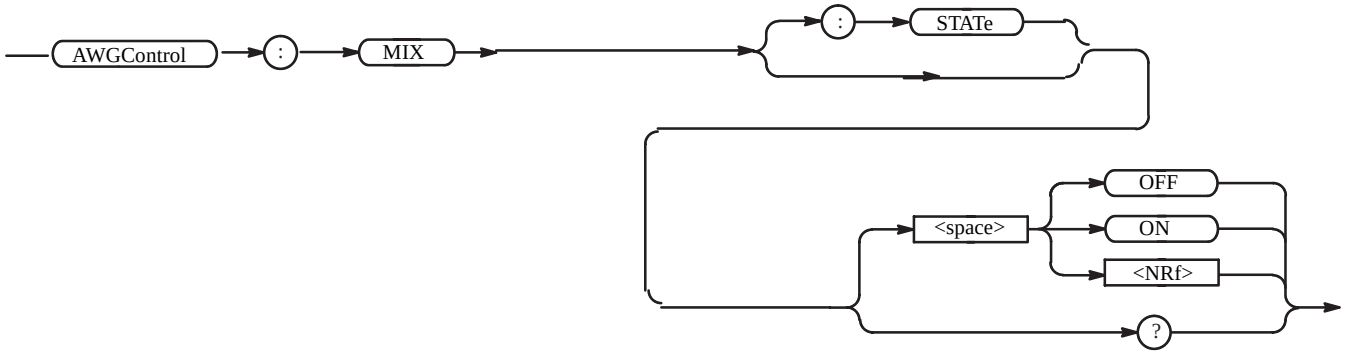
This command turns the Waveform Mixing mode on or off.
This query returns status indicating whether the waveform generator is set to the Waveform Mixing mode.

The command which sets up each parameter in Waveform Mixing mode is not supported.

**Group** AWG Control

**Syntax** `:AWGControl:MIX[:STATe] <boolean>`

`:AWGControl:MIX[:STATe]?`




**Arguments** `<boolean> ::={ OFF | ON | <NRf> }`

`OFF` or `0` sets the Waveform Mixing mode to OFF .
`ON` or `1` sets the Waveform Mixing mode to ON.

At `*RST`, this value is set to `OFF`.

**Examples** `:AWGControl:MIX ON`
sets the Waveform Mixing mode to ON.

## :AWGControl:RMODe (?)

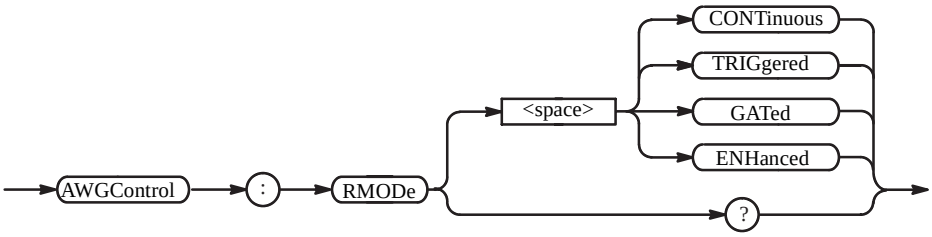
This command selects the mode used to output waveforms or sequences.

**Group** AWG Control

**Related Commands** `:AWGControl:RUN[:IMMediate], AWGControl:STOP[:IMMediate], [:SOURce[1]]:FUNCtion:USER, *TRG`

**Syntax** `:AWGControl:RMODe { CONTinuous | TRIGgered | GATed | ENHanced }`

`:AWGControl:RMODe?`



**Arguments** You can select the modes listed in Table 2-18.

**Table 2-18: Selecting run modes**

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

At *RST, this parameter is set to CONTinuous.

**Examples** SOURce:FUNCtion:USER "SAMPLE1.SEQ";:AWGControl:RMODe ENHanced;RUN outputs waveform according to the sequence file SAMPLE1.SEQ.

:AWGControl:RMODe?
can return the following response:
TRIG

## :AWGControl:RSTate? (Query Only)

This command returns the current running status.

**Group** AWG Control

**Syntax** :AWGControl:RSTate?


**Arguments** None

**Returns** <NR1>

0 The waveform generator is stopped.
1 The waveform generator is waiting for a trigger.
2 The waveform generator is running.

**Examples** :AWGControl:RSTate?
can return the following response:
1

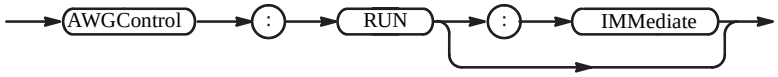
## :AWGControl:RUN[:IMMediate] (No Query Form)

This command initiates the output of a waveform or a sequence. This has the same effect as manually pressing the RUN button on the front panel.

**Group** AWG Control

**Related Commands** :AWGControl:STOP[:IMMediate], *TRG

**Syntax** :AWGControl:RUN[:IMMediate]




**Arguments** None

**Examples** :AWGControl:RUN[:IMMediate]
initiates the output of a waveform or a sequence.

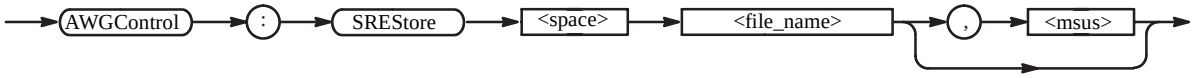
## :AWGControl:SREStore (No Query Form)

This command resets the default settings of a specified file.

**Group** AWG Control

**Related Commands** :AWGControl:SSAVe, :MMEMory:CDIRectory, :MMEMory:MSIS:

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




**Arguments** <file_name>::=<string> specifies the file to restore the settings.

<msus> (mass storage unit specifier)::=<string> is the media on which the file exists:

| | |
|---|---|
| MAIN | Internal hard disk drive |
| FLOPpy | Internal floppy disk drive |
| NET1, NET2, or NET3 | Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

**Examples** :AWGControl:SREStore "SAMPLE1.SET","FLOPpy"
resets the default settings of the file SAMPLE1.SET on the floppy disk.

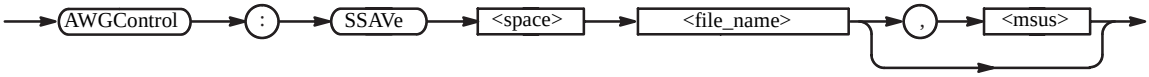
## :AWGControl:SSAVe (No Query Form)

This command stores the current settings to a specified file.

**Group** AWG Control

**Related Commands** :AWGControl:SREStore, :MMEMory:CDIRectory, :MMEMory:MSIS

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




**Arguments** `<file_name>::=<string>` specifies the file to store the settings.

`<msus>` (mass storage unit specifier)`::=<string>` is the media on which the file exists:

| | |
|---|---|
| `MAIN` | Internal hard disk drive |
| `FLOPpy` | Internal floppy disk drive |
| `NET1, NET2, or NET3` | Network drive 1, 2, or 3 (specified with the `SYSTem:COMMunicate:LAN` commands) |

**Examples** `:AWGControl:SSAVe "SAMPLE1.SET","FLOPpy"`
stores the current settings to the file SAMPLE1.SET on the floppy disk.

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

This command terminates waveform output. When the mode is not set to continuous, it also resets the sequence pointer to output the waveform from the top of the sequence with the next trigger event.

**Group** AWG Control

**Related Commands** `:AWGControl:RUN[:IMMediate], *TRG`

**Syntax** `:AWGControl:STOP[:IMMediate]`


**Arguments** None

**Examples** `:AWGControl:STOP[:IMMediate]`
stops the output of a waveform.

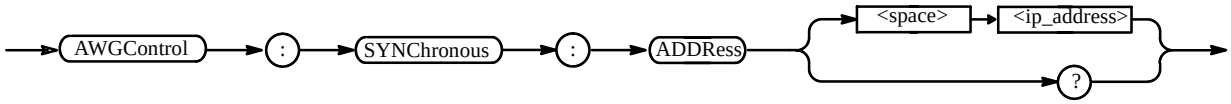
## :AWGControl:SYNChronous:ADDRess (?) (AWG710B only)

This command sets the IP address of the slave AWG710B for the Synchronous Operation.

**Group** AWG Control

**Syntax** `:AWGControl:SYNChronous:ADDRess <ip_address>`

`:AWGControl:SYNChronous:ADDRess?`




**Arguments** `<ip_address>::=<string>` is the IP address of the slave AWG710B.

`*RST` has no effect on the value.

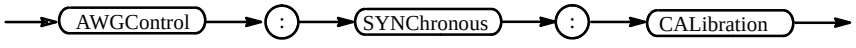
**Examples** `:AWGControl:SYNChronous:ADDRess "192.150.30.54"`
sets the IP address of the slave AWG710B to “192.150.30.54”.

## :AWGControl:SYNChronous:CALibration (No Query Form) (AWG710B only)

This command executes the Trigger Timing Calibration for the Synchronous Operation.

**Group** AWG Control

**Syntax** `:AWGControl:SYNChronous:CALibration`




**Examples** `:AWGControl:SYNChronous:CALibratioin`
executes the Trigger Timing Calibration.

## :AWGControl:SYNChronous:CONNect (?) (AWG710B only)

This command establishes a Master-Slave relationship of two waveform generators in the Synchronous Operation.

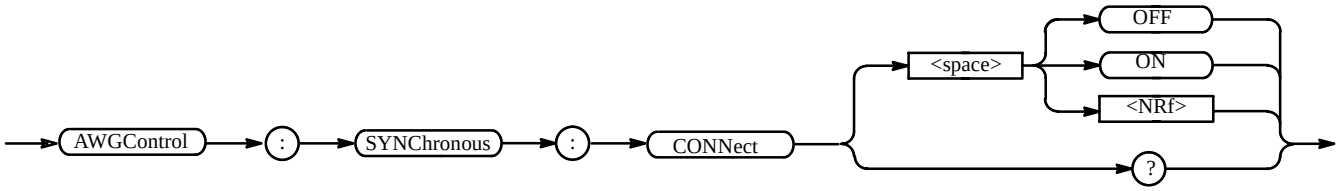
This query returns status indicating whether the communication of two waveform generators has been established.

**Group** AWG Control

**Syntax**

```
:AWGControl:SYNChronous:CONNect <boolean>
```

```
:AWGControl:SYNChronous:CONNect?
```




**Arguments**

```
<boolean> ::={ OFF | ON | <NRf> }
```

OFF or 0 sets the Waveform Mixing mode to OFF .

ON or 1 sets the Waveform Mixing mode to ON.

At *RST, this value is set to OFF.

**Examples**

```
:AWGControl:MIX ON
```

sets the Waveform Mixing mode to ON.

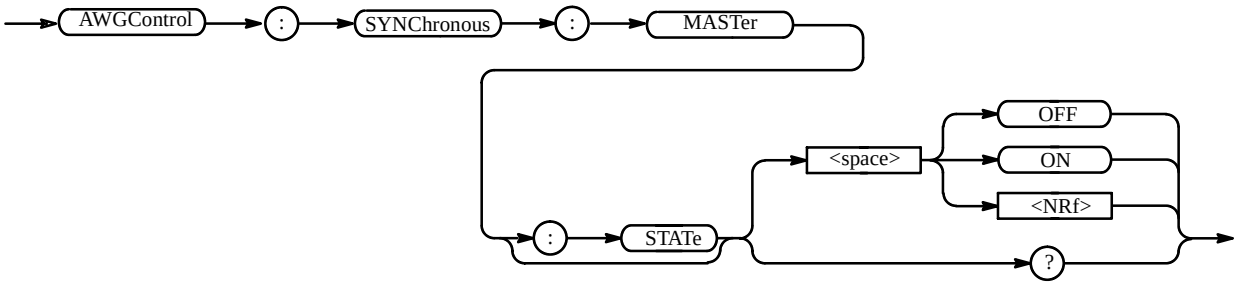
## :AWGControl:SYNChronous:MASTer[:STATe] (?) (AWG710B only)

This command turns the Synchronous Operation mode on or off, and sets the waveform generator as a master

**Group** AWG Control

**Syntax** `:AWGControl:SYNChronous:MASTer[:STATe] <boolean>`

`:AWGControl:SYNChronous:MASTer[:STATe]?`




**Arguments** `<boolean> ::={ OFF | ON | 0 | 1 }`

`OFF` or `0` sets the Synchronous Operation mode to OFF and an operation mode turns to normal operaion.

`ON` or `1` sets the Synchronous Operation mode to ON and the waveform generator as a master instrument.

At `*RST`, this value is set to `OFF`.

**Examples** `:AWGControl:SYNChronous:MASTer[:STATe] ON`
sets the Synchronous Operation to ON as a master.

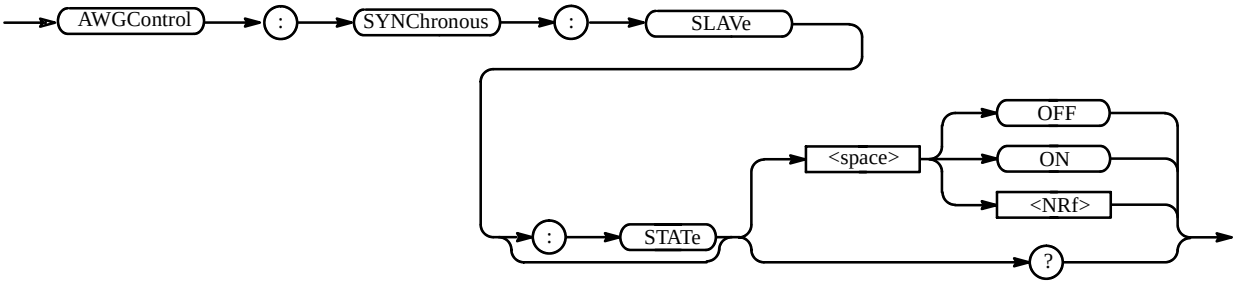
## :AWGControl:SYNChronous:SLAVe[:STATe] (?) (AWG710B only)

This command turns the Synchronous Operation mode on or off, and sets the waveform generator as a slave.

**Group** AWG Control

**Syntax** `:AWGControl:SYNChronous:SLAVe[:STATe] <boolean>`

`:AWGControl:SYNChronous:SLAVe[:STATe]?`




**Arguments** `<boolean> ::={ OFF | ON | 0 | 1 }`

`OFF` or `0` sets the Synchronous Operation mode to OFF and an operation mode turns to normal operaion.

`ON` or 1 sets the Synchronous Operation mode to ON and the waveform generator as a slave instrument.

At `*RST`, this value is set to `OFF`.

**Examples** `:AWGControl:SYNChronous:SLAVe[:STATe] ON`
sets the Synchronous Operation to ON as a slave.

## *CAL? (Query Only)

The *CAL? query performs an internal calibration and returns a status that indicates whether or not the waveform generator completed the calibration successfully. If an error is detected during calibration, execution immediately stops, and an error code is returned. This query performs the same function as the CALibration[:ALL]? query.

***NOTE.*** *A period of time is required to complete the internal calibration. During this time, the waveform generator does not respond to any commands or queries issued.*

**Group** Calibration

**Related Commands** :CALibration[:ALL]?

**Syntax** *CAL?


**Arguments** None

**Returns** <NR1>

0 Terminated without error.
-340 Calibration failed.

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

## :CALibration[:ALL] (?)

The :CALibration[:ALL] command performs a full calibration of the waveform generator.

The :CALibration[:ALL]? query performs a full calibration and responds with a <NR1> indicating the success of the calibration. This query has the same function as the *CAL? query.

If an error is detected during calibration, a message is queued in the error/event queue, and the error code "-340" is returned.

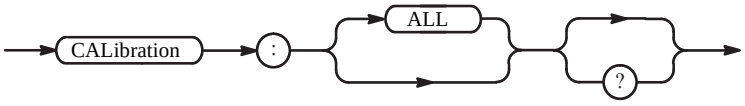
***NOTE.*** *A period of time is required to complete the internal calibration. During this time, the waveform generator does not respond to any commands or queries issued.*

**Group** Calibration

**Related Commands** `*CAL?`

**Syntax** `:CALibration[:ALL]`

`:CALibration[:ALL]?`




**Arguments** None

**Returns** `<NR1>`

0 Terminated without error.
-340 Calibration failed.

**Examples** `:CALibration[:ALL]`
performs a full calibration.

`:CALibration[:ALL]?`
performs a full calibration and returns the results. For example, it might return 0, which indicates the calibration terminated without any errors.

## *CLS (No Query Form)

This command clears all the event registers and queues, used by the waveform generator status and event reporting system. For more details, refer to the, *Status and Events* section.

**Group** Status

**Syntax** *CLS


**Arguments** None

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

## :DIAGnostic:DATA? (Query Only)

This command returns the results of a self–test.

**Group** Diagnostic

**Related Commands** :DIAGnostic[:IMMediate], :DIAGnostic:SELect

**Syntax** :DIAGnostic:DATA?


**Arguments** None

**Returns** <NR1>

0 Terminated without error.
-330 Self–test failed.

**Examples** :DIAGnostic:DATA?
might return 0.

## :DIAGnostic[:IMMediate] (?)

The :DIAGnostic[:IMMediate] command executes the self–test routine(s) selected by the :DIAGnostic:SELect command. The query :DIAGnostic[:IMMediate]? executes the routine(s) and returns the results.

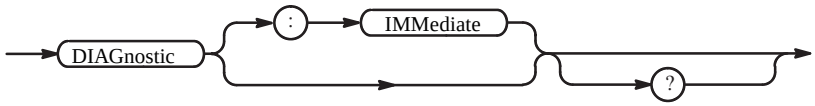
If an error is detected during execution, the routine that detected the error terminates. If all of the self–test routines are selected, self–testing continues with execution of the next self–test routine.

**Group** Diagnostic

**Related Commands** :DIAGnostic:SELect, :DIAGnostic:DATA?

**Syntax** :DIAGnostic[:IMMediate]

:DIAGnostic[:IMMediate]?




**Arguments** None

**Returns** <NR1>

0 Terminated without error.
-330 Self–test failed.

**Examples** :DIAGnostic:SELect ALL;IMMediate?
executes all of the self–test routines. After all self–test routines finish, the results of the self–tests are returned.

## :DIAGnostic:SELect (?)

This command selects the self–test routine(s).

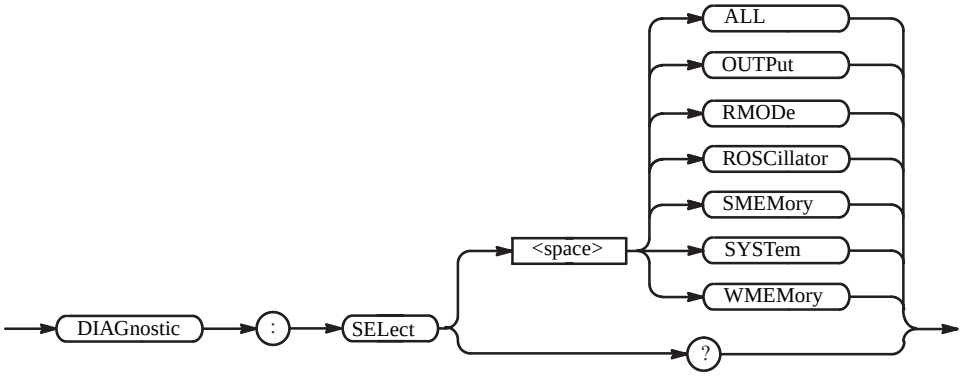
**Group** Diagnostic

**Related Commands** :DIAGnostic[:IMMediate]

**Syntax**

```
:DIAGnostic:SELect { ALL | OUTPut | RMODe | ROSCillator
| SMEMory | SYSTem | WMEMory }

:DIAGnostic:SELect?
```




**Arguments** You can select the following self–test routines:

**Table 2-19: Self–test routines**

| Argument | Description |
|---|---|
| ALL | Checks all routines that follow |
| OUTput | Checks the analog output unit |
| RMODe | Checks the control unit |
| ROSCillator | Checks the reference oscillator unit |
| SMEMory | Checks the sequence memory |
| SYSTem | Checks the system unit, such as the system memory |
| WMEMory | Checks the waveform memory |

At *RST, this parameter is set to ALL.

**Examples**

```
:DIAGnostic:SELect WMEMory;IMMediate
```

executes the waveform memory self–test routine.

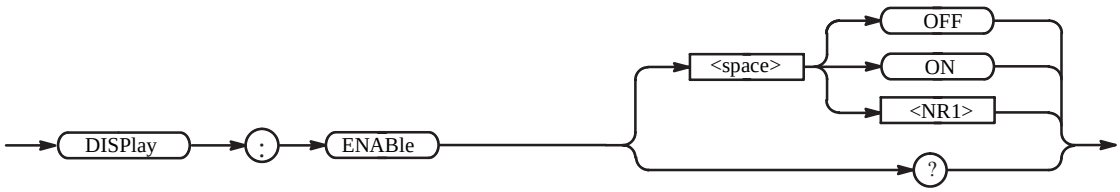
## :DISPlay:ENABle (?)

This command controls ON/OFF of the display.

**Group** Display

**Syntax** `:DISPlay:ENABle <boolean>`

`:DISPlay:ENABle?`




**Arguments** `<boolean> ::={ OFF | ON | 0 | 1 }`

`OFF` or `0` sets OFF the display.
`ON` or `1` sets ON the display.

At `*RST`, this value is set to `OFF`.

**Examples** `:DISPlay:ENABle ON`
sets ON the display.

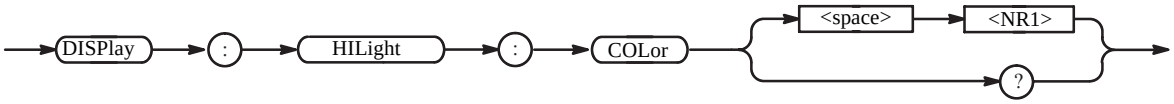
## :DISPlay:HILight:COLor (?)

This command controls the HILight color.

**Group** Display

**Syntax** :DISPlay:HILight:COLor <NR1>

:DISPlay:HILight:COLor?




**Arguments** <NR1> is the color number. The range is 0 to 7.

At *RST, this value is set to 0.

**Returns** <NR1> indicates the number of color.

**Examples** :DISPlay:HILight:COLor 1
sets the number 1 color.

## *ESE (?)

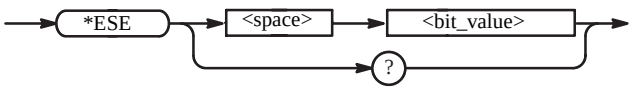
The *ESE command sets the bits of the ESER (Event Status Enable Register) used in the status and events reporting system of the waveform generator. The *ESE? query returns the contents of the ESER. Refer to the *Status and Events* for more information about the ESER.

**Group** Status

**Related Commands** *CLS, *ESR?, *PSC, *SRE, *STB?

**Syntax** *ESE <bit_value>

*ESE?



**Arguments**

<bit_value>::=<NR1>
where <NR1> is a decimal integer in the range 0 to 255. The binary bits of the ESER are set according to this value.

The power–on default for ESER is 0 if *PSC is 1. If *PSC is 0, the ESER maintains its value through a power cycle.

**Examples**

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

*ESE?
might return 176, which indicates that the ESER contains the binary number 10110000.

## *ESR? (Query Only)

This command returns the contents of the Standard Event Status Register (SESR) used in the status and events reporting system in the waveform generator. *ESR? also clears the SESR (since reading the SESR clears it). Refer to Section 3 *Status and Events* for more information.

**Group**

Status

**Related Commands**

*CLS, *ESE?, *SRE, *STB?

**Syntax**

*ESR?


**Returns**

<NR1> indicates the content of the SESR in a decimal integer.

**Examples**

*ESR?
might return 181, which indicates that the SESR contains the binary number 10110101.

## :HCOPy:DESTination (No Query Form)

This command sets the hardcopy destination. For the waveform generator, the destination is always set to MMEMory (mass memory). This command is included only for compatibility with the SCPI standard. The destination file on the mass memory device is specified by the :MMEMory:NAME command. For more information about hardcopy, see *Hardcopy Commands* on page 2-18.

**Group** Hardcopy

**Related Commands** :MMEMory:NAME

**Syntax** :HCOPy:DESTination <data_handle>


**Arguments** <data_handle>::=<string>
where <string> is fixed to "MMEMory" for the waveform generator.

**Examples** :HCOPy:DESTination "MMEMory"
sets the hardcopy destination to a file specified with the MMEMory:NAME command.

## :HCOPy:DEVice:COLor (?)

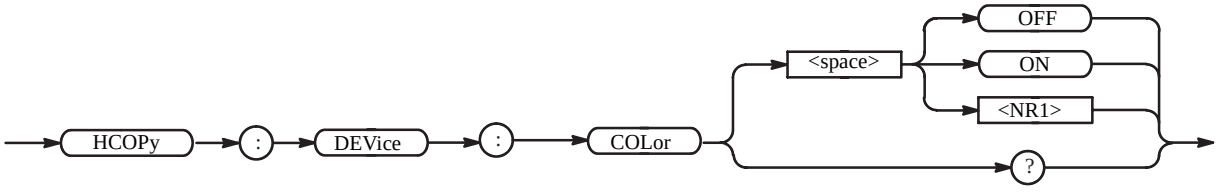
This command sets the hardcopy color mode.

**Group** Hardcopy

**Related Commands** :HCOPy:DEVice:LANGuage

**Syntax** :HCOPy:DEVice:COLor { OFF | ON | <NRf> }

:HCOPy:DEVice:COLor?




**Arguments** OFF or <NRf>=0 sets the hardcopy color mode to OFF.
ON or <NRf>≠0 sets the hardcopy color mode to ON.

At *RST, this value is set to OFF.

**Returns** ON : when hardcopy color mode is set to ON.
OFFN : when hardcopy color mode is set to OFF

**Examples** :HCOPy:DEVice:COLor ON
sets the hardcopy color mode to ON.

## :HCOPy:DEVice:LANGuage (?)

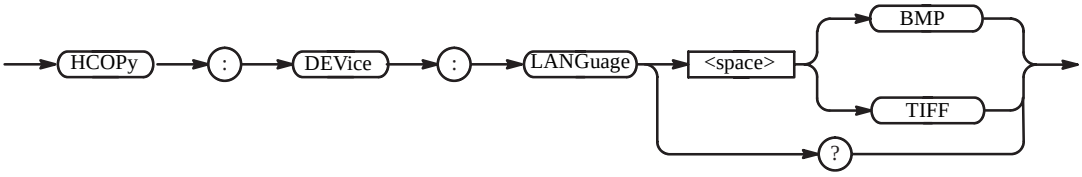
This command sets the hardcopy data format.

**Group** Hardcopy

**Related Commands** :HCOPy:DEVice:COLor

**Syntax** :HCOPy:DEVice:LANGuage { BMP | TIFF }

:HCOPy:DEVice:LANGuage?




**Arguments** BMP specifies the Windows bitmap file format.

TIFF specifies the TIFF format.

At *RST, the parameter is set to BMP.

**Examples** :HCOPy:DEVice:LANGuage TIFF
specifies the TIFF data format for hardcopy.

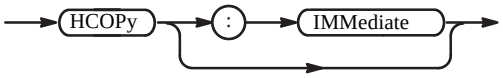
## :HCOPy[:IMMediate] (No Query Form)

This command immediately initiates hardcopy output according to the current HCOPy setup parameters. For the waveform generator, this command is the same as `:HCOPy:SDUMp[:IMMediate]`. For more information about hardcopy, see *Hardcopy Commands* on page 2-18.

**Group** Hardcopy

**Related Commands** `:HCOPy:DESTination`, `:HCOPy:SDUMp[:IMMediate]`

**Syntax** `:HCOPy[:IMMediate]`




**Arguments** None

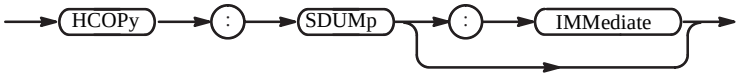
**Examples** `:HCOPy:IMMediate`
starts hardcopy output.

## :HCOPy:SDUMp[:IMMediate] (No Query Form)

This command initiates a screen dump of the entire screen. For the waveform generator, this is the same as the `:HCOPy[:IMMediate]` command. For more information about hardcopy, see *Hardcopy Commands* on page 2-18.

**Group** Hardcopy

**Syntax** `:HCOPy:SDUMp[:IMMediate]`




**Arguments** None

**Examples** `:MMEMory:NAME "SAMPLE1.BMP";:HCOPy:SDUMp:IMMediate`
prints the entire screen to the file SAMPLE1.BMP.

## *IDN? (Query Only)

This command returns identification information for the waveform generator.

**Group** System

**Syntax** *IDN?


**Arguments** None

**Returns** <manufacturer>, <model>, <serial_number>, <firmware_level>
where
<manufacturer>::=SONY/TEK (AWG710), TEKTRONIX (AWG710B)
<model>::= AWG710 (AWG710), AWG710B (AWG710B)
<serial_number>::=0
<firmware_level>::=SCPI:99.0 OS:x.y USR:x.y

**Examples** *IDN?
might return SONY/TEK,AWG710B,0,SCPI:99.0 OS:1.2 USR:4.0

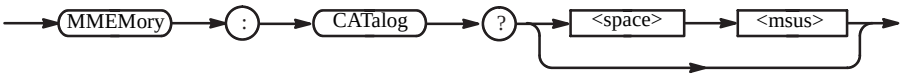
## :MMEMory:CATalog? (Query Only)

This command returns information about the current contents and state of the mass storage media.

**Group** Mass Memory

**Related Commands** :MEMory:CDIRectory, :MMEMory:MSIS

**Syntax** :MMEMory:CATalog?[ <msus>]




**Arguments** <msus> (mass storage unit specifier)::=<string> is one of the following:

| | |
|---|---|
| MAIN | The internal hard disk drive |
| FLOPpy | The internal floppy disk drive |
| NET1, NET2, or NET3 | The network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

**Returns**

```
<NR1>,<NR1>[,<file_name>,<file_type>,<file_size>]...
where:
The first <NR1> is the total amount of storage currently used, in
bytes.
For the network drives, <NR1>=0.
The second <NR1> is the total amount of storage available.
For the network drives, <NR1>=0.

<file_name>,<file_type>,<file_size>::=<string>
where
<file_name> is the exact name of a file,
<file_type> is DIR for directory, otherwise it is blank, and
<file_size> is the size of the file, in bytes.
```

**Examples**

```
:MMEMory:CATalog? "MAIN"
```

might return the following response:

```
484672,3878652,"SAMPLE1.WFM,2948"
```

## :MMEMory:CDIRectory (?)

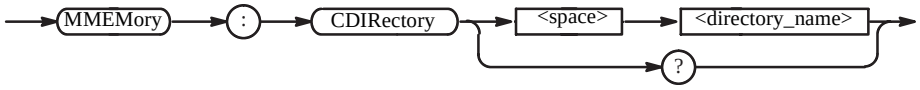
This command changes the default directory for a mass memory file system. The default mass storage device is selected by :MMEMory:MSIS command.

**Group** Mass Memory

**Related Commands** :MMEMory:CDIRectory, :MMEMory:MSIS

**Syntax**

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




**Arguments** <directory_name>::=<string>
is the default directory for a mass memory file system.

If you do not specify a parameter, the directory is set to the *RST value.

At *RST, this parameter is set to the root.

**Examples** :MMEMory:CDIRectory "/AWG/WORK0"
changes the default directory to /AWG/WORK0.

## :MMEMory:CLOSe (No Query Form)

This command closes the file specified in the MMEMory:NAME command. This command is included only for compatibility with the SCPI standard and may not be used.

**Group** Mass Memory

**Related Commands** :MMEMory:NAME, :MMEMory:OPEN

**Syntax** :MMEMory:CLOSe


**Arguments** None

**Examples** :MMEMory:NAME "SAMPLE1.WFM";CLOSe
closes the file SAMPLE1.WFM.

## :MMEMory:COPY (No Query Form)

This command copies an existing file to a new file. An error is generated if the source file does not exist.

**Group** Mass Memory

**Related Commands** :MMEMory:CDIRectory, :MMEMory:DELete, :MMEMory:MSIS

**Syntax** :MMEMory:COPY <file_source>,<file_destination>


**Arguments** <file_source>::=<file_name>[,<msus>]
<file_destination>::=<file_name>[,<msus>]

where:
<file_name>::=<string> is the source or destination file name.

<msus> (mass storage unit specifier)::=<string> is the media on which the file exists:

| | |
|---|---|
| MAIN | Internal hard disk drive |
| FLOPpy | Internal floppy disk drive |
| NET1, NET2, or NET3 | Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

**Examples** :MMEMory:COPY "FILE1.WFM","MAIN","FILE2.WFM","FLOPpy"
copies the file FILE1.WFM on the waveform generator hard disk to the file FILE2.WFM on the floppy disk.

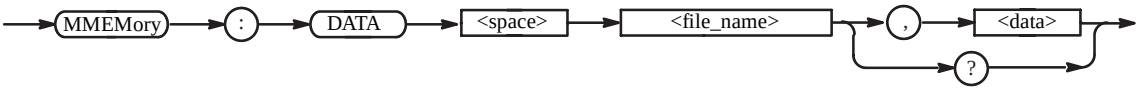
## :MMEMory:DATA (?)

This command loads block data into the file on the default mass storage device, or returns the contents of the file.

**Group** Mass Memory

**Related Commands** :MMEMory:CDIRectory, :MMEMory:MSIS

**Syntax** :MMEMory:DATA <file_name>,<data>
:MMEMory:DATA <file_name>?




**Arguments** `<file_name>::=<string>` specifies the file to be loaded with data.

`<data>` is in 488.2 block format.

**Examples** `:MMEMory:DATA "FILE1",#41024xxxxx...`
loads data into the file FILE1.

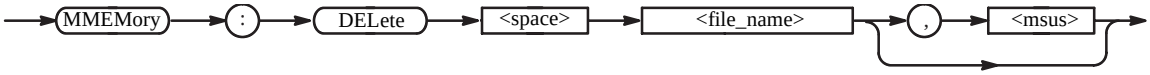
## :MMEMory:DELete (No Query Form)

This command removes a file from the specified mass storage device.

**Group** Mass Memory

**Related Commands** `:MMEMory:CDIRectory, MMEMory:MSIS`

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




**Arguments** `<file_name>::=<string>` specifies the file to be removed.

`<msus>` (mass storage unit specifier)`::=<string>` is the media on which the file exists:

| | |
|---|---|
| `MAIN` | Internal hard disk drive |
| `FLOPpy` | Internal floppy disk drive |
| `NET1, NET2, or NET3` | Network drive 1, 2, or 3 (specified with the `SYSTem:COMMunicate:LAN` commands) |

**Examples** `:MMEMory:DELete "FILE1.WFM","FLOPpy"`
removes the file FILE1.WFM on the floppy disk.

# :MMEMory:FEED (?)

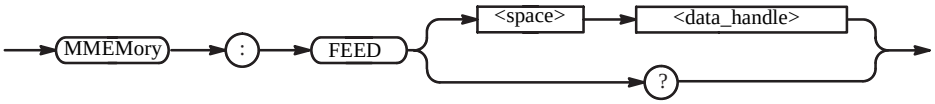
This command sets the data handle to be used to feed data into the file specified by :MMEMory:NAME. For the waveform generator, the data handle is fixed to HCOPy. This command is included only for compatibility with the SCPI standard, and may not be used (refer to *Hardcopy Commands* on page 2-18).

**Group** Mass Memory

**Related Commands** MMEMory:NAME

**Syntax** :MMEMory:FEED <data_handle>

MMEMory:FEED?




**Arguments** <data_handle>::=<string> for the waveform generator, the data handle is fixed to HCOPy.

At *RST, this parameter is set to ”HCOP”.

**Examples** :MMEMory:FEED "HCOPy"
sets the data handle.

## :MMEMory:INITialize (No Query Form)

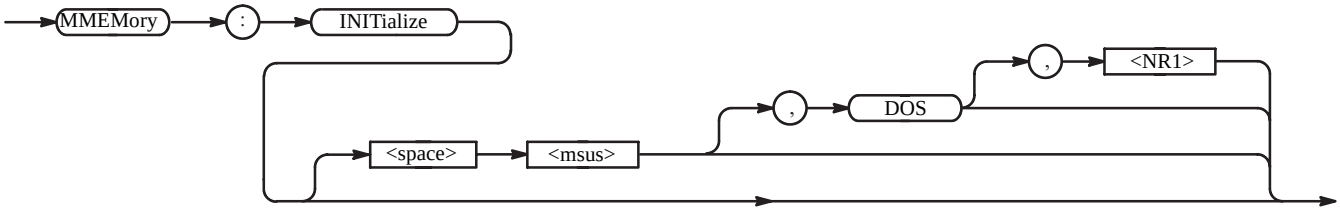
This command initializes a specified mass storage media. In this application, you can initialize the internal hard disk or floppy disk.

***NOTE.*** *The initializing process erases all information that is already on the disk. This command is a "Quick Format" command, which cannot format MAC format, other OS format and damaged disk.*

**Group** Mass Memory

**Related Commands** `:MMEMory:MSIS`

**Syntax** `:MMEMory:INITialize[ <msus>[,DOS[,<NR1>]]]`




**Arguments** `<msus>` (mass storage unit specifier)`::=<string>` is the media containing the specified mass storage:`{ "MAIN" | "FLOPpy" }`

where `MAIN` means the internal hard disk, and `FLOPpy` means the floppy disk.

The media is initialized in DOS format.

`<NR1>` is ignored in this application (It usually specifies media–dependent information).

When you specify `MAIN`, this command returns the instrument settings to the factory defaults, except for the communication parameters (see *Appendix E: Factory Initialization Settings*).

**Examples** `:MME Mory:INITialize "FLOPpy"`
initializes a floppy disk in DOS format.

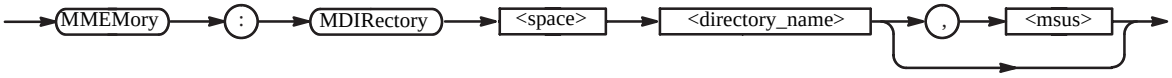
## :MMEMory:MDIRectory (No Query Form)

This command creates a directory on the specified mass storage unit.

**Group** Mass Memory

**Related Commands** :MMEMory:CDIRectory, :MMEMory:MSIS

**Syntax** :MMEMory:MDIRectory <directory_name>[,<msus>]




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

<msus> (mass storage unit specifier)::=<string> is the media on which you make the directory:

| | |
|---|---|
| MAIN | Internal hard disk drive |
| FLOPpy | Internal floppy disk drive |
| NET1, NET2, or NET3 | Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

**Examples** :MMEMory:MDIRectory "WAVEFORM","FLOPpy"
makes the directory “WAVEFORM” on the floppy disk.

## :MMEMory:MOVE (No Query Form)

This command moves an existing file to another file name. If the source file does not exist, an error occurs.

**Group** Mass Memory

**Related Commands** :MMEMory:CDIRectory, :MMEMory:COPY, :MMEMory:DELete, :MMEMory:MSIS

**Syntax** :MMEMory:MOVE <file_source>,<file_destination>


**Arguments** <file_source>, <file_destination>
::=<file_name>[,<msus>]

where:

<file_name>::=<string> is the source or destination file name.

<msus> (mass storage unit specifier)::=<string> is the media on which the file exists:

| | |
|---|---|
| MAIN | Internal hard disk drive |
| FLOPpy | Internal floppy disk drive |
| NET1, NET2, or NET3 | Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

**Examples** :MMEMory:MOVE "FILE1.WFM","MAIN","FILE2.WFM","FLOPpy"
moves the file FILE1.WMF on the waveform generator hard disk to FILE2.WFM on the floppy disk.

## :MMEMory:MSIS (?)

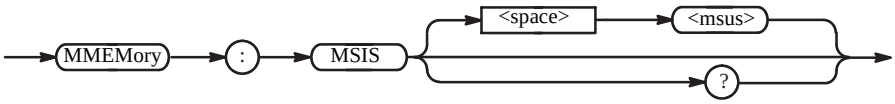
The “Mass Storage IS” command selects a default mass storage device that is used by all MMEMory commands except INITialize.

**Group** Mass Memory

**Related Commands** All MMEMory commands except INITialize.

**Syntax** :MMEMory:MSIS[ <msus>]

:MMEMory:MSIS?




**Arguments** <msus>(Mass Storage Unit Specifier)::=<string> specifies a default mass storage device.

| | |
|---|---|
| MAIN | Internal hard disk drive |
| FLOPpy | Internal floppy disk drive |
| NET1, NET2, or NET3 | Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

At *RST, this parameter is set to MAIN.

**Examples** :MMEMory:MSIS "FLOPpy"
selects the floppy disk drive as the default mass storage device.

## :MMEMory:NAME (?)

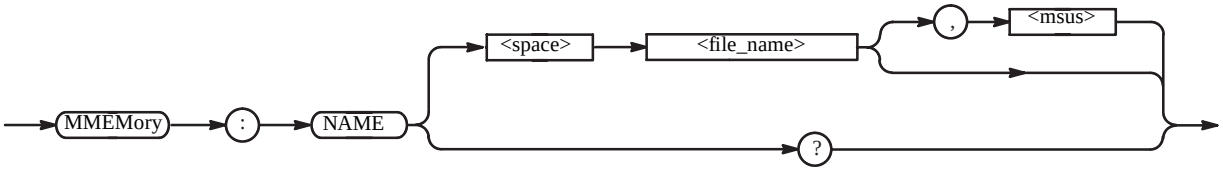
This command specifies the name of the file specification used by MMEMory:OPEN or CLOSe commands.

**Group** Mass Memory

**Related Commands** :MMEMory:CLOSe, :MMEMory:OPEN

**Syntax** :MMEMory:NAME <file_name>[,<msus>]

:MMEMory:NAME?




**Arguments** `<file_name>::=<string>` is the name of the file to be opened or closed.

`<msus>` (mass storage unit specifier)`::=<string>` is the media on which the file exists:

| | |
|---|---|
| `MAIN` | Internal hard disk drive |
| `FLOPpy` | Internal floppy disk drive |
| `NET1, NET2, or NET3` | Network drive 1, 2, or 3 (specified with the `SYSTem:COMMunicate:LAN` commands) |

At `*RST`, this parameter is set to ”`HARDCOPY`”.

**Examples** `:MMEMory:NAME "SAMPLE1.WFM","NET1";OPEN`
opens the file SAMPLE1.WFM on the network drive 1.

## :MMEMory:OPEN (No Query Form)

This command opens the file specified in the `MMEMory:NAME` command. This command is included only for compatibility, and may not be used.

**Group** Mass Memory

**Related Commands** `:MMEMory:CDIRectory, :MMEMory:CLOSe, :MMEMory:MSIS, :MMEMory:NAME`

**Syntax** `:MMEMory:OPEN`


**Arguments** None

**Examples** `:MMEMory:NAME "SAMPLE1.WFM","NET1";OPEN`
opens the file SAMPLE1.WFM on the network drive 1.

## *OPC (?)

Operation complete command (query). Use this command between two other commands to ensure completion of the first command before processing the second command.

In this application, all commands are designed to be executed in the order in which they are sent from the external controller. The *OPC (?) command is included to ensure compliance with the SCPI standard. You do not need to use this command.

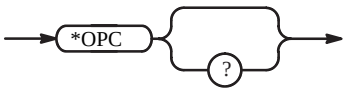
Refer to page 3-6 about the OPC bit of SESR (Standard Event Status Register).

**Group** Synchronization

**Related Commands** *WAI

**Syntax** *OPC

*OPC?




**Arguments** None

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

## *OPT? (Query Only)

This command returns the implemented options of the waveform generator.

**Group** System

**Syntax** *OPT?


**Arguments** None

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

where:

| | |
|---|---|
| 0 | the waveform generator has no options installed. |
| 01 | the waveform generator has Long Memory Option installed. |
| 02 | the waveform generator has High Bandwidth Output Option installed |
| 10 | the waveform generator has ATE Option installed. |

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

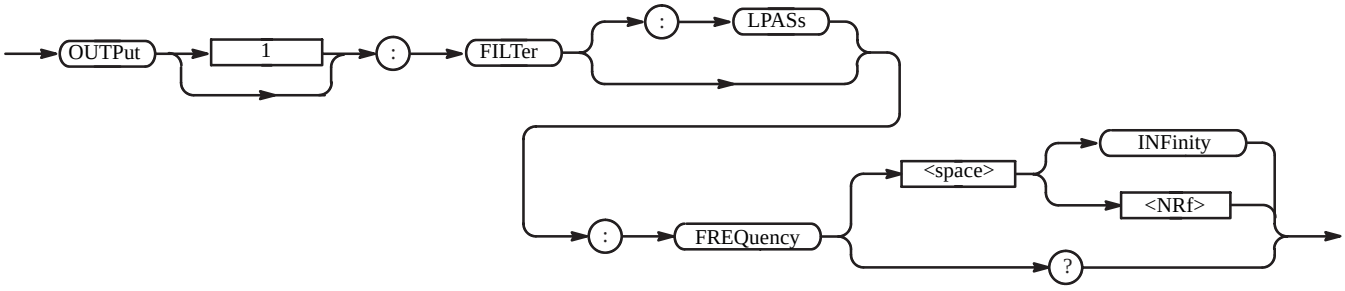
## :OUTPut[1]:FILTer[:LPASs]:FREQuency (?) (except option02)

This command determines the cutoff frequency of the low pass filter for a specified channnel.

**Group** Output

**Syntax** `:OUTPut[1]:FILTer[:LPASs]:FREQuency { <numeric_value> | INFinity }`

`:OUTPut[1]:FILTer[:LPASs]:FREQuency?`




**Arguments** `<numeric_value>` is the cutoff frequency of the low pass filter, in Hz. You can select 20e6 (20MHz), 50e6 (50MHz), 100e6 (100MHz), 200e6 (200MHz), or 9.9e37 (INFinity, that means “through”).

At `*RST`, this value is set to `9.9e37` (“through”).

**Examples** `:OUTPut1:FILTer:LPASs:FREQuency 100MHZ`
sets the cutoff frequency of the low pass filter for CH 1 to 100 MHz.

At `*RST`, this value is set to `9.9e37` (“through”).

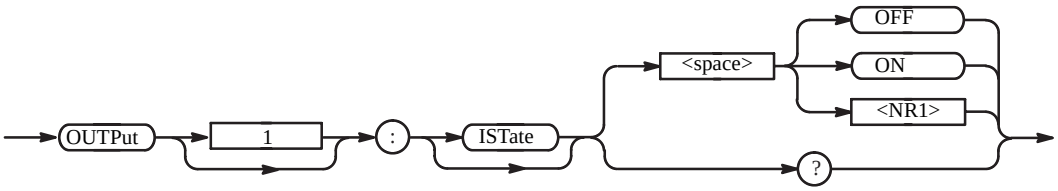
## :OUTPut[1]:ISTate (?)

Controls whether the waveform generator inverted CH 1 output terminal ($\overline{CH1}$) is enabled or disabled. When the function is OFF, the $\overline{CH1}$ terminal is at maximum isolation from the signal.

**Group** Output

**Syntax** `:OUTPut[1]:ISTate <boolean>`

`:OUTPut[1]:ISTate?`




**Arguments** `<boolean>::={ ON | OFF | 1 | 0 }`

`<ON>` or `1` enables the $\overline{CH1}$ output.

`<OFF>` or `0` disables the $\overline{CH1}$ output.

At `*RST`, this value is set to 0 (OFF).

**Examples** `:OUTPUT1:ISTate ON`
enables the $\overline{CH1}$ output.

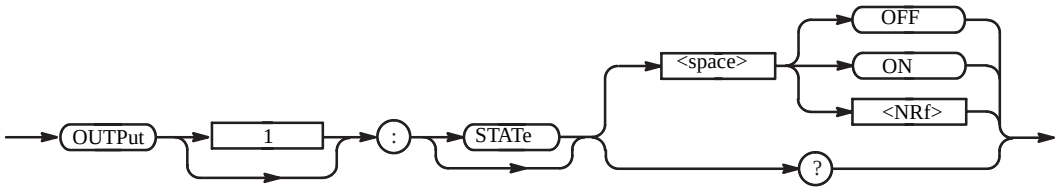
## :OUTPut[1][:STATe] (?)

This command controls whether the output terminal is open or close. When the function is OFF, the terminal is at maximum isolation from the signal.

**Group** Output

**Syntax** `:OUTPut[1][:STATe] <boolean>`

`:OUTPut[1][:STATe]?`




**Arguments** `<boolean>::={ ON | OFF | 1 | 0 }`

`<ON> or 1` turns the output on.
`<OFF> or 0` turns the output off.

At `*RST`, this value is set to 0 (OFF).

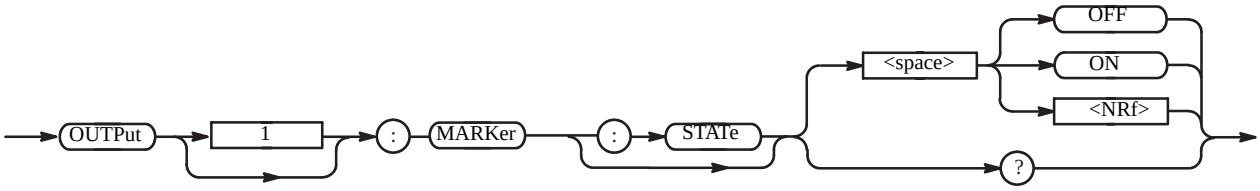
**Examples** `:OUTPut1:STATe ON`
turns the CH 1 output on.

## :OUTPut[1]:MARKer[:STATe] (?) (AWG710B only)

This command controls whether the marker output awitch is on or off. When the function is OFF, the marker output terminal is at maximum isolation from the signal.

**Group** Output

**Syntax** `:OUTPut[1]:MARKer[:STATe] <boolean>`
`:OUTPut[1]MARKer[:STATe]?`




**Arguments** `<boolean>::={ ON | OFF | 1 | 0 }`
`<ON> or 1 turns the output on.`
`<OFF> or 0 turns the output off.`

At `*RST`, this value is set to 0 (OFF).

**Examples** `:OUTPut:MARKer:STATe ON`
turns all the MARKER OUT on.

# *PSC (?)

This command sets and queries the power–on status flag that controls the automatic power–on handling of the SRER, ESER, OENR, and QENR registers. When *PSC is true, the registers are set to 0 at power–on. When *PSC is false, the current values in the registers are preserved in nonvolatile memory when power is shut off, and then are restored at power–on. For a complete discussion of the use of these registers, refer to the *Status and Event Reporting* section on page 3-1.

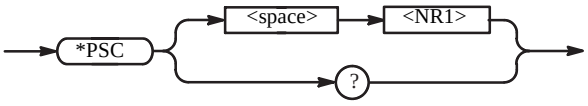
**Group** Status

**Related Commands** *ESE, *SRE, :STATus:OPERation:ENABle, :STATus:QUEStionable:ENABle

**Syntax**

```
*PSC <NR1>

*PSC?
```




**Arguments** <NR1>=0 sets the power–on status clear flag to false, disables the power–on clear, and allows the waveform generator 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. Using an out–of–range value causes an execution error.

**Examples**

```
*PSC 0
```

sets the power–on status clear flag to false.

```
*PSC?
```

might return the value 1, showing that the power–on status clear flag is set to true.

## *RST (No Query Form)

This command resets the waveform generator to the default state. This command has no effect on the network and communication settings, such as GPIB or IP address. Refer to *Appendix E: Factory Initilization Settings*.

**Group** System

**Related Commands** `:SYSTem:SECurity:IMMediate`

**Syntax** `*RST`


**Arguments** None

**Examples** `*RST`
resets the instrument.

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

This command sets the sampling frequency to output a waveform or pattern file. The file is specified by the `:SOURce[1]:FUNCtion:USER` command.

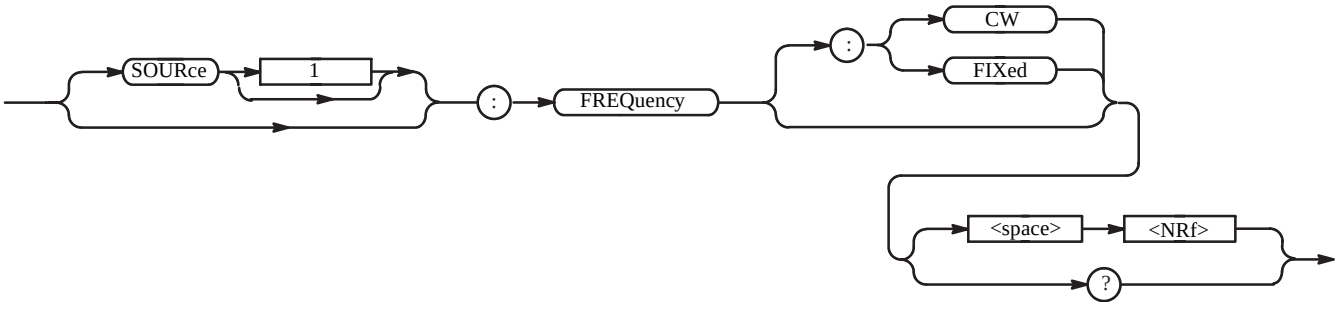
CW (Continuous Wave) and FIXed are aliases, and have the same effect.

**Group** Source

**Related Commands** `[:SOURce[1]]:FUNCtion:USER`

**Syntax** `[:SOURce[1]]:FREQuency[:CW|:FIXed] <numeric_value>`

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



**Arguments** <numeric_value> is the sampling frequency. The range is as follows.

AWG710: 50 kHz to 4.0 GHz
AWG710B: 50 kHz to 4.2 GHz

At *RST, this value is set to 100 MHz.

**Examples** :SOURce1:FREQuency 10MHz
sets the sampling frequency to 10 MHz.

## [:SOURce[1]]:FUNCtion:USER (?)

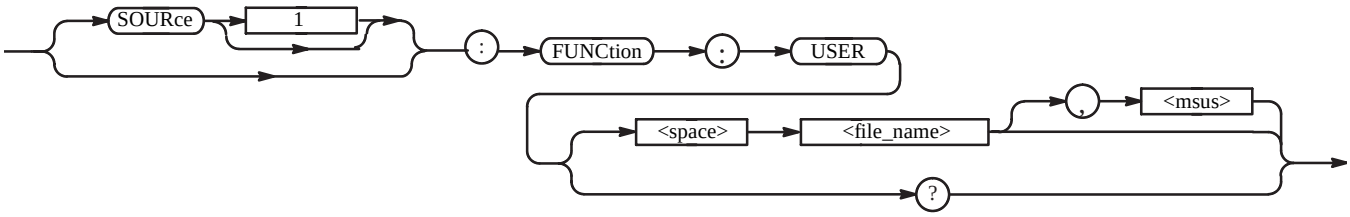
This command specifies a waveform or pattern file that you created as the output source. This command loads the file into the waveform generator's RAM prior to output.

**Group** Source

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

**Syntax** [:SOURce[1]]:FUNCtion:USER <file_name>[,<msus>]

[:SOURce[1]]:FUNCtion:USER?




**Arguments** <file_name>::=<string> is the name of a waveform or pattern file to output.

<msus> (mass storage unit specifier) ::=<string> is the media on which the file exists:

| | |
|---|---|
| MAIN | The internal hard disk drive |
| FLOPpy | The internal floppy disk drive |
| NET1, NET2, or NET3 | The network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands) |

At *RST, this value is set to "" (null).

**Examples** :SOURce1:FUNCtion:USER "SAMPLE1.WFM","FLOPpy"
specifies the file SAMPLE1.WFM on the floppy disk as the CH 1 output source.

## [:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH (?)

This command sets the high level for the marker output.

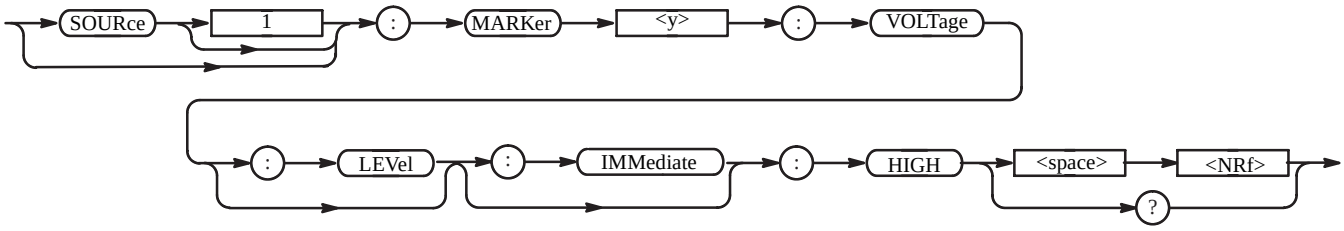
**Group** Source

**Related Commands** `[:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW`

**Syntax**

```
[:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:HIGH
<numeric_value>
```

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




**Arguments** `<numeric_value>` is the high level voltage of the marker output. Note that the high level must be larger than the low level. The range is as follows:

AWG710:
-1.1 V to 3.0 V (into 50 Ω) with a resolution of 0.05 V. Note that the diference between high and low level is restricted to within 2.5 V.
At `*RST`, this value is set to 2 V.

AWG710B:
-1.00V to 2.45 V (into 50 Ω) with a resolution of 0.05 V. Note that the diference between high and low level is restricted to within 1.25 V.
At `*RST`, this value is set to 1 V.

**Examples**

```
:SOURce1:MARKer1:VOLTage:LEVel:IMMediate:HIGH 1.2
```

sets the high level of the marker 1 output on CH 1 to 1.2 V.

## [:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVel][:IMMediate]:LOW (?)

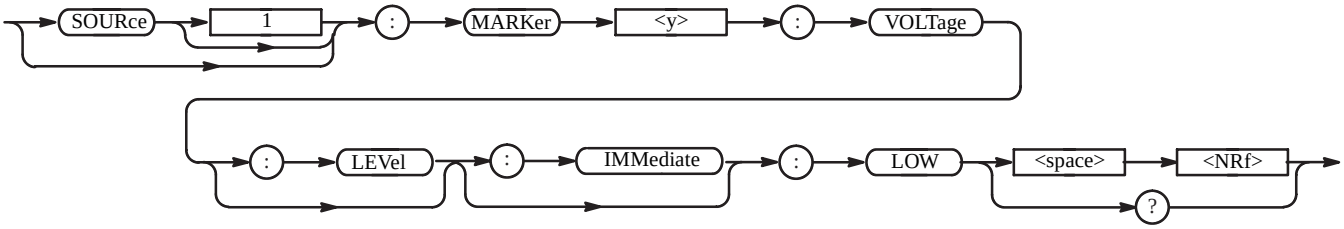
This command sets the low level voltage for the marker output.

**Group** Source

**Related Commands** `[:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:HIGH`

**Syntax** `[:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW <numeric_value>`

`[:SOURce[1]]:MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW?`




**Arguments** `<numeric_value>` is the low level voltage of the marker output. Note that the high level must be larger than the low level. The range is as follows:

AWG710:
-1.1 V to 3.0 V (into 50 Ω) with a resolution of 0.05 V. Note that the diference between high and low level is restricted to within 2.5 V.
At `*RST`, this value is set to 0.

AWG710B:
-2.00 V to 2.4 V (into 50 Ω) with a resolution of 0.05 V. Note that the diference between high and low level is restricted to within 1.25 V.
At `*RST`, this value is set to 0.

**Examples** `:SOURce1:MARKer1:VOLTage:LEVel:IMMediate:LOW -1.2V`
sets the low level voltage of the marker 1 output on CH 1 to -1.2 V.

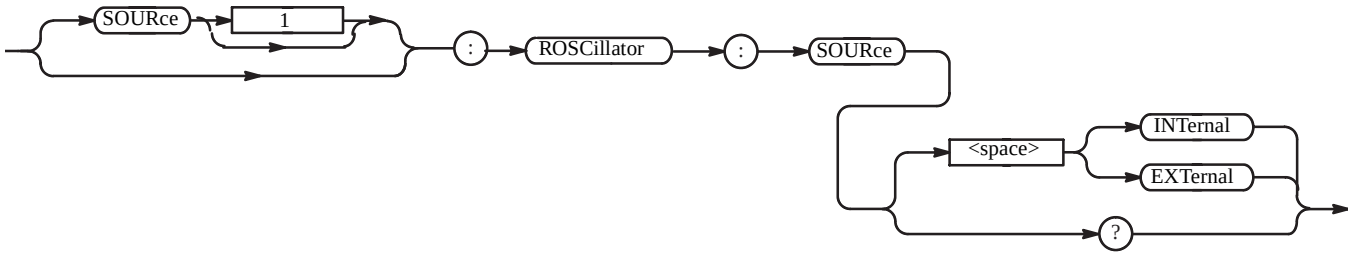
## [:SOURce[1]]:ROSCillator:SOURce (?)

This command selects the reference oscillator.

**Group** Source

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

`[:SOURce[1]]:ROSCillator:SOURce?`




**Arguments** `INTernal` means that the referenece frequency is derived from the internal precision oscillator.

`EXTernal` means the referenece frequency is derived from an external signal supplied through the Reference Clock Input connector.

At `*RST`, this parameter is set to `INTernal`.

**Examples** `:SOURce1:ROSCillator:SOURce EXTernal`
selects the external clock source.

## [:SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude] (?)

This command sets the actual magnitude of the output signal.

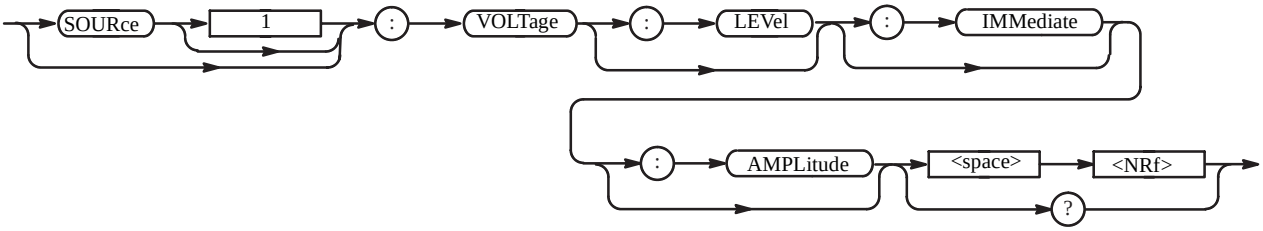
**Group** Source

**Related Commands** `[:SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet`

**Syntax**

```
[:SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]
<numeric_value>
```

```
[:SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
```




**Arguments** `<numeric_value >` is the amplitude:
The range is 20 mV to 2.0 V (into 50 Ω), in 1 mV steps.

Note that when DOUT is set to 1 (ON), the range is 20 mV to 1.0 V. (In the case of Option02: 500 mV to 1.0 V.)

At `*RST`, this value is set to 1 V.

**Examples**

```
:SOURce1:VOLTage:LEVel:IMMediate:AMPLitude 230mV
```

sets the amplitude of CH 1 waveform to 230 mV.

## [:SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet (?) (except option02)

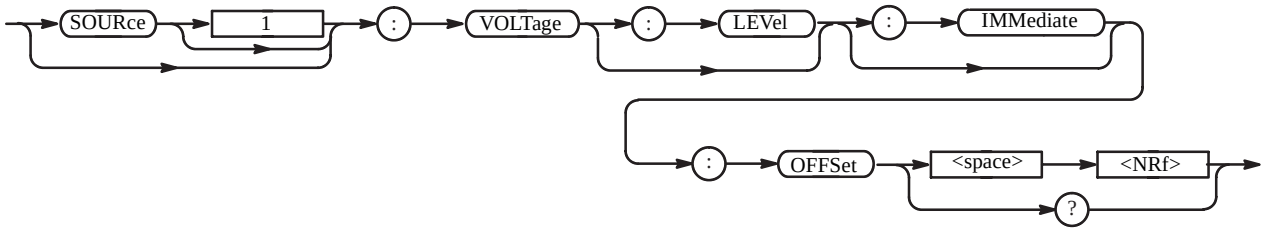
This command sets the non–time–varying component of the signal that is added to SOURce1 (CH 1).

**Group** Source

**Related Commands** [:SOURce[1]]:VOLTage[:LEVel][:IMMediate][:AMPLitude]

**Syntax** [:SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric_value>

[:SOURce[1]]:VOLTage[:LEVel][:IMMediate]:OFFSet?




**Arguments** <numeric_value> is the offset voltage.
The range is -0.500 V to +0.500 V, in 1 mV steps.

Note that when DOUT is set to 1 (ON), this command is ignored.

At *RST, this value is set to 0.

**Examples** :SOURce1:VOLTage:LEVel:IMMediate:OFFSet 50mV
sets the offset voltage of the CH 1 output to 50 mV.

## *SRE (?)

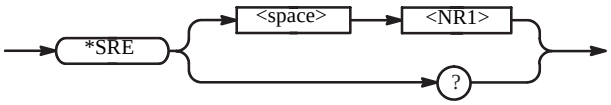
This command sets and queries the bits in the Service Request Enable Register (SRER). For a complete discussion of the use of these registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** *SRE <NR1>

*SRE?




**Arguments** <NR1> is a value in the range from 0 to 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 1. If *PSC is 0, the SRER maintains its value through a power cycle.

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

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

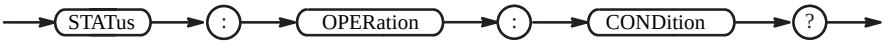
## :STATus:OPERation:CONDition? (Query Only)

This command returns the contents of the Operation Condition Register (OCR). For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** :STATus:OPERation:CONDition?



**Arguments** None

**Returns** <NR1> indicates that the content of the OCR in a decimal number.

**Examples** :STATus:OPERation:CONDition?
might return 32, which indicates that the OCR contains the binary number 00000000 00100000, and the instrument is waiting for trigger.

## :STATus:OPERation:ENABle (?)

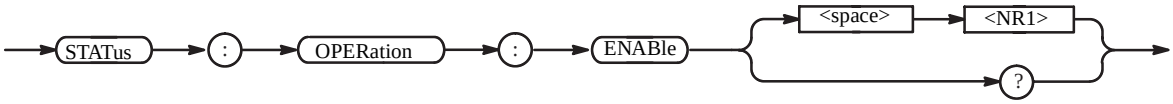
This command sets the enable mask for the Operation Enable Register (OENR). For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** :STATus:OPERation:ENABle <NR1>

:STATus:OPERation:ENABle?




**Arguments** <NR1> is the enable mask for the OENR. The range is 0 to 65535.

**Returns** <NR1> indicates that the content of the OENR in a decimal number.

**Examples** :STATus:OPERation:ENABle 1
sets the CALibrating bit in the OENR to “enable”.

:STATus:OPERation:ENABle?
might return 1 which indicates that the OENR contains the binary number 00000000 00000001, and the CAL bit is set to “enable”.

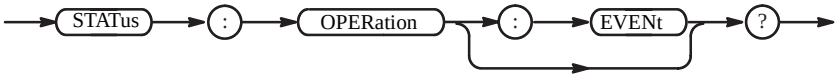
## :STATus:OPERation[:EVENt]? (Query Only)

This command returns the contents of the Operation Event Register (OEVR) and clears it. For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** `:STATus:OPERation[:EVENt]?`




**Returns** `<NR1>` indicates the content of the OEVR in a decimal number.

**Examples** `:STATus:OPERation:EVENt?`
might return 1, which indicates that the OEVR contains the binary number 00000000 00000001, and the CAL bit is set.

## :STATus:PRESet (No Query Form)

This command presets the SCPI enable registers OENR and QENR. For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

**Syntax** `:STATus:PRESet`


**Arguments** None

**Examples** `:STATus:PRESet`
presets the SCPI enable registers.

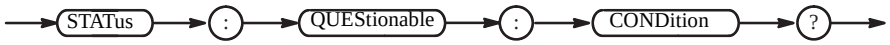
## :STATus:QUEStionable:CONDition? (Query Only)

This command returns the contents of the Questionable Condition Register (QCR). For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** :STATus:QUEStionable:CONDition?




**Returns** <NR1> indicates that the content of the QCR in a decimal number.

**Examples** :STATus:QUEStionable:CONDition?
might return 32, which indicates that the QCR contains the binary number 00000000 00100000, and the accuracy of frequency is questionable.

## :STATus:QUEStionable:ENABle (?)

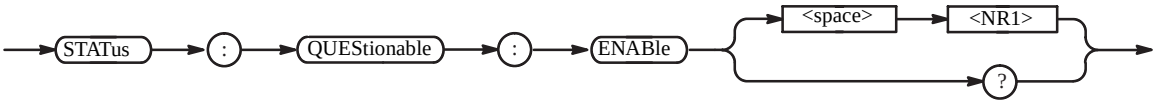
This command sets the enable mask for the Questionable Enable Register (QENR). For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

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

:STATus:QUEStionable:ENABle?




**Arguments** <NR1> is the content of the QENR. The range is 0 to 65535.

**Returns** <NR1> indicates that the content of the QENR in a decimal number.

**Examples** :STATus:QUEStionable:ENABle #H20
sets the FREQuency bit in the QENR to “enable”.

:STATus:QUEStionable:ENABle?
might return 32, which indicates that the QENR contains the binary number 00000000 00100000, and the FREQ bit is set to “enable.”

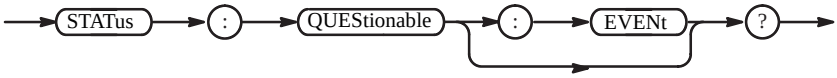
## :STATus:QUEStionable[:EVENt]? (Query Only)

This command returns the contents of the Questionable Event Register (QEVR) and clears it. For more information on registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** `:STATus:QUEStionable[:EVENt]?`




**Returns** <NR1> indicates that the contents of the QEVR in a decimal number.

**Examples** `:STATus:QUEStionable:EVENt?`
might return 32, which indicates that the QEVR contains the binary number 00000000 00100000, and the FREQ bit is set".

## *STB? (Query Only)

This command returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For a complete discussion of the use of these registers, refer to the *Status and Events* section of this manual.

**Group** Status

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

**Syntax** `*STB?`


**Arguments** None

**Returns** <NR1> indicates that the content of the SBR in a decimal number.

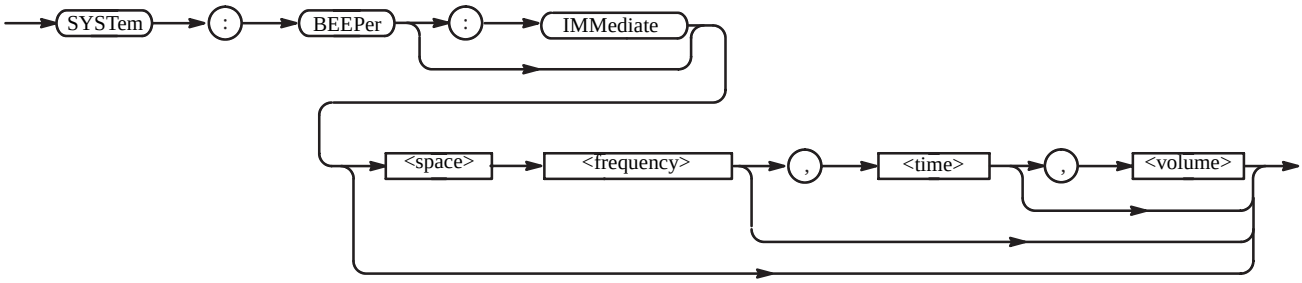
**Examples** `*STB?`
might return 96, which indicates that the SBR contains the binary number 0110 0000.

## :SYSTem:BEEPer[:IMMediate] (No Query Form)

This command causes the waveform generator to emit an audible tone.

**Group** System

**Syntax** `:SYSTem:BEEPer[:IMMediate] [<frequency>[,<time>[,<volume>]]]`




**Arguments** The following parameters are available, but are ignored:

| | |
|---|---|
| `<frequency>` | The pitch of audible tones |
| `<time>` | The duration of audible tones |
| `<volume>` | The volume of audible tones |

**Examples** `:SYSTem:BEEPer:IMMediate`
turns on a beep sound.

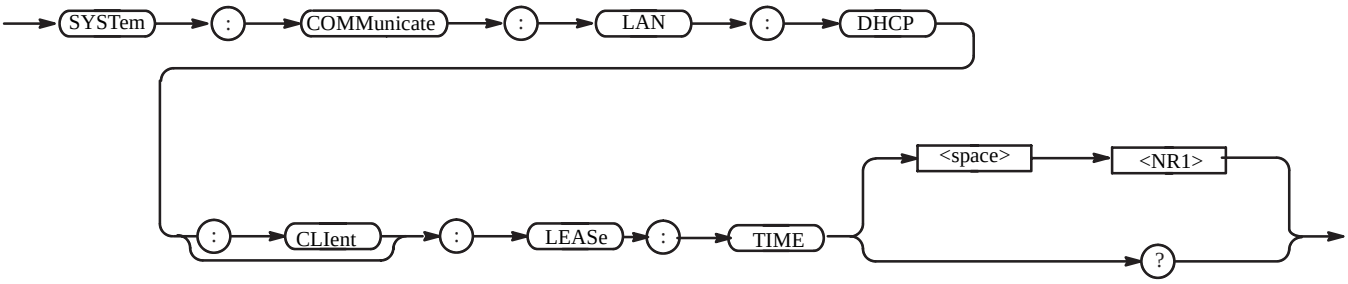
## :SYSTem:COMMunicate:LAN:DHCP[:CLIent]:LEASe:TIME (?)

This command sets the IP address lease time of the DHCP client function.

**Group** System

**Syntax** `:SYSTem:COMMunicate:LAN:DHCP[:CLIent]:LEASe:TIME <NR1>`

`:SYSTem:COMMunicate:LAN:DHCP[:CLIent]:LEASe:TIME?`




**Arguments** `<NR1>` – lease time. The range is 30 to 86400, unit is ”s”.

At `*RST`, this value is set to 28800.

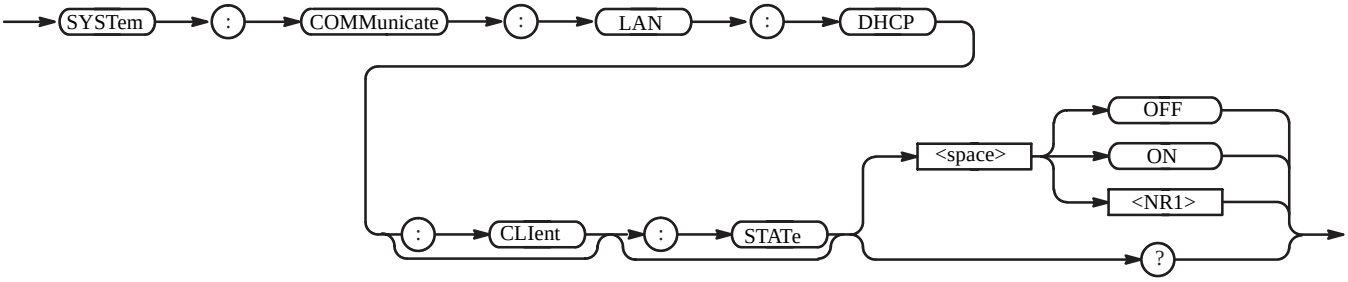
**Examples** `:SYSTem:COMMunicate:LAN:DHCP:CLIent:LEASe:TIME 7200`
sets the lease time to 7200sec.

## :SYSTem:COMMunicate:LAN:DHCP[:CLIent][:STATe] (?)

This command turns on or off the DHCP client function.

**Group** System

**Syntax** `:SYSTem:COMMunicate:LAN:DHCP[:CLIent][:STATe] <boolean>`
`:SYSTem:COMMunicate:LAN:DHCP[:CLIent][:STATe]?`



**Arguments** <boolean>::=<boolean>{ ON | OFF | 1 | 0 }
OFF or 0 turns off the DHCP client function.
ON or 1 turns on the DHCP client function.

*RST has no effect on the value.

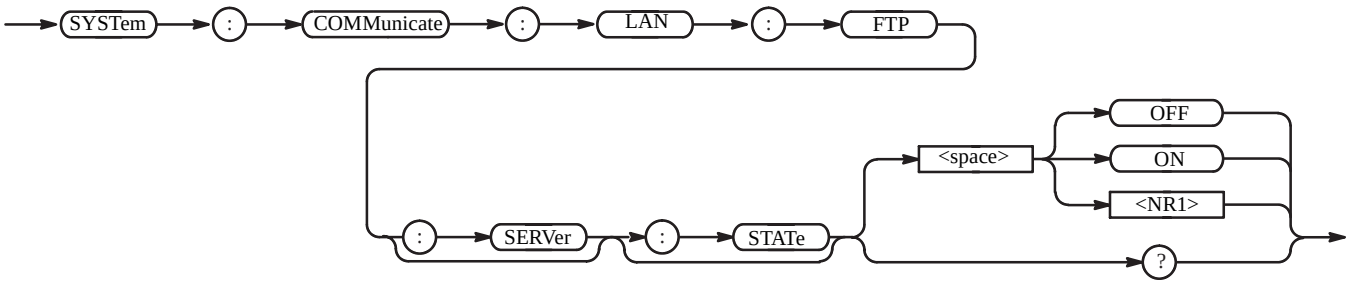
**Examples** :SYSTem:COMMunicate:LAN:DHCP:CLIent:STATe ON
sets the DHCP client function to on.

## :SYSTem:COMMunicate:LAN:FTP[:SERVer][:STATe] (?)

This command turns on or off the FTP (File Transfer Protocol) server function.

**Group** System

**Syntax** SYSTem:COMMunicate:LAN:FTP[:SERVer][:STATe] <boolean>
SYSTem:COMMunicate:LAN:FTP[:SERVer][:STATe]?




**Arguments** <boolean>::={ ON | OFF | 1 | 0 }
OFF or 0 turns off the FTP server function.
ON or 1 turns on the FTP server function.

*RST has no effect on the value.

**Examples** :SYSTem:COMMunicate:LAN:FTP:SERVer:STATe ON
sets the FTP server function on.

## :SYSTem:COMMunicate:LAN:FTP[:SERVer]:VERSion (?)

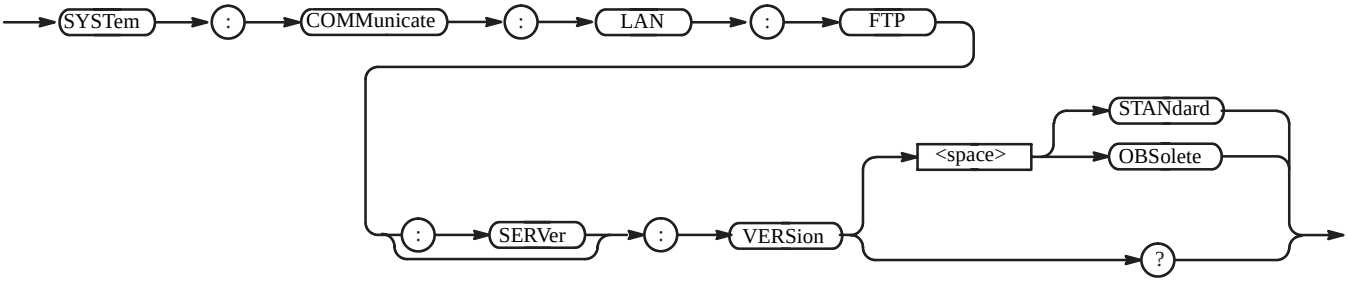
This command changes the version of the FTP (File Transfer Protocol) server.

**Group** System

**Syntax**

```
:SYSTem:COMMunicate:LAN:FTP[:SERVer]VERSion[:STATe] { STANdard |
OBSolete}
```

```
:SYSTem:COMMunicate:LAN:FTP[:SERVer]:VERSion?
```




**Arguments** `STANdar` change the FTP server version to standard.
`OBSolete` change the FTP server version to obsolete(program version 2.x).

`*RST` has no effect on the value.

**Examples**

```
:SYSTem:COMMunicate:LAN:FTP:SERVer:VERSion OBSolete
```

sets the FTP server version to obsolete.

## :SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRess (?)

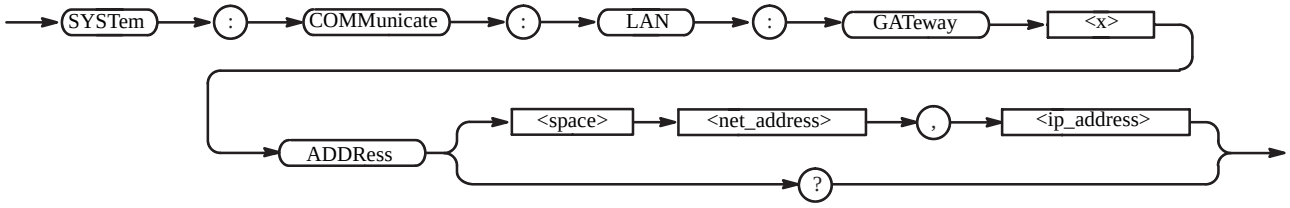
This command sets the IP address of the gateway when you communicate with the AWG710/AWG710B Arbitrary Waveform Generator from anywhere other than the local network segment.

**Group** System

**Syntax**

```
:SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRess
 <net_address>,<ip_address>
```

```
:SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRess?
```




**Arguments** <net_address>::=<string> is the network address.
<ip_address>::=<string> is the IP address of the gateway.

*RST has no effect on the value.

**Examples** :SYSTem:COMMunicate:LAN:GATeway1:ADDRess "91.0.0.0","90.0.0.2"
sets the IP address of gateway 1 to 90.0.0.2 on the net 91.0.0.0.

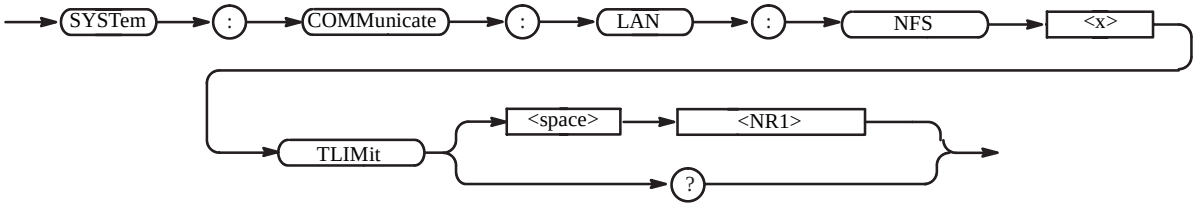
## :SYSTem:COMMunicate:LAN:NFS:TLIMit (?)

This command sets the NFS timeout.

**Group** System

**Syntax** :SYSTem:COMMunicate:LAN:NFS:TLIMit <NR1>

:SYSTem:COMMunicate:LAN:NFS:TLIMit?




**Arguments** <NR1> is the NFS timeout. The range is 25 to 300, the unit is “sec”.

At *RST, the parameter is set to 300.

**Examples** :SYSTem:COMMunicate:LAN:NFS:TLIMit 60
sets the NFS timeout to 60 sec.

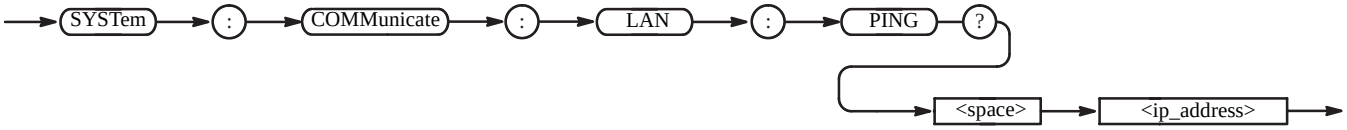
## :SYSTem:COMMunicate:LAN:PING? (Query Only)

This command executes the ping test, and sends the ICMP ECHO_REQUEST packet to a specified IP address.

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN:GATeway:ADDRess
:SYSTem:COMMunicate:LAN[:SELF]:ADDRess

**Syntax** :SYSTem:COMMunicate:LAN:GATeway:PING? <ip_address>




**Arguments** <ip_address>::=<string> is the IP address to be tested.

**Returns** <NR1>=1 indicates there was a response to the ECHO_REQUEST packet.

<NR1>=0 indicates there was no response to the ECHO_REQUEST packet.

**Examples** :SYSTem:COMMunicate:LAN:PING? "2.199.55.1"
might return a 1, indicating that there was a response from the host 2.199.55.1.

## :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess (?)

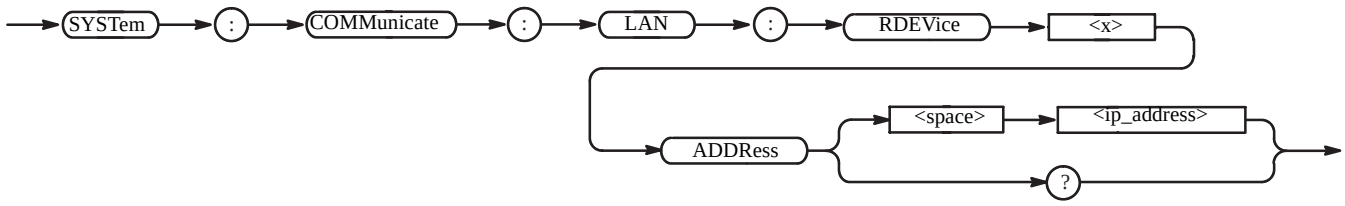
This command sets the IP address of the remote host. The host corresponds to "NET<x>" in the menu display. (You can change this name by using the :SYSTem:COMMunicate:LAN:RDEVice<x>:NAME command.)

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:FSYStem
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:NAME

**Syntax** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess <ip_address>

:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess?




**Arguments** `<ip_address>::=<string>` is the IP address of the remote host.

`*RST` has no effect on the value.

**Examples** `:SYSTem:COMMunicate:LAN:RDEVice1:ADDRess "2.199.55.1"`
sets the IP address of the remote host 1 (NET1) to 2.199.55.1.

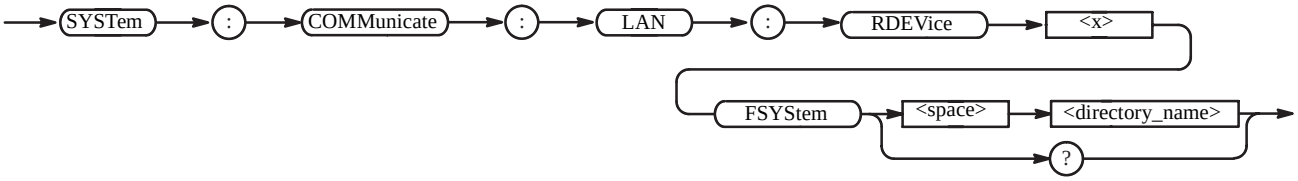
## :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:FSYStem (?)

This command sets the mount directory on a specified remote host.

**Group** System

**Related Commands** `:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess`

**Syntax**
```
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:FSYStem <directory_name>
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:FSYStem?
```




**Arguments** `<directory_name>::=<string>` is the mount directory on the remote host.
`*RST` has no effect on the value.

**Examples** `:SYSTem:COMMunicate:LAN:RDEVice1:FSYStem "/AWG/SAMPLE"`
sets the mount directory to /AWG/SAMPLE on the remote host 1 (NET1).

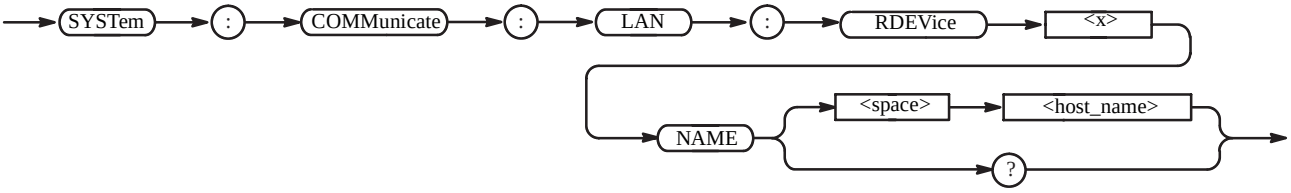
## :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:NAME (?)

This command sets the name of a specified remote host. The factory default name is “NET<x>”, which may be displayed on the waveform generator menu. You can change the displayed host name using this command.

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess

**Syntax**
```
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:NAME <host_name>
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:NAME?
```




**Arguments** <host_name>::=<string> is the name of the remote host. The name must be ten characters or less.

*RST has no effect on the parameter.

**Examples** :SYSTem:COMMunicate:LAN:RDEVice1:NAME "HOST1"
sets the name of the remote host 1 to HOST1.

## :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:PROTocol (?)

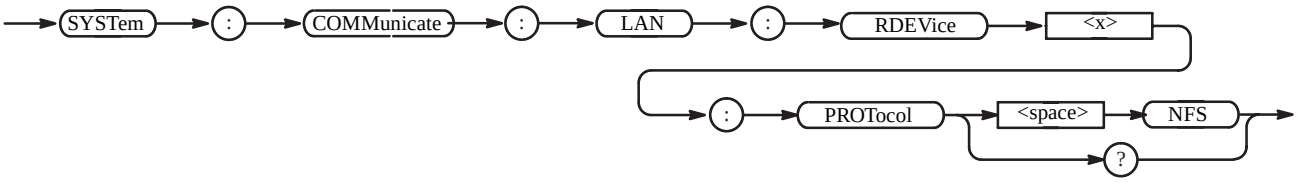
This command selects the protocol of communication with the remote host. For this application, however, the protocol is fixed to NFS (Network File System), and this command exists only for compatibility.

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess

**Syntax**
```
:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:PROTocol NFS

:SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:PROTocol?
```




**Arguments** NFS selects the NFS protocol. This is fixed.

*RST has no effect on this parameter.

**Examples** :SYSTem:COMMunicate:LAN:RDEVice1:PROTocol NFS
selects the NFS protocol.

## :SYSTem:COMMunicate:LAN:RDEVice[1|2|3][:STATe] (?)

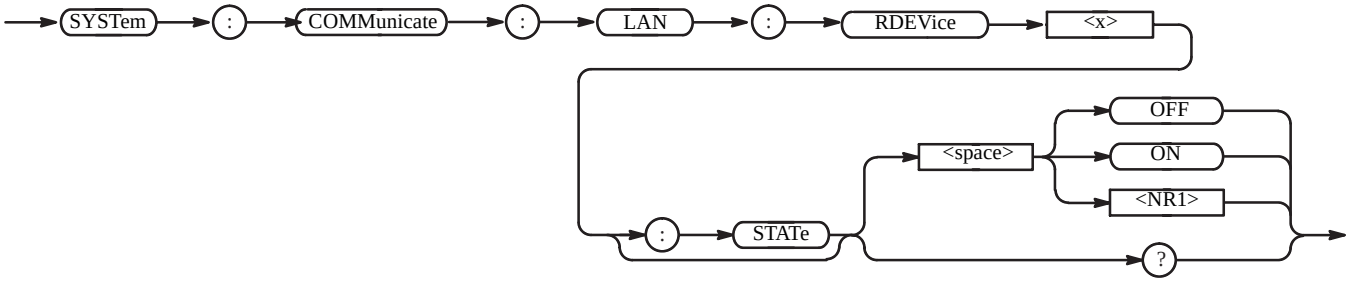
This command turns the LAN communication (Remote host's directory mount of NFS protocol) on or off, using the remote host.

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRess

**Syntax** :SYSTem:COMMunicate:LAN:RDEVice[1|2|3][:STATe] <boolean>

:SYSTem:COMMunicate:LAN:RDEVice[1|2|3][:STATe]?




**Arguments** <boolean>::={ ON | OFF | 1 | 0 }
OFF or 0 turns off the LAN communication with the remote host.
ON or 1 turns on the LAN communication with the remote host.

*RST has no effect on the value.

**Examples** :SYSTem:COMMunicate:LAN:RDEVice1:STATe ON
turns on LAN communication with the remote host.

## :SYSTem:COMMunicate:LAN[:SELF]:ADDRess (?)

This command sets the IP address of the AWG710/AWG710B Arbitrary Waveform Generator.

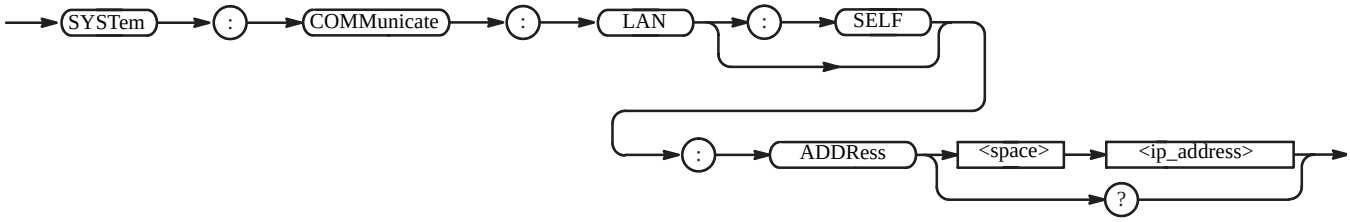
***NOTE.*** *You must set the IP address of the AWG710 /AWG710B Arbitrary Waveform Generator in order to use its LAN functions. If you specify "" (null) for the IP address, the LAN functions do not work.*

**Group** System

**Related Commands** `:SYSTem:COMMunicate:LAN[:SELF]:SMASk`

**Syntax** `:SYSTem:COMMunicate:LAN[:SELF]:ADDRess <ip_address>`

`:SYSTem:COMMunicate:LAN[:SELF]:ADDRess?`




**Arguments** `<ip_address>::=<string>` is the IP address of the AWG710 Arbitrary Waveform Generator.

`*RST` has no effect on the value.

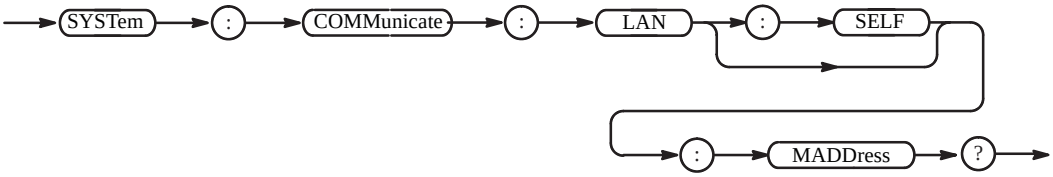
**Examples** `:SYSTem:COMMunicate:LAN:SELF:ADDRess "2.199.55.1"`
sets the IP address of the AWG710/AWG710B Arbitrary Waveform Generator.

## :SYSTem:COMMunicate:LAN[:SELF]:MADDress? (Query Only)

This command returns the MAC address.

**Group** System

**Syntax** `:SYSTem:COMMunicate:LAN[:SELF]:MADDress?`




**Arguments** <string> is the MAC address.

**Examples** :SYSTem:COMMunicate:LAN:SELF:MADDress?
might return the following response:

"XX:XX:XX:XX:XX:XX"

This response indicates the MAC address.

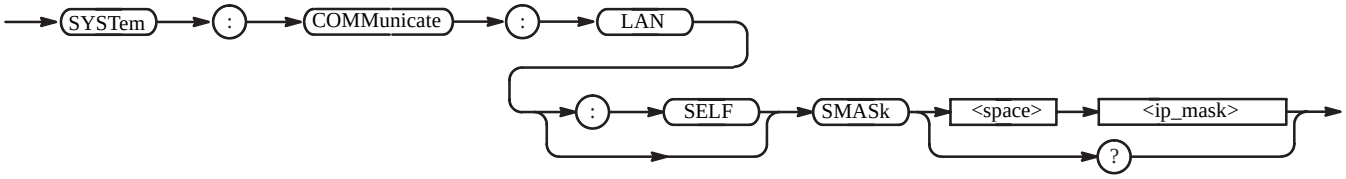
## :SYSTem:COMMunicate:LAN[:SELF]:SMASk (?)

This command sets the subnet mask of the AWG710/AWG710B Arbitrary Waveform Generator.

**Group** System

**Related Commands** :SYSTem:COMMunicate:LAN[:SELF]:ADDRess

**Syntax** :SYSTem:COMMunicate:LAN[:SELF]:SMASk <ip_mask>
:SYSTem:COMMunicate:LAN[:SELF]:SMASk?




**Arguments** <ip_mask>::=<string> is the subnet mask of the AWG710/AWG710B Arbitrary Waveform Generator.

*RST has no effect on the value.

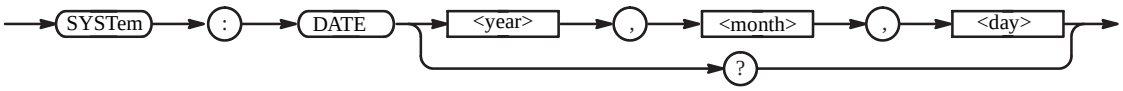
**Examples** :SYSTem:COMMunicate:LAN:SELF:SMASk "255.0.0.0"
sets the subnet mask to 255.0.0.0 for the AWG710/AWG710B Arbitrary Waveform Generator.

## :SYSTem:DATE (?)

This command sets the date for the AWG710/AWG710B Arbitrary Waveform Generator operating system.

**Group** System

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




**Arguments** <year>::=<numeric_value> must be entered as a four–digit number.

<month>::=<numeric_value> ranges 1 to 12.

<day>::=<numeric_value> ranges 1 to 31.

The range :2001.1.1 - 2099.12.31

<numeric_value> is rounded to the nearest integer.

*RST has no effect on the value.

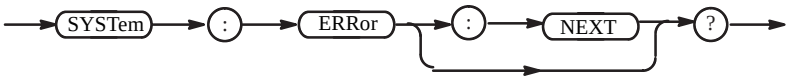
**Examples** `:SYSTem:DATE 2001,10,31`
sets the date.

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

This command retrieves and returns error data from the Error and Event Queue. For more details, refer to the *Status and Event* section of this manual.

**Group** System

**Syntax** `:SYSTem:ERRor[:NEXT]?`




**Arguments** None

**Returns** <error/event_number>,
"<error/event_description>[;<device_dependent_info>]"
where:
<error/event_number> is an integer between -32768 and 32767.
0 indicates that no error or event has occurred.
Positive values are error/event numbers determined by this instrument.
Negative values are error/event numbers reserved in SCPI standards.

<error/event_description> is a message relating to the error/event number.

<device_dependent_info> is more detailed information relating to the error/event number.

**Examples** :SYSTem:ERRor:NEXT?
might return the following response:

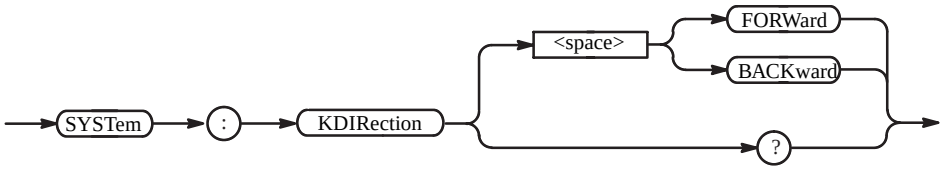
-102,"Syntax error;possible invalid suffix - :SOUR:FREQ 2V"
This response indicates that the unit is invalid.

## :SYSTem:KDIRrection (?)

This command determines the direction the cursor moves in response to the general purpose knob.

**Group** System

**Syntax** :SYSTem:KDIRection { FORWard | BACKward }
:SYSTem:KDIRection?




**Arguments** FORWard means the cursor moves to the right when the general purpose knob turns clockwise.

BACKward means the cursor moves to the left when the general purpose knob turns clockwise.

At *RST, the parameter is set to FORWard.

**Examples** :SYSTem:KDIRection BACKward
makes the cursor move to the left when you turn the general purpose knob clockwise.

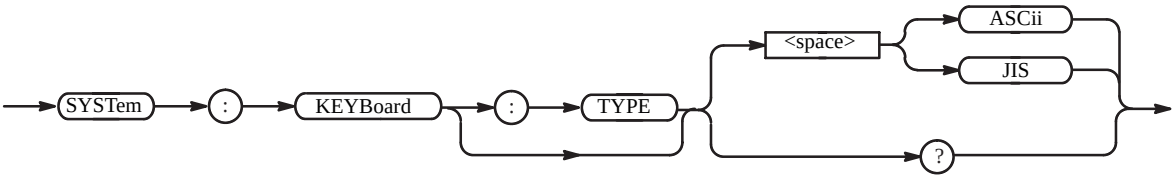
## :SYSTem:KEYBoard[:TYPE] (?)

This command selects the type of keyboard that connects to the AWG710/AWG710B Arbitrary Waveform Generator.

**Group** System

**Syntax** `:SYSTem:KEYBoard[:TYPE] { ASCii | JIS }`

`:SYSTem:KEYBoard[:TYPE]?`




**Arguments** `ASCii` selects the ASCII 101–key keyboard.
`JIS` selects the JIS 106–key keyboard.

At `*RST`, the parameter is set to `ASCii`.

**Examples** `:SYSTem:KEYBoard:TYPE JIS`
selects the JIS 106–key keyboard.

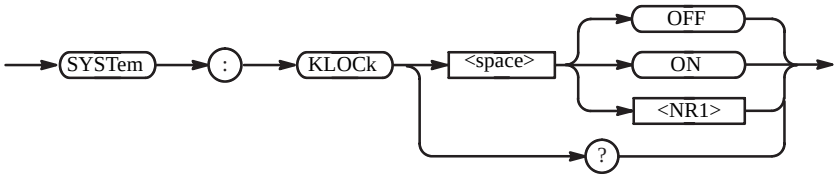
## :SYSTem:KLOCk (?)

This command locks or unlocks the front panel and keyboard. Use this command to disable manual operation while the waveform generator is being controlled externally. If the front panel and keyboard are not explicitly locked out using this command, the waveform generator accepts input from both the external controller and the front panel and keyboard.

***NOTE.*** *Pushing "CLEAR MENU" key 2 times makes the panel control* (`SYSTem:KLOCk`) *unlock. But Local Lock Out(GPIB) cannot be unlocked.*

**Group** System

**Syntax** `:SYSTem:KLOCk <boolean>`
`:SYSTem:KLOCk?`




**Arguments** `<boolean>::={ ON | OFF | 1 | 0 }`
`OFF` or `0` unlocks controls of the front panel and keyboard.
`ON` or `1` locks controls of the front panel and keyboard.

`*RST` has no effect on the parameter.

**Returns** 0 indicates the front panel and keyboard are unlocked.
1 indicates the front panel and keyboard are locked.

**Examples** `:SYSTem:KLOCk ON`
locks the front panel and keyboard.

`:SYSTem:KLOCk?`
might return 1, which indicates that the front panel and keyboard are locked.

## :SYSTem:SECurity:IMMediate (No Query Form)

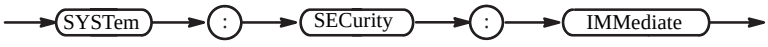
This command immediately destroys all waveform generator data and settings. Current settings are initialized to their `*RST` values.

***NOTE.*** *This command erases all information on the internal hard disk ("MAIN").*

**Group** System

**Related Commands** `*RST`

**Syntax** `:SYSTem:SECurity:IMMediate`




**Arguments** None.

**Examples** `:SYSTem:SECurity:IMMediate`
destroys all waveform generator data and settings.

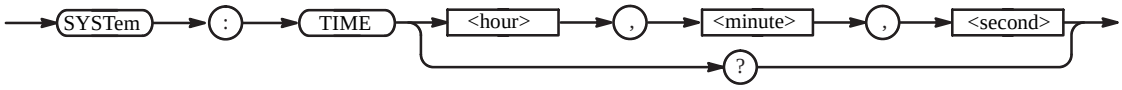
## :SYSTem:TIME (?)

This command sets the internal clock.

**Group** System

**Related Commands** :SYSTem:DATE

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




**Arguments**
```
<hour>,<minute>,<second>
where
<hour>::=<NRf> ranges 0 to 23.
<minute>::=<NRf> ranges 0 to 59.
<second>::=<NRf> ranges 0 to 59.
```

It is always rounded to the nearest integer.

**Examples**
```
:SYSTem:TIME 11,23,58
```
sets the time.

## :SYSTem:UPTime? (Query Only)

This command queries how much time has elapsed from the generator power–on.

**Group** System

**Syntax**
```
:SYSTem:UPTime?
```


**Returns**
```
<hour>,<minute>,<second>
where
<hour>::=<NR1> ranges 0 to 23.
<minute>::=<NR1> ranges 0 to 59.
<second>::=<NR1> ranges 0 to 59.
```

**Examples** :SYSTem:UPTime?
might return 3,18,52, with which indicates 3 hours 18 minutes and 52 seconds have elapsed after you powered on the waveform generator.

## :SYSTem:VERSion? (Query Only)

This command returns the SCPI version number with the waveform generator complies.

**Group** System

**Syntax** :SYSTem:VERSion?


**Returns** <NR2>::=YYYY.V
where YYYY represents the year version and V represents an approved revision number for that year.

**Examples** :SYSTem:VERSion?
might return 1999.0.

## *TRG (No Query Form)

This command generates a trigger event. This command is equivalent to the TRIGger[:SEQuence][:IMMediate] command or pressing the FORCE TRIGGER button on the front panel.

**Group** Trigger

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

**Syntax** *TRG


**Arguments** None

**Examples** *TRG
generates a trigger event.

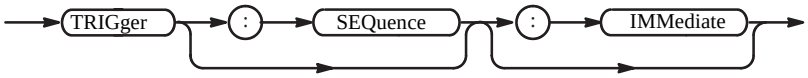
## :TRIGger[:SEQuence][:IMMediate] (No Query Form)

This command generates a trigger event. This command is equivalent to the *TRG command or pressing the FORCE TRIGGER button on the front panel.

**Group** Trigger

**Related Commands** *TRG

**Syntax** :TRIGger[:SEQuence][:IMMediate]




**Arguments** None

**Examples** :TRIGger:SEQuence:IMMediate
generates the trigger event.

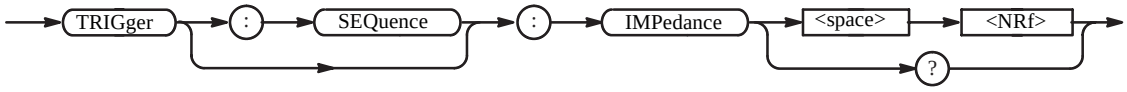
## :TRIGger[:SEQuence]:IMPedance (?)

This command selects the impedance of the external trigger input.

**Group** Trigger

**Syntax** :TRIGger[:SEQuence]:IMPedance <numeric_value>

:TRIGger[:SEQuence]:IMPedance?




**Arguments** <numeric_value> is 50 (50 Ω) or 1e3 (1 kΩ).

At *RST, the value is set to 1 kΩ.

**Examples** :TRIGger:SEQuence:IMPedance 50
selects 50 Ω impedance for the external trigger input.

## :TRIGger[:SEQuence]:LEVel (?)

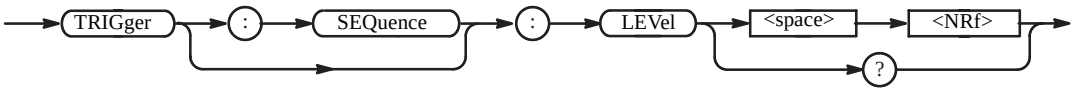
This command sets the trigger level on the selected SOURce.

**Group** Trigger

**Related Commands** :TRIGger[:SEQuence]:SOURce

**Syntax** :TRIGger[:SEQuence]:LEVel <numeric_value>

:TRIGger[:SEQuence]:LEVel?




**Arguments** <numeric_value> is the trigger level. The range is -5.0 V to +5.0 V, in 0.1 V steps.

At *RST, the value is set to 1.4 V.

**Examples** :TRIGger:SEQuence:LEVel 200mV
sets the trigger level to 200 mV.

## :TRIGger[:SEQuence]:POLarity (?)

This command selects the polarity relative to the trigger level that is required to activate the gate signal. This command is effective only when the waveform generator is in the gated mode.

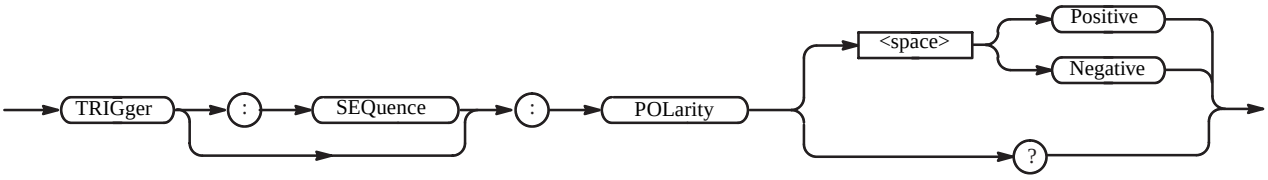
**Group** Trigger

**Related Commands** `:AWGControl:RMODe`, `:TRIGger[:SEQuence]:LEVel`

**Syntax**

```
:TRIGger[:SEQuence]:POLarity { Positive | Negative }
```

```
:TRIGger:POLarity?
```




**Arguments** `Positive` means the gate signal is activated when the external trigger signal is greater (more Positive) than the trigger level.

`Negative` means the gate signal is activated when the external trigger signal is less (more Negative) than the trigger level.

At `*RST`, the parameter is set to Positive.

**Examples**

```
:TRIGger[:SEQuence]:POLarity Negative
```

selects the `Negative` polarity.

## :TRIGger[:SEQuence]:SLOPe (?)

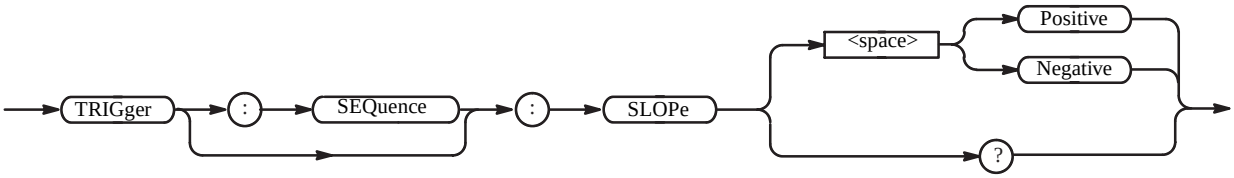
This command determines whether the event occurs on the the rising edge or falling edge of the external trigger signal.

**Group** Trigger

**Related Commands** :TRIGger[:SEQuence]:SOURce

**Syntax** :TRIGger[:SEQuence]:SLOPe { Positive | Negative }

:TRIGger[:SEQuence]:SLOPe?




**Arguments** Positive means the event occurs on the rising edge of the external trigger signal.

Negative means the event occurs on the falling edge of the external trigger signal.

At *RST, the parameter is set to Positive.

**Examples** :TRIGger:SEQuence:SLOPe Negative
selects the Negative slope.

## :TRIGger[:SEQuence]:SOURce (?)

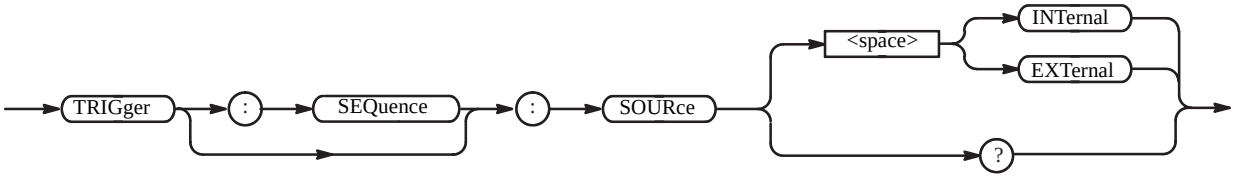
This command selects the trigger source.

**Group** Trigger

**Related Commands** :TRIGger[:SEQuence]:LEVel, :TRIGger[:SEQuence]:POLarity, :TRIGger[:SEQuence]:SLOPe, :TRIGger[:SEQuence]:TIMer

**Syntax** :TRIGger[:SEQuence]:SOURce { INTernal | EXTernal }

:TRIGger[:SEQuence]:SOURce?




**Arguments** INTernal selects the internal clock as the trigger source.

EXTernal selects the external trigger input as the trigger source.

At *RST, the parameter is set to EXTernal.

**Examples** :TRIGger:SEQuence:SOURce INTernal
selects the internal clock as the trigger source.

## :TRIGger[:SEQuence]:TIMer (?)

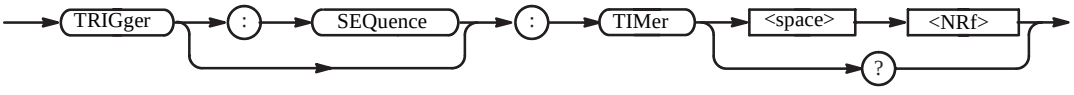
This command sets the period of the internal clock when you select the internal clock as the trigger source with the :TRIGger[:SEQuence]:SOURce command.

**Group** Trigger

**Related Commands** :TRIGger[:SEQuence]:SOURce

**Syntax** :TRIGger[:SEQuence]:TIMer <numeric_value>

:TRIGger[:SEQuence]:TIMer?



**Arguments** <numeric_value> is the internal trigger rate. The range is 1.0 ms to 10.0 s.

At *RST, this value is set to 100 ms.

**Examples** :TRIGger:SEQuence:TIMer 5ms
sets the internal trigger rate to 5 ms.

## *TST? (Query Only)

This command performs the selftest and returns the results. If an error is detected during selftest, execution is stopped immediately.

***NOTE.*** *This self test takes several minutes to complete. The waveform generator will not respond to any commands and queries during this time.*

**Group** Diagnostic

**Related Commands** *CAL?, :CALibration[:ALL], :DIAGnostic[:IMMediate]

**Syntax** *TST?


**Arguments** None

**Returns** <NR1>

0 Terminated without error.
-330 Selftest failed.

**Examples** *TST?
might return -330 indicating the selftest failed.

# *WAI (No Query Form)

This command prevents the waveform generator from executing further commands or queries until all pending operations finish.

In AWG710/AWG710B and in this application, all commands are designed to be executed in the order in which they are sent from the external controller. The *WAI command is included to ensure compliance with the SCPI standard. You do not need to use this command.

**Group** Synchronization

**Related Commands** *OPC

**Syntax** *WAI


**Arguments** None

**Examples** *WAI
prevents the execution of any commands or queries until all pending operations complete.

# Retrieving Response Messages

When a query command is sent from the external controller, the waveform generator places a response message on the output queue. To retrieve this response message you must perform a retrieval operation through the external controller. For example, you can call the IBRD subroutine with the National Instruments drivers for the GPIB interface (see Figure 2-7).

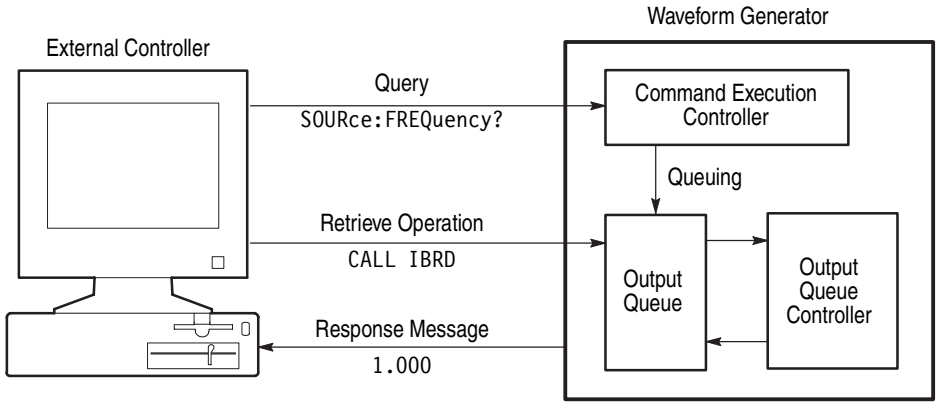



**Figure 2-7: Retrieving response messages**

Before a response message is placed in the output queue, the previous response message, if any, is deleted. Thus, if a second query occurs before the first response message is retrieved, the first response message will be lost.

The SBR (status byte register) MAV bit can be used to check the response message queuing state. Refer to the *Status and Events* section in this manual for more information about the output queue, SBR, and control methods.

# Data Transfer

You can transfer data between the waveform generator and external devices through the GPIB and Ethernet LAN interface. This section describes required data formats and transfer procedures.

## Data File

The waveform generator uses the following file types:

- The Waveform file contains waveform data in single precision floating point format.
- The Pattern file contains waveform data in binary format.
- The Sequence file defines the output sequence.
- The Equation file uses numeric formulas to describe the output waveform.
- The Code Convert file contains the Code Convert Table.

During front panel operation, the waveform generator creates these files automatically; when you remotely operate the waveform generator, you must create these files through editing or programming according to the formats described in the topics that follow.

## About Waveform and Pattern Files

To output a wave form you can load both the Waveform and Pattern file. When you load a Waveform file, it is converted and stored to waveform memory with 8 bit digital patterns.

The instrument stores data in the Pattern file to waveform memory without conversion.

The difference between these two files is the internal format and the editor. The Waveform file format is composed of a 4–byte Little Endian format specified in IEEE488.2 floating point numbers, and 1–byte of marker data (see page 2-115 for format details). The Pattern file format is composed of 2–bytes, including data and markers (see page 2-116 for format details).

### Guidelines for using files

Following are some guidelines for choosing either the Waveform file or Pattern file to ouput waveforms.

- Select a Pattern file to shorten the transfer time when you do not need to perform further edits or operations in the instrument. Although both files are the same data length, the volume of the Pattern file is always less than the volume of the Waveform file.
- Use a Waveform file when you use waveform data to generate another waveform with a mathematical operation. The Waveform file format retains the data precision required for mathematical operations.

For more details about file formats, refer to *Data Transfer* section in one of these manuals:

*AWG710&AWG710B Arbitrary Waveform Generator User Manual.*

## Waveform File

The Waveform file contains waveform data in single precision floating–point numbers and marker data.

**File Format.** The Waveform file consists of three main parts (see Figure 2-8).

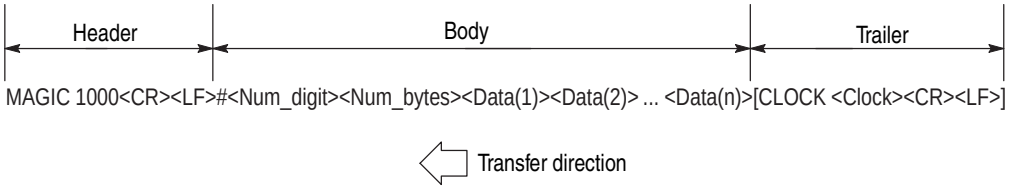



**Figure 2-8: The Waveform file format**

<Waveform File>::=<Header><Body>[<Trailer>]

where:

<Header>::=MAGIC<space>1000<CR><LF>

<Body>::=#<Num_digits><Num_bytes><Data(1)><Data(2)>...<Data(n)>

<Num_digits> is the number of digits in <Num_bytes>.

<Num_bytes> is the byte count of the data that follows.

<Data(n)>::=<Waveform><Marker>

<Waveform> is the single precision floating–point number of 4–byte Little Endian format specified in IEEE488.2. The full scale of the D/A converter of the waveform generator corresponds to -1.0 to 1.0.

<Marker> is one byte of marker data. The bit 0 (LSB) and bit 1 represent markers 1 and 2, respectively.

<Trailer>::=CLOCK<space><Clock><CR><LF>

<Clock> is the value of the sample clock in ASCII.

**Example.** This example shows the contents of a Waveform file that contains two point data.

```
4D 41 47 49 43 20 31 30 30 30 0D 0A 23 32 31 30        MAGIC 1000..#210
00 00 00 00 03 00 00 00 00 00 43 4C 4F 43 4B 20        ..........CLOCK
31 2E 30 30 30 30 30 30 30 30 30 30 65 2B 30 38        1.0000000000E+08
0D 0A                                                  ..
```

## Pattern File

The Pattern file contains waveform data in binary format.

**File Format.** The data consists of three main parts (see Figure 2-9).

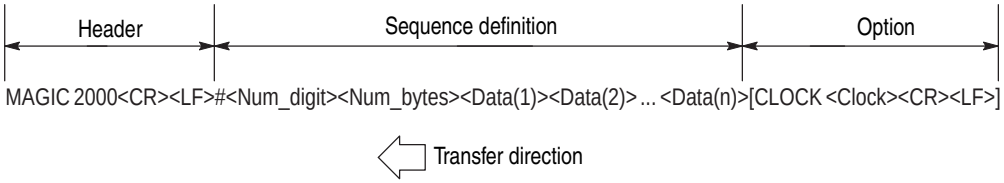



**Figure 2-9: The Pattern File format**

<Pattern File>::=<Header><Body>[<Trailer>]

where:

<Header>::=MAGIC<space>2000<CR><LF>

<Body>::=#<Num_digits><Num_bytes><Data(1)><Data(2)>...<Data(n)>

<Num_digits> is the number of digits in <Num_bytes>.

<Num_bytes> is the byte count of the data that follows.

<Data(n)>

In the AWG710, this represents each data point in two bytes (16 bits). The low byte is transferred first.

Bits 2 (LSB) - 9 are D0 - D79 of the rear panel.

Bits 13 and 14 are used for Markers 1 and 2, respectively.

Bits 0, 1, 10 through 12, and 15 are unused and must be 0 (zero).

<Trailer>::=CLOCK<space><Clock><CR><LF>

<Clock> is the value of the sample clock in ASCII.

**Sequence File** The Sequence file defines the output sequence in ASCII format.

**File Format.** The data consists of three main parts (see Figure 2-10).

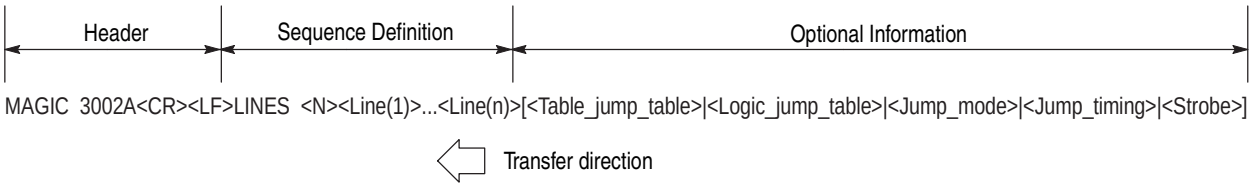



**Figure 2-10: The Sequence File format**

```
<Sequence File>
::=<Header><Sequence Definition>[<Optional Information>]
```

where:

```
<Header>::=MAGIC<space>3002A<CR><LF>
```

```
<Sequence Definition>
::=LINES<space><N><Line(1)><Line(2)>...<Line(n)>
```

<N> is the number of lines that follow.

```
<Line(n)>::=<CH1_file_name>,<CH2_file_name>,<Repeat_count>
[,<Wait_trigger>[<Goto-1>[,<Logic_jump_target>
[,<Goto-N>]]]]<CR><LF>
```

<CHx_file_name>::=<string> is the waveform or pattern file name for the specified channel.
CH2 field is for compatible with AWG500 series. In AWG710 & AWG710B, CH2 field is ””(null string).

<Repeat_count>::=<NR1> is the repeat count for the line. 0 (zero) is infinity.

<Wait_trigger>::=<NR1> specifies whether or not to wait for a trigger. <NR1>=0 is Off, ≠0 is On.

<Goto-1>::=<NR1> not be used in AWG710 & AWG710B. But write 0 or 1.

<Logic_jump_target>::=<NR1> is line number for the Logic–Jump. 0 is Off, -1 is Next, and -2 is Table–Jump. The default is Off.

<Goto-N>::=<NR1> is line number to go after current line. 0 is Next, N is 0<N≤8000.

<Optional Information>
::={ <Table_jump_table> | <Logic_jump_table> | <Jump_mode> |
<Jump_timing> | <Strobe> }

<Table_jump_table>
::=TABLE_JUMP<space><Jump_target(1)>,<Jump_target(2)>,
...<Jump_target(m)><CR><LF>

m::=128 (AWG710B), 16 (AWG710)
<Jump_target(n)>::=<NR1> is the line number to the Table–Jump or 0 (Off). The default is Off.

<Logic_jump_table>
::=LOGIC_JUMP<space><Jump_on/off(1)>,<Jump_on/off(2)>,
...,<Jump_on/off(m)><CR><LF>

m::=7 (AWG710B), 4 (AWG710)
<Jump_on/off(n)>::=<NR1> sets the Logic–Jump on or off.
<NR1>=0 is Off, 0> is On, and <0 is Ignore. The default is Ignore.

<Jump_mode>::=JUMP_MODE<space>{ LOGIC | TABLE | SOFTWARE }
<CR><LF>
sets the jump mode. The default is TABLE.

<Jump_timing>::=JUMP_TIMING<space>{ SYNC | ASYNC }<CR><LF>
sets the jump mode. The default is ASYNC.

<Strobe>::=STROBE<space><NR1><CR><LF> determines whether or not to use the STROBE signal from the EVENT IN connector on the rear panel.
<NR1>=0 is Off, ≠0 is On. The default is Off.

**Example.** This Sequence file contains two lines of sequence definitions for CH 1.

```
MAGIC 3002
LINES 2
"SAMPLE1.WFM","",1,0,0,0,0
"SAMPLE3.WFM","",1,0,0,0,0
TABLE_JUMP 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
LOGIC_JUMP -1,-1,-1,-1
JUMP_MODE TABLE
JUMP_TIMING ASYNC
STROBE 0
```

## Equation File

The Equation file describes the numerical formula that defines the output waveform in ASCII format.

**File Format.** The Equation file consists of ASCII characters (see Figure 2-11).

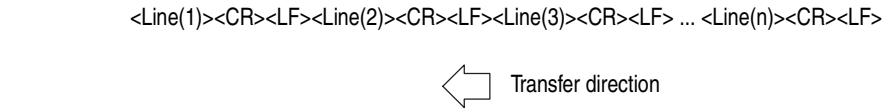



**Figure 2-11: The Equation File format**

<Line(n)> represents each line of the Equation file. From single (') quotation marks to the end of the line is a comment. Characters enclosed in double (") quotation marks are a character string.

Detailed information about the functions and operators that can be used to describe the Equation file, can be found in the *AWG710 Arbitrary Waveform Generator User Manual*.

**Example**. This Equation file describes the log sweep waveform.

```
´frequency sweep sine (log)
clock=800e6
size=8800
k0=11e-6      ´sweep period
k1=1e6        ´starting frequency
k2=10e6       ´ending frequency
k3=log(k2/k1)
"log_swp.wfm"=sin(2*pi*k1*k0/k3*(exp(k3*scale)-1))
```

## Code Convert File

The Code Convert file is an ASCII text file that describes the Code Convert Table as displayed in the Edit menu.

**File Format.** The Code Convert file consists of bit pattern definitions (see Figure 2-12).

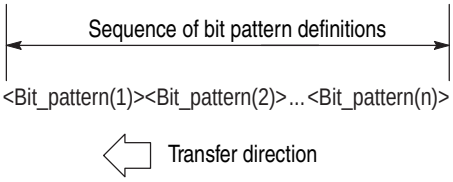



**Figure 2-12: The Code Convert File format**

<Code Convert File>::=<Bit_pattern(1)><Bit_pattern(2)> ...
<Bit_pattern(n)>

where:

<Bit_pattern(n)>::=[<Past Source>,<Current Source>,
<Next Source>,<Past Output>,<Output Code><CR><LF>]

<Past Source>, <Current Source>, <Next Source>, <Past Output>, and <Output Code> specifies the bit patterns in the Code Convert Table. The bit pattern is specified with “0”, “1”, and “-” (don’t care).

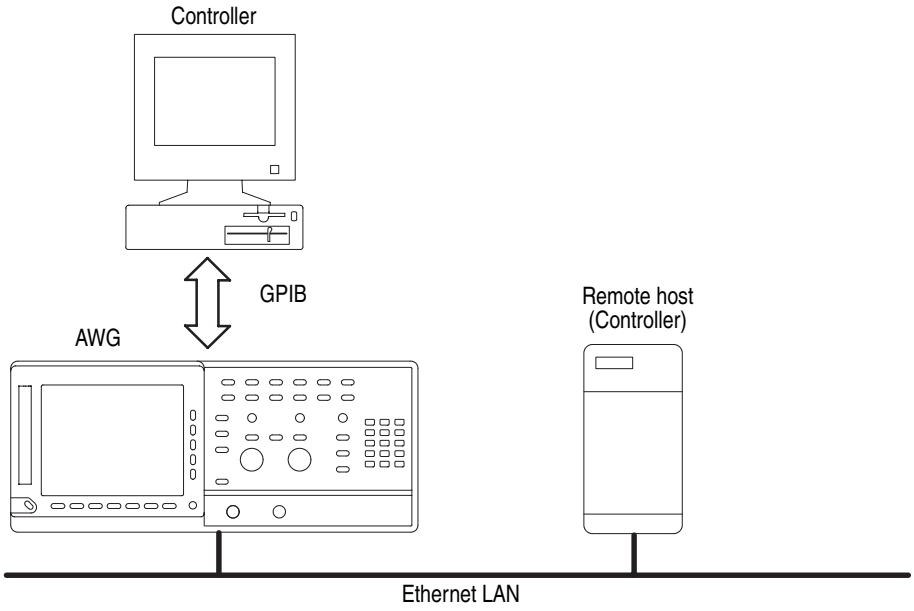
For more information about the Code Convert Table, refer to one of these manuals:
*AWG710&AWG710B Arbitrary Waveform Generator User Manual.*

**Example.** This Code Convert file describes NRZI conversion.

```
-,0---,,0,0
-,0---,,1,1
-,0---,,0,1
-,0---,,1,0
```

## Data Transfer Procedures

Data can be loaded from the external controller to the waveform generator or from the waveform generator to the external controller through the GPIB interface, or through the Ethernet interface.




### External Device to Waveform Generator

Use the following command to transfer data from the external controller to the waveform generator:

```
MMEMory:DATA <file_name>,<data>
```

This command downloads <data> into the file <file_name> on the internal hard disk, floppy disk, or the network drive. The default directory and mass memory device are specified by the `MMEMory:CDIRectory` and `MMEMory:MSIS` commands respectively. The <data> is in IEEE488.2 block format.

For example, the following command string will load 2048 bytes of data to the file AWG1.

```
MMEMory:DATA "AWG1",#42048<data(1)><data(2)>...<data(2048)>
```

### Waveform Generator to External Device

Use the following command to transfer data from the waveform generator to the external controller.

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

This command uploads the file <file_name> on the internal hard disk, floppy disk, or the network drive. The response format is in IEEE488.2 block format.

For example, the following command string will upload the file FILE–AWG on the waveform generator to the external controller.

```
MMEMory:DATA? "FILEÐAWG"
```

# Status and Events

# Status and Event Reporting

This section provides details about the status information and events the waveform generator reports.

## Status Reporting Structure

The waveform generator status reporting functions conform to IEEE–488.2 and SCPI standards. Use the status reporting function to check for instrument errors and to identify the types of events that have occurred on the instrument.

Figure 3-1 is a diagram of the instrument's status reporting function. The status reporting function is separated into three functional blocks:

- Standard/Event Status
- Operation Status
- Questionable Status

The operations processed in these three blocks are summarized in status bytes, which provide the error and event data.

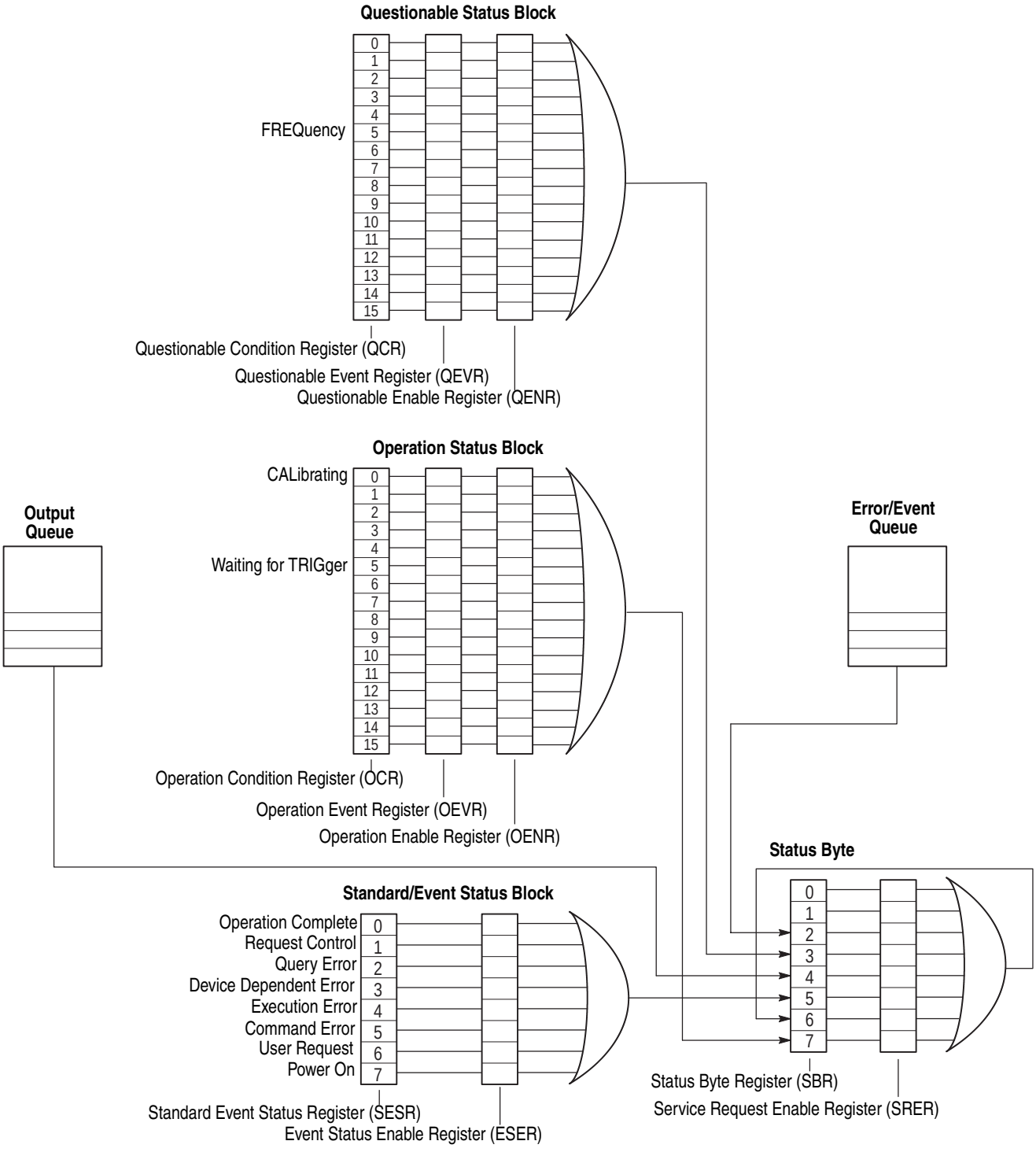



**Figure 3-1: Error and Event handling process overview**

### Standard/Event Status Block

This block is used to report power on/off, command error, and command execution status.

The block has two registers: the Standard Event Status Register (SESR) and the Event Status Enable Register (ESER). Refer to the Standard/Event Status Block shown at the bottom of Figure 3-1 on page 3-2.

The SESR is an eight–bit status register. When an error or other type of event occurs on the instrument, the corresponding bit is set. You cannot write to this register. The ESER is an eight–bit enable register that masks the SESR. You can set this mask, and take AND with the SESR to determine whether or not the ESB bit in the Status Byte Register (SBR) should be set. Refer to *Event Status Enable Register (ESER)* on page 3-8, and *Standard Event Status Register (SESR)* on page 3-6, for the contents of these registers.

### Operation Status Block

This block is used to report on the status of several operations being executed by the waveform generator.

The block is made up of three registers: the Operation Condition Register (OCR), the Operation Event Register (OEVR) and the Operation Enable Register (OENR). Refer to the Operation Status Block shown in the middle of Figure 3-1 on page 3-2.

When the instrument achieves a certain status, the corresponding bit is set to the OCR. You cannot write to this register. OCR bits that have changed from false (reset) to true (set) status are set in the OEVR. The function of the OENR is to mask the OEVR. You can set this mask and take AND with the OEVR to determine whether or not the OSS bit in the Status Byte Register (SBR) should be set. Refer to *Operation Condition Register (OCR)* on page 3-7, *Operation Event Register (OEVR)* on page 3-7, and *Operation Enable Register (OENR)* on page 3-9, for the contents of these registers.

### Questionable Status Block

This block reports on the status of signals and data, such as the accuracy of entered data and signals generated by the instrument. The register configuration and process flow are the same as for the Questionable Status Block. Refer to *Questionable Condition Register (QCR)* on page 3-7, *Questionable Event Register (QEVR)* on page 3-8, and *Questionable Enable Register (QENR)* on page 3-9, for the contents of these registers.

## Registers

There are two main types of registers:

- **Status Registers**: store data relating to instrument status. These registers are set by the waveform generator.
- **Enable Registers**: determine whether to set events that occur in the instrument to the appropriate bits in the status registers and event queues. You can set this register.

## Status Registers

There are six types of status registers:

- Status Byte Register (SBR)
- Standard Event Status Register (SESR)
- Operation Condition Register (OCR)
- Operation Event Register (OEVR)
- Questionable Condition Register (QCR)
- Questionable Event Register (QEVR)

Read the contents of these registers to determine errors and conditions.

## Status Byte Register (SBR)

The SBR is made up of 8 bits. Bits 4, 5 and 6 are defined in accordance with IEEE Std 488.2–1987 (see Figure 3-2 and Table 3-1). These bits are used to monitor the output queue, SESR, and service requests, respectively. The contents of this register are returned when the *STB? query is used.

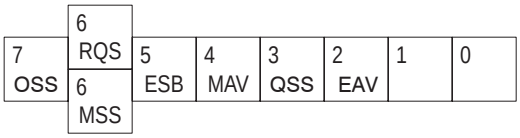



**Figure 3-2: The Status Byte Register (SBR)**

**Table 3-1: SBR bit functions**

| Bit | Function |
|---|---|
| 7 | Operation Summary Status (OSS). |
| 6 | RQS (Request Service)/MSS (Master Summary Status). When the instrument is accessed using the GPIB serial poll command, this bit is called the Request Service (RQS) bit and indicates to the controller that a service request has occurred (in other words, that the GPIB bus SRQ line is LOW). The RQS bit is cleared when serial poll ends.<br><br>When the instrument is accessed using the *STB? query, this bit is called the Master Summary Status (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query. |
| 5 | Event Status Bit (ESB). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed. |
| 4 | Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved. |
| 3 | Questionable Summary Status (QSS). |
| 2 | Event Queue Available (EAV). |
| 1-0 | Not used |

### Standard Event Status Register (SESR)

The SESR is made up of 8 bits. Each bit records the occurrence of a different type of event, shown in Figure 3-3 and Table 3-2. The contents of this register are returned when the *ESR? query is used.

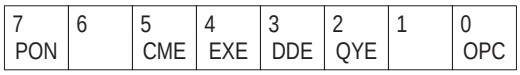



**Figure 3-3: The Standard Event Status Register (SESR)**

**Table 3-2: SESR bit functions**

| Bit | Function |
|---|---|
| 7 | Power On (PON). Indicates that the power to the instrument is on. |
| 6 | Not used. |
| 5 | Command Error (CME). Indicates that a command error has occurred while parsing by the command parser was in progress. |
| 4 | Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons:<br>■ A value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument<br>■ The command could not be executed properly because the conditions for execution differed from those essentially required |
| 3 | Device–Specific Error (DDE). An instrument error has been detected. |
| 2 | Query Error (QYE). Indicates that a query error has been detected by the output queue controller. Query errors occur for one of the following reasons:<br>■ An attempt was made to retrieve messages from the output queue, despite the fact that the output queue is empty or in pending status.<br>■ The output queue messages have been cleared despite the fact that they have not been retrieved. |
| 1 | Not used. |
| 0 | Operation Complete (OPC). This bit is set with the results of the execution of the *OPC command. It indicates that all pending operations have been completed. |

### Operation Condition Register (OCR)

The OCR is made up of 16 bits, which record the occurrence of three types of events, shown in Figure 3-4 and Table 3-3.


**Figure 3-4: The Operation Condition Register (OCR)**

**Table 3-3: OCR bit functions**

| Bit | Function |
|---|---|
| 15 - 6 | Not used. |
| 5 | Waiting for Trigger (TRIG). Indicates whether the instrument is waiting for a trigger. This bit is set when CH 1 or another channel is waiting for a trigger. It is reset when the waiting–for–trigger status is canceled. |
| 4- 1 | Not used. |
| 0 | Calibration (CAL): Indicates whether the instrument is being calibrated. This bit is set when calibration is in progress and is reset when calibration ends. |

### Operation Event Register (OEVR)

In this instrument, this register has the same content as the Operation Condition Register (OCR), described above.

### Questionable Condition Register (QCR)

The QCR is made up of 16 bits, which note the occurrence of only one type of event, as explained below.


**Figure 3-5: The Questionable Condition Register (QCR)**

**Table 3-4: QCR bit functions**

| Bit | Function |
|---|---|
| 15 - 6 | Not used. Must be set to zero for the waveform generator operation. |
| 5 | Frequency (FREQ). Indicates whether frequency accuracy of the signal is of questionable quality. |
| 4 - 0 | Not used. Must be set to zero for the waveform generator operation. |

### Questionable Event Register (QEVR)

This register holds the same content as the Questionable Condition Register (QCR).

## Enable Registers

There are four types of enable registers:

- Event Status Enable Register (ESER)
- Service Request Enable Register (SRER)
- Operation Enable Register (OENR)
- Questionable Enable Register (QENR)

Each bit in the enable registers corresponds to a bit in the controlling status register. By setting and resetting the bits in the enable register, you can determine whether or not events that occur will be registered to the status register and queue.

### Event Status Enable Register (ESER)

The ESER is made up of bits defined exactly the same as bits 0 through 7 in the SESR register (see Figure 3-6). You can use this register to designate whether or not the SBR ESB bit should be set when an event has occurred, and to determine if the corresponding SESR bit is set.

To set the SBR ESB bit (when the SESR bit has been set), set the ESER bit corresponding to that event. To prevent the ESB bit from being set, reset the ESER bit corresponding to that event.

Use the *ESE command to set the bits of the ESER. Use the *ESE? query to read the contents of the ESER.

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

**Figure 3-6: The Event Status Enable Register (ESER)**

### Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR (see Figure 3-7). 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.

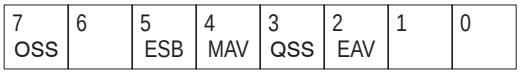



**Figure 3-7: The Service Request Enable Register (SRER)**

### Operation Enable Register (OENR)

The OENR is made up of bits that are defined exactly the same as bits 0 through 15 in the OEVR register (see Figure 3-8). The operator uses this register to define whether or not the OSS bit 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.


**Figure 3-8: The Operation Enable Register (OENR)**

### Questionable Enable Register (QENR)

The QENR is made up of bits that are defined exactly the same as bits 0 through 15 in the QEVR register (see Figure 3-9). You can use this register to define whether the QSS bit 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.

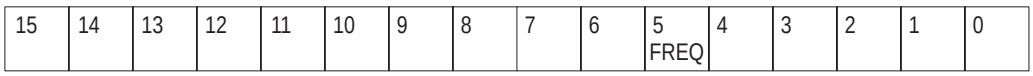



**Figure 3-9: The Questionable Enable Register (QENR)**

## Queues

There are two types of queues in the status reporting system: output queues and error/event queues.

### Output Queue

The output queue is a FIFO (first–in, first–out) queue that holds response messages to queries awaiting retrieval. When there are messages in the queue, the SBR MAV bit is set.

The output queue is emptied each time a command or query is received, so the controller must read the output queue before the next command or query is issued. If this is not done, an error occurs and the output queue is emptied; however, the operation proceeds even if an error occurs.

### Error/Event Queue

The event queue is a FIFO queue, which stores events as they occur in the instrument. If more than 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.

## Status and Event Processing Sequence

### Operation Status Block

As illustrated in Figure 3-10 below, a signal is sent to the OEVR (1) when an event occurs. If the corresponding bit in the OENR is also enabled (2), the OSS bit in the SBR is set to one (3). [See Figure 3-12 on page 3-12].

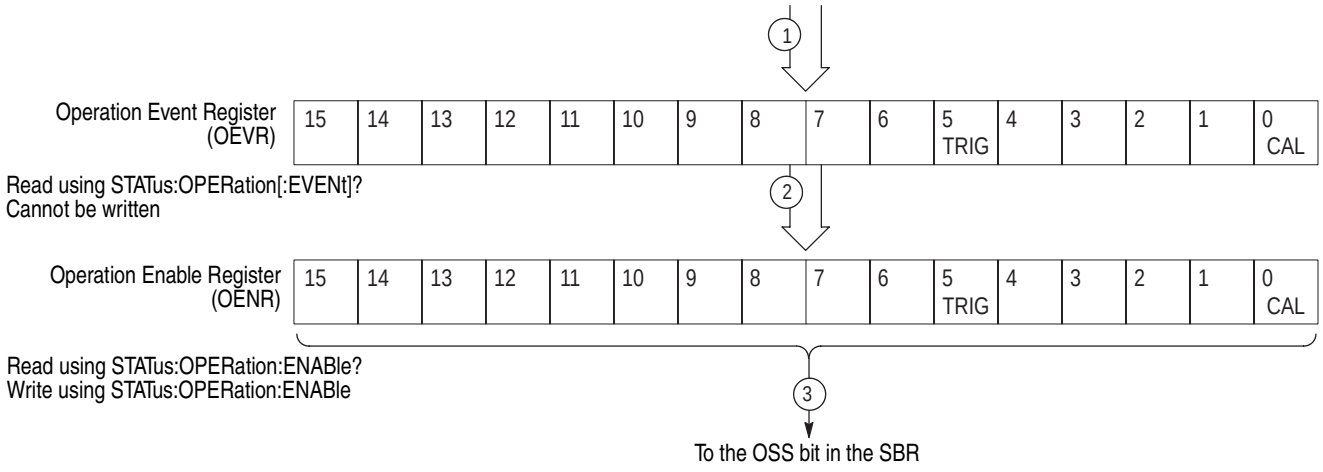



**Figure 3-10: Status and Event processing sequence — Operation status block**

### Questionable Status Block

As illustrated in Figure 3-11, when an event occurs, a signal is sent to the QEVR (1). If the corresponding bit in the QENR is also enabled (2), the QSS bit in the SBR is set to one (3). [See Figure 3-12 on on page 3-12].

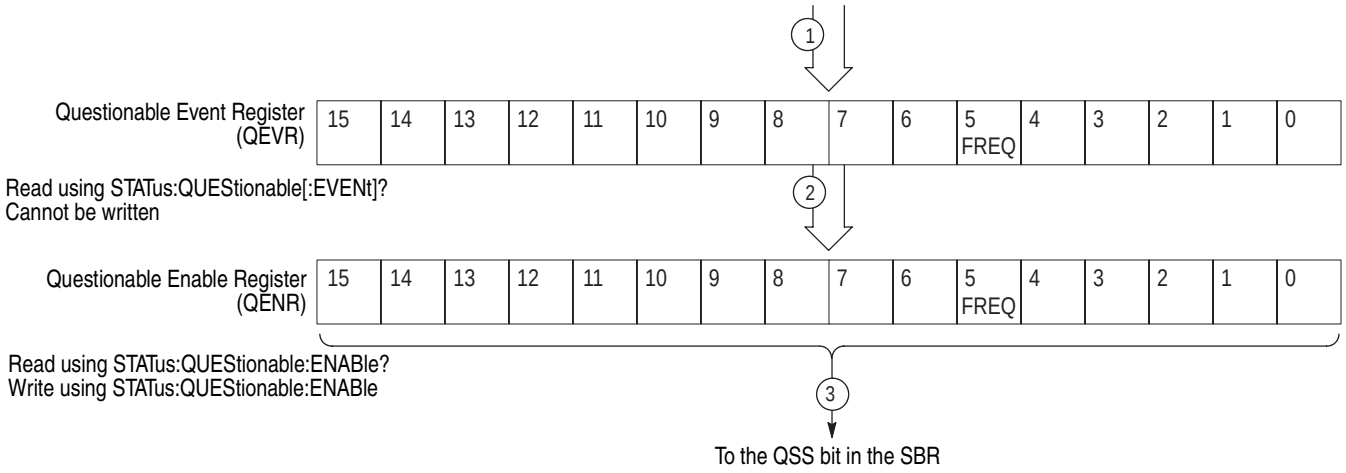



**Figure 3-11: Status and Event processing sequence — Questionable status block**

### Standard/Event Status Block

As illustrated in Figure 3-12, when an event occurs, a signal is sent to the SESR and the event is recorded in the Event Queue (1). If the corresponding bit in the ESER is also enabled (2), the ESB bit in the SBR is set to one (3).

When output is sent to the Output Queue, the MAV bit in the SBR is set to one (4).

When a bit in the SBR is set to one and the corresponding bit in the SRER is enabled (5), the MSS bit in the SBR is set to one and a service request is generated (6).

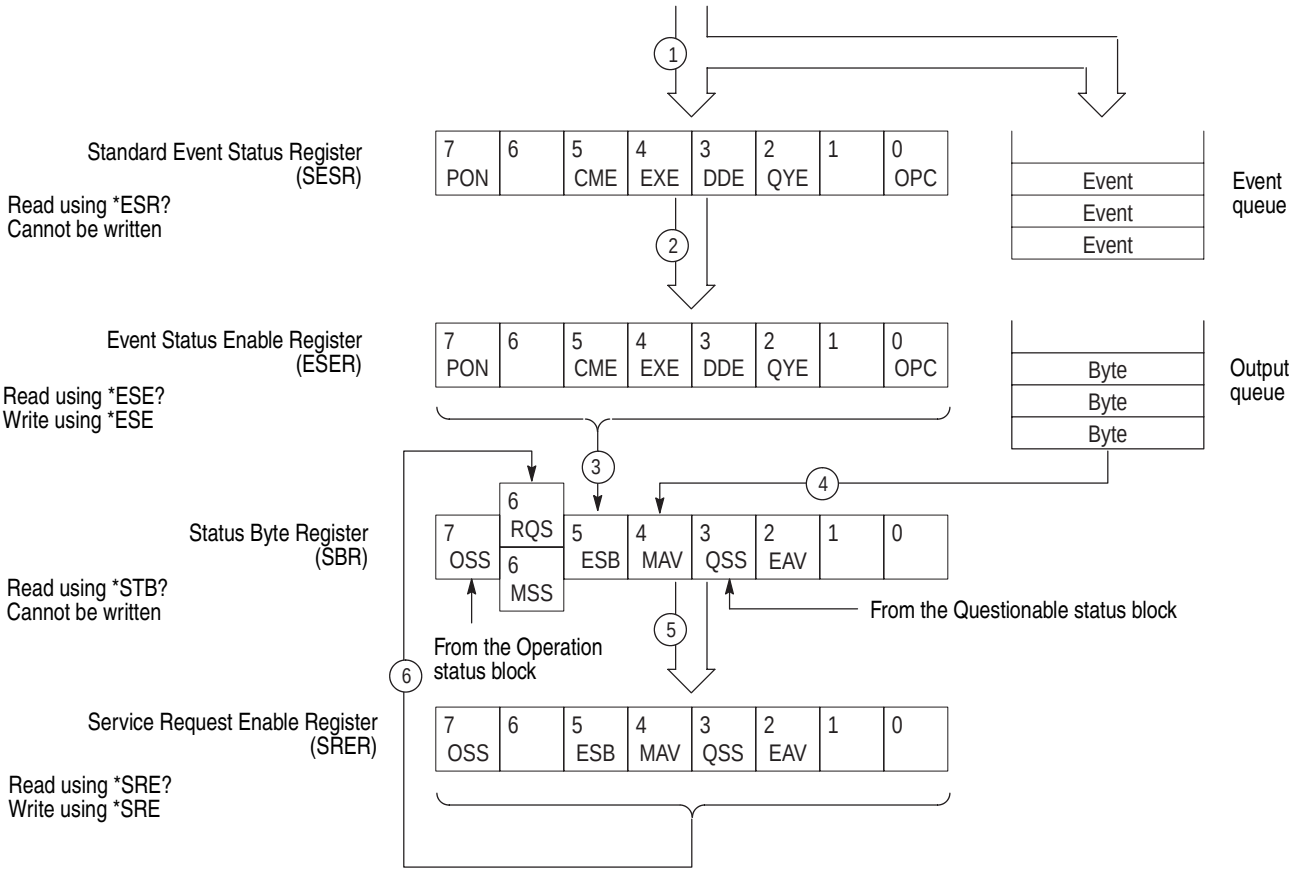



**Figure 3-12: Status and Event processing sequence — Standard/Event status block**

## I/O Status and Event Screen

Figure 3-13 shows the contents of the GPIB status and event reporting system displayed on the status and event screen. Use this procedure to display the screen:

1. Press the **UTILITY** menu button on the front panel. The UTILITY menu appears on the screen.
2. Press the **Status** bottom menu button to display the Status submenu.
3. Press the **SCPI registers** side menu button to display the status and event screen.

The status and event screen displays the registers: SESR, ESER, SBR, SRER, OEVR, and QEVR. Each of these registers is displayed with the decimal equivalent of its contents shown in brackets. All events currently in the queue are listed in the Event Queue area of the display.

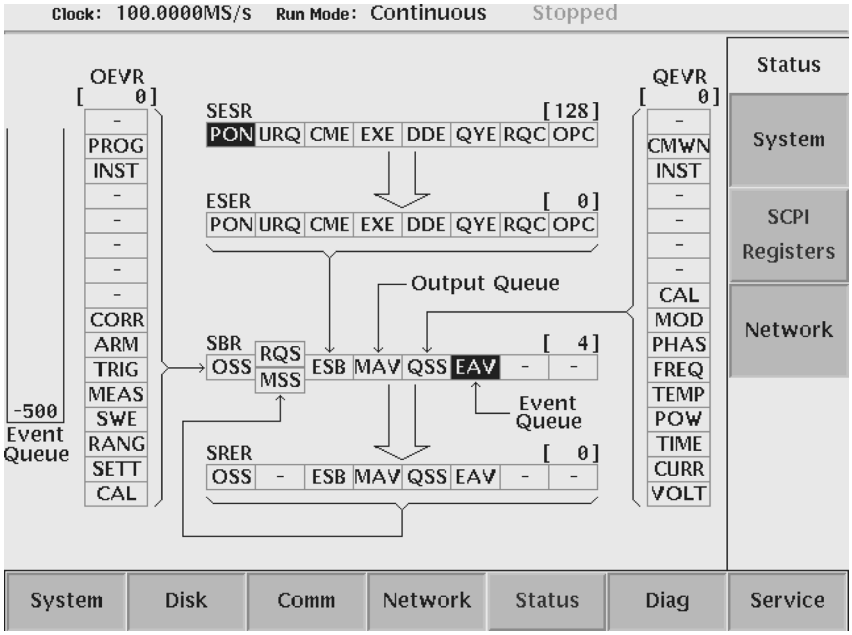



**Figure 3-13: Status and Event screen**

## Synchronizing Execution

All commands used in the waveform generator are designed to be executed in the order in which they are sent from the external controller. The following synchronization commands are included to ensure compliance with the SCPI standard.

```
*WAI
*OPC
*OPC?
```

## Messages

Tables 3-6 through 3-14 show the codes and messages used in the status and event reporting system.

Event codes and messages can be obtained by using the queries `SYSTem:ERRor[:NEXT]?`. Responses are returned in the following format:

```
<event code>,"<event message>"
```

# 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 waveform generator series number.

Table3-5 lists event code definitions. When an error occurs, you can find its error class by checking for the code range in Tables 3-6 through3-14Events in these tables are organized by event class.

**Table 3-5: Definition of event codes**

| Event class | Code range | Description |
|---|---|---|
| No error | 0 | No event or testatus |
| 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

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

**Table 3-6: Command errors**

| Error code | Error message |
|---|---|
| -100 | Command error |
| -101 | Invalid character |
| -102 | Syntax error |
| -103 | Invalid separator |
| -104 | Data type error |
| -105 | GET not allowed |
| -108 | Parameter not allowed |
| -109 | Missing parameter |
| -110 | Command header error |
| -111 | Header separator error |
| -112 | Program mnemonic too long |
| -113 | Undefined 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 |

**Table 3-6: Command errors (cont.)**

| Error code | Error message |
|---|---|
| -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 Errors

These error codes are returned when an error is detected during command execution.

**Table 3-7: Execution errors**

| Error code | Error message |
|---|---|
| -200 | Execution error |
| -201 | Invalid while in local |
| -202 | Settings lost due to RTL |
| -203 | Command protected |
| -210 | Trigger error |
| -211 | Trigger ignored |
| -212 | Arm ignored |
| -213 | Init ignored |
| -214 | Trigger deadlock |
| -215 | Arm deadlock |
| -220 | Parameter error |
| -221 | Settings conflict |
| -222 | Data out of range |
| -223 | Too much data |
| -224 | Illegal parameter value |
| -225 | Out of memory |
| -226 | Lists not same length |
| -230 | Data corrupt or stale |
| -231 | Data questionable |
| -232 | Invalid format |
| -233 | Invalid version |
| -240 | Hardware error |
| -241 | Hardware missing |
| -250 | Mass storage error |
| -251 | Missing mass storage |
| -252 | Missing media |

**Table 3-7: Execution errors (cont.)**

| Error code | Error message |
|---|---|
| -253 | Corrupt media |
| -254 | Media full |
| -255 | Directory full |
| -256 | File name not found |
| -257 | File name error |
| -258 | Media protected |
| -260 | Expression error |
| -261 | Math error in expression |
| -270 | Macro error |
| -271 | Macro syntax error |
| -272 | Macro execution error |
| -273 | Illegal macro label |
| -274 | Macro parameter error |
| -275 | Macro definition too long |
| -276 | Macro recursion error |
| -277 | Macro redefinition not allowed |
| -278 | Macro header not found |
| -280 | Program error |
| -281 | Cannot create program |
| -282 | Illegal program name |
| -283 | Illegal variable name |
| -284 | Program currently running |
| -285 | Program syntax error |
| -286 | Program runtime error |
| -290 | Memory use error |
| -291 | Out of memory |
| -292 | Referenced name does not exist |
| -293 | Referenced name already exists |
| -294 | Incompatible type |

## Device Specific Errors

These error codes are returned when an internal instrument error is detected. This type of error can indicate a hardware problem.

**Table 3-8: Device specific errors**

| Error code | Error message |
|---|---|
| -300 | Device specific error |
| -310 | System error |
| -311 | Memory error |
| -312 | PUD memory lost |
| -313 | Calibration memory lost |
| -314 | Save/recall memory lost |
| -315 | Configuration memory lost |
| -320 | Storage fault |
| -321 | Out of memory |
| -330 | Self–test failed |
| -340 | Calibration failed |
| -350 | Queue overflow |
| -360 | Communication error |
| -361 | Parity error in program message |
| -362 | Framing error in program message |
| -363 | Input buffer overrun |
| -365 | Time out error |

## Query Errors

These error codes are returned in response to an unanswered query.

**Table 3-9: Query errors**

| Error code | Error message |
|---|---|
| -400 | query error |
| -410 | query INTERRUPTED |
| -420 | query UNTERMINATED |
| -430 | query DEADLOCKED |
| -440 | query UNTERMINATED after indefinite response |

## Power–On Events

These events occur when the instrument detects an off to on transition in its power supply.

**Table 3-10: Power–on events**

| Event code | Event message |
|---|---|
| -500 | Power on |

## User Request Events

These events are unused in AWG.

**Table 3-11: User request events**

| Event code | Event message |
|---|---|
| -600 | User request |

## Request Control Events

This event is unused in AWG.

**Table 3-12: Request control events**

| Event code | Event 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.

**Table 3-13: Operation complete events**

| Event code | Event message |
|---|---|
| -800 | Operation complete |

## Device Errors

The error codes in Table 3-14 are unique to AWG710/AWG710B

**Table 3-14: Device errors**

| Error code | Error message |
|---|---|
| 1101 | CH1 Internal Offset calibration failure |
| 1104 | $\overline{\text{CH1}}$ Internal Offset calibration failure |
| 1201 | CH1 Output Offset calibration failure |
| 1204 | $\overline{\text{CH1}}$ Output Offset calibration failure |
| 1301 | CH1 Gain calibration failure |
| 1304 | $\overline{\text{CH1}}$ Gain calibration failure |
| 1401 | CH1 Gain difference calibration failure |
| 1404 | $\overline{\text{CH1}}$ Gain difference calibration failure |
| 1501 | CH1 Direct Output Gain calibration failure |
| 1504 | $\overline{\text{CH1}}$ Direct Output Gain calibration failure |
| 1601 | CH1 Attenuator calibration failure |
| 1604 | $\overline{\text{CH1}}$ Attenuator calibration failure |
| 1611 | CH1 x5dB 1 Attenuator calibration failure |
| 1614 | $\overline{\text{CH1}}$ x5dB 1 Attenuator calibration failure |
| 1621 | CH1 x5dB 2 Attenuator calibration failure |
| 1624 | $\overline{\text{CH1}}$ x5dB 2 Attenuator calibration failure |
| 1631 | CH1 x10dB Attenuator calibration failure |
| 1634 | $\overline{\text{CH1}}$ x10dB Attenuator calibration failure |
| 1641 | CH1 x20dB Attenuator calibration failure |
| 1644 | $\overline{\text{CH1}}$ x20dB Attenuator calibration failure |
| 1701 | CH1 Filter calibration failure |
| 1704 | $\overline{\text{CH1}}$ Filter calibration failure |
| 1711 | CH1 20MHz Filter calibration failure |
| 1714 | $\overline{\text{CH1}}$ 20MHz Filter calibration failure |
| 1721 | CH1 50MHz Filter calibration failure |
| 1724 | $\overline{\text{CH1}}$ 50MHz Filter calibration failure |
| 1731 | CH1 100MHz Filter calibration failure |
| 1734 | $\overline{\text{CH1}}$ 100MHz Filter calibration failure |
| 1741 | CH1 200MHz Filter calibration failure |
| 1744 | $\overline{\text{CH1}}$ 200MHz Filter calibration failure |
| 1801 | CH1 Reference level calibration failure |
| 1804 | $\overline{\text{CH1}}$ Reference level calibration failure |
| 2100 | System failure |

**Table 3-14: Device errors (cont.)**

| Error code | Error message |
|---|---|
| 2101 | Real–time clock power |
| 2102 | Configuration record and checksum status |
| 2103 | Incorrect configuration |
| 2104 | Memory size miscompare |
| 2105 | Fixed–disk drive initialization status |
| 2106 | Time status |
| 2110 | Front panel failure |
| 2111 | Front panel configuration |
| 2112 | Front panel communication |
| 2113 | Front panel RAM |
| 2114 | Front panel ROM |
| 2115 | Front panel A/D |
| 2116 | Front panel timer |
| 2700 | Calibration data failure |
| 2701 | Calibration data not found |
| 2702 | Calibration data checksum |
| 2703 | Calibration data invalid |
| 3000 | Run mode failure |
| 3100 | Run mode control register0 failure |
| 3101 to 3108 | Run mode control1 register bit0 to bit7 |
| 3200 | Run mode control register1 failure |
| 3201 | Run mode control register1 reg0 |
| 3211 | Run mode control register1 reg10 |
| 4000 | Clock failure |
| 4100 | PLL lock/unlock failure |
| 5000 | Sequence memory failure |
| 5100 | Sequence memory data bus failure |
| 5101 to 5132 | Sequence memory data bus bit0 to bit31 |
| 5200 | Sequence memory address bus failure |
| 5201 to 5216 | Sequence memory address bus bit0 to bit15 |
| 5300 | Sequence memory chip cell failure |
| 5301 to 5302 | Sequence memory chip 0 to chip 1 |
| 5350 | Sequence memory chip select failure |
| 5351 to 5352 | Sequence memory chip select 0 to select 1 |
| 6000 | Waveform memory failure |
| 6100 | CH1 Waveform memory data bus failure |
| 6101 to 6132 | CH1 Waveform memory data bus bit0 to bit31 |

**Table 3-14: Device errors (cont.)**

| Error code | Error message |
|---|---|
| 6150 | CH1 Waveform memory chip data bus failure |
| 6151 to 6186 | CH1 Waveform memory chip data bus bit0 to bit35 |
| 6200 | CH1 Waveform memory address bus failure |
| 6201 to 6219 | CH1 Waveform memory address bus bit0 to bit18 |
| 6300 | CH1 Waveform memory chip cell failure |
| 6301 to 6336 | CH1 Waveform memory chip 0 to chip 35 |
| 6350 | CH1 Waveform memory chip select failure |
| 6351 to 6386 | CH1 Waveform memory chip select 0 to select 35 |
| 7000 | Output failure |
| 7100 | Internal offset failure |
| 7101 | CH1 internal offset |
| 7104 | $\overline{\text{CH1}}$ internal offset |
| 7200 | Output offset failure |
| 7201 | CH1 output offset |
| 7204 | $\overline{\text{CH1}}$ output offset |
| 7300 | Arb gain failure |
| 7301 | CH1 Arb gain |
| 7304 | $\overline{\text{CH1}}$ Arb gain |
| 7400 | Direct gain failure |
| 7401 | CH1 Direct gain |
| 7404 | $\overline{\text{CH1}}$ Direct gain |
| 7510 | 5dB 1 attenuator failure |
| 7511 | CH1 1 5dB attenuator |
| 7514 | $\overline{\text{CH1}}$ 1 5dB attenuator |
| 7520 | 5dB 2 attenuator failure |
| 7521 | CH1 5dB 2 attenuator |
| 7524 | $\overline{\text{CH1}}$ 5dB 2 attenuator |
| 7530 | 10dB attenuator failure |
| 7531 | CH1 10dB attenuator |
| 7534 | $\overline{\text{CH1}}$ 10dB attenuator |
| 7540 | 20dB attenuator failure |
| 7541 | CH1 20dB attenuator |
| 7544 | $\overline{\text{CH1}}$ 20dB attenuator |
| 7610 | 20MHz filter failure |
| 7611 | CH1 20MHz filter |
| 7614 | $\overline{\text{CH1}}$ 20MHz filter |
| 7620 | 50MHz filter failure |

**Table 3-14: Device errors (cont.)**

| Error code | Error message |
|---|---|
| 7621 | CH1 50MHz filter |
| 7624 | $\overline{\text{CH1}}$ 50MHz filter |
| 7630 | 100MHz filter failure |
| 7631 | CH1 100MHz filter |
| 7634 | $\overline{\text{CH1}}$ 100MHz filter |
| 7640 | 200MHz filter failure |
| 7641 | CH1 200MHz filter |
| 7644 | $\overline{\text{CH1}}$ 200MHz filter |
| 7700 | Reference level failure |
| 7701 | Reference level failure |
| 7704 | Reference level failure |
| 9111 | Waveform/Sequence load error: waveform memory full |
| 9112 | Waveform/Sequence load error: invalid waveform length |
| 9113 | Waveform/Sequence load error: waveform length too short |
| 9114 | Waveform/Sequence load error: waveform length changed |
| 9121 | Sequence load error: missing file name in sequence |
| 9122 | Sequence load error: too many nesting levels |
| 9123 | Sequence load error: infinite loop in sub–sequence |
| 9124 | Sequence load error: infinite sub–sequence loop |
| 9125 | Sequence load error: max sequence elements exceeded |
| 9126 | Sequence load error: invalid jump address |
| 9127 | Sequence load error: sequence memory full |
| 9128 | Sequence load error: infinite loop and Goto One not allowed |
| 9129 | Sequence load error: infinite loop and Goto <N> not allowed |
| 9151 | Waveform load warning: output disabled in some channels |
| 9152 | Waveform/Sequence output warning: output disabled |

# Examples

# Programming Examples

The floppy disk supplied with the waveform generator contains programming samples to use the GPIB and Ethernet interfaces. These programs are written in Microsoft Visual C++ and Visual BASIC.

Figure 4-1 displays what you need to run the GPIB example programs. GPIB programs run on a PC–compatible system. To use the GPIB interface, your PC–compatible system must be equipped with a National Instruments GPIB board and associated drivers. GPIB programs are also compatible with National Instruments LabVIEW.

The diskette also contains the file *README.TXT*. Refer to the file for details about how to run the programs.

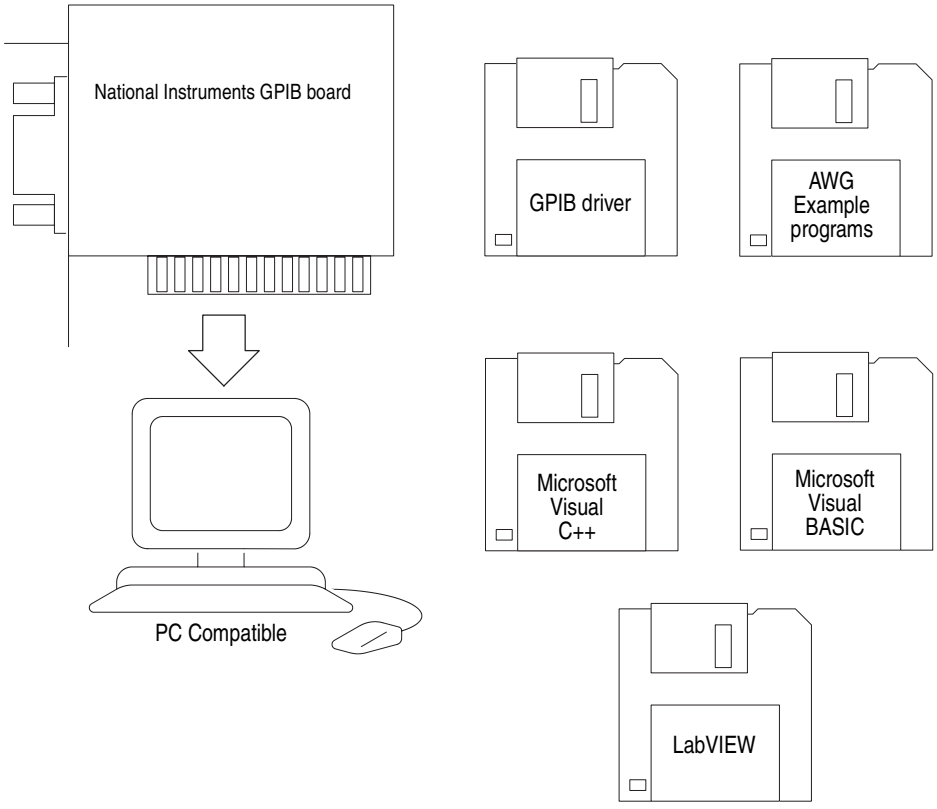



**Figure 4-1: Equipment needed to run the GPIB example programs**

# Appendices

# Appendix A: Character Charts

**Table A-1: The AWG character set**

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL 0 | 16 | space 32 | 0 48 | @ 64 | P 80 | ‘ 96 | p 112 |
| 1 | 1 | 17 | ! 33 | 1 49 | A 65 | Q 81 | a 97 | q 113 |
| 2 | 2 | 18 | ” 34 | 2 50 | B 66 | R 82 | b 98 | r 114 |
| 3 | 3 | 19 | # 35 | 3 51 | C 67 | S 83 | c 99 | s 115 |
| 4 | 4 | 20 | $ 36 | 4 52 | D 68 | T 84 | d 100 | t 116 |
| 5 | 5 | 21 | % 37 | 5 53 | E 69 | U 85 | e 101 | u 117 |
| 6 | 6 | 22 | & 38 | 6 54 | F 70 | V 86 | f 102 | v 118 |
| 7 | 7 | 23 | ’ 39 | 7 55 | G 71 | W 87 | g 103 | w 119 |
| 8 | 8 | 24 | ( 40 | 8 56 | H 72 | X 88 | h 104 | x 120 |
| 9 | HT 9 | 25 | ) 41 | 9 57 | I 73 | Y 89 | i 105 | y 121 |
| A | LF 10 | 26 | * 42 | : 58 | J 74 | Z 90 | j 106 | z 122 |
| B | 11 | ESC 27 | + 43 | ; 59 | K 75 | [ 91 | k 107 | { 123 |
| C | 12 | 28 | , 44 | < 60 | L 76 | \ 92 | l 108 | \| 124 |
| D | CR 13 | 29 | — 45 | = 61 | M 77 | ] 93 | m 109 | } 125 |
| E | 14 | 30 | . 46 | > 62 | N 78 | ^ 94 | n 110 | ~ 126 |
| F | 15 | 31 | / 47 | ? 63 | O 79 | _ 95 | o 111 | rubout 127 |

**Table A-2: ASCII & GPIB code chart**

| B7 B6 B5 / BITS B4 B3 B2 B1 | 0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0 | 1 1 1 |
|---|---|---|---|---|---|---|---|---|
| | CONTROL | | NUMBERS SYMBOLS | | UPPER CASE | | LOWER CASE | |
| 0 0 0 0 | 0 NUL 0 0 | 20 DLE 10 16 | 40 LA0 SP 20 32 | 60 LA16 0 30 48 | 100 TA0 @ 40 64 | 120 TA16 P 50 80 | 140 SA0 ` 60 96 | 160 SA16 p 70 112 |
| 0 0 0 1 | 1 GTL SOH 1 1 | 21 LL0 DC1 11 17 | 41 LA1 ! 21 33 | 61 LA17 1 31 49 | 101 TA1 A 41 65 | 121 TA17 Q 51 81 | 141 SA1 a 61 97 | 161 SA17 q 71 113 |
| 0 0 1 0 | 2 STX 2 2 | 22 DC2 12 18 | 42 LA2 " 22 34 | 62 LA18 2 32 50 | 102 TA2 B 42 66 | 122 TA18 R 52 82 | 142 SA2 b 62 98 | 162 SA18 r 72 114 |
| 0 0 1 1 | 3 ETX 3 3 | 23 DC3 13 19 | 43 LA3 # 23 35 | 63 LA19 3 33 51 | 103 TA3 C 43 67 | 123 TA19 S 53 83 | 143 SA3 c 63 99 | 163 SA19 s 73 115 |
| 0 1 0 0 | 4 SDC EOT 4 4 | 24 DCL DC4 14 20 | 44 LA4 $ 24 36 | 64 LA20 4 34 52 | 104 TA4 D 44 68 | 124 TA20 T 54 84 | 144 SA4 d 64 100 | 164 SA20 t 74 116 |
| 0 1 0 1 | 5 PPC ENQ 5 5 | 25 PPU NAK 15 21 | 45 LA5 % 25 37 | 65 LA21 5 35 53 | 105 TA5 E 45 69 | 125 TA21 U 55 85 | 145 SA5 e 65 101 | 165 SA21 u 75 117 |
| 0 1 1 0 | 6 ACK 6 6 | 26 SYN 16 22 | 46 LA6 & 26 38 | 66 LA22 6 36 54 | 106 TA6 F 46 70 | 126 TA22 V 56 86 | 146 SA6 f 66 102 | 166 SA22 v 76 118 |
| 0 1 1 1 | 7 BEL 7 7 | 27 ETB 17 23 | 47 LA7 ' 27 39 | 67 LA23 7 37 55 | 107 TA7 G 47 71 | 127 TA23 W 57 87 | 147 SA7 g 67 103 | 167 SA23 w 77 119 |
| 1 0 0 0 | 10 GET BS 8 8 | 30 SPE CAN 18 24 | 50 LA8 ( 28 40 | 70 LA24 8 38 56 | 110 TA8 H 48 72 | 130 TA24 X 58 88 | 150 SA8 h 68 104 | 170 SA24 x 78 120 |
| 1 0 0 1 | 11 TCT HT 9 9 | 31 SPD EM 19 25 | 51 LA9 ) 29 41 | 71 LA25 9 39 57 | 111 TA9 I 49 73 | 131 TA25 Y 59 89 | 151 SA9 i 69 105 | 171 SA25 y y79 121 |
| 1 0 1 0 | 12 LF A 10 | 32 SUB 1A 26 | 52 LA10 * 2A 42 | 72 LA26 : 3A 58 | 112 TA10 J 4A 74 | 132 TA26 Z 5A 90 | 152 SA10 j 6A 106 | 172 SA26 z 7A 122 |
| 1 0 1 1 | 13 VT B 11 | 33 ESC 1B 27 | 53 LA11 + 2B 43 | 73 LA27 ; 3B 59 | 113 TA11 K 4B 75 | 133 TA27 [ 5B 91 | 153 SA11 k 6B 107 | 173 SA27 { 7B 123 |
| 1 1 0 0 | 14 FF C 12 | 34 FS 1C 28 | 54 LA12 , 2C 44 | 74 LA28 < 3C 60 | 114 TA12 L 4C 76 | 134 TA28 \ 5C 92 | 154 SA12 l 6C 108 | 174 SA28 ¦ 7C 124 |
| 1 1 0 1 | 15 CR D 13 | 35 GS 1D 29 | 55 LA13 - 2D 45 | 75 LA29 = 3D 61 | 115 TA13 M 4D 77 | 135 TA29 ] 5D 93 | 155 SA13 m 6D 109 | 175 SA29 } 7D 125 |
| 1 1 1 0 | 16 SO E 14 | 36 RS 1E 30 | 56 LA14 . 2E 46 | 76 LA30 > 3E 62 | 116 TA14 N 4E 78 | 136 TA30 ^ 5E 94 | 156 SA14 n 6E 110 | 176 SA30 ~ 7E 126 |
| 1 1 1 1 | 17 SI F 15 | 37 US 1F 31 | 57 LA15 / 2F 47 | 77 UNL ? 3F 63 | 117 TA15 O 4F 79 | 137 UNT - 5F 95 | 157 SA15 o 6F 111 | 177 RUBOUT (DEL) 7F 127 |
| | ADDRESSED COMMANDS | UNIVERSAL COMMANDS | LISTEN ADDRESSES | | TALK ADDRESSES | | SECONDARY ADDRESSES OR COMMANDS | |

**KEY**


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

# Appendix B: GPIB Interface Specification

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

## Interface Functions

Table B-1 lists the GPIB interface functions this instrument implements. Each function is briefly described on the next page.

**Table B-1: GPIB interface function implementation**

| Interface function | Implemented subset | Capability |
|---|---|---|
| Acceptor Handshake (AH) | AH1 | Complete |
| Source Handshake (SH) | SH1 | Complete |
| Talker (T) | T6 | Basic Talker, Serial Poll<br>Unaddress if my–listen–address (MLA)<br>No Talk Only mode |
| Listener (L) | L4 | Basic Listener<br>Unaddress if my talk address (MTA)<br>No Listen Only mode |
| Service Request (SR) | SR1 | Complete |
| Remote/Local (RL) | RL1 | Complete |
| Parallel Poll (PP) | PP0 | None |
| Device Clear (DC) | DC1 | Complete |
| Device Trigger (DT) | DT1 | Complete |
| Controller (C) | C0 | None |
| Electrical Interface | E2 | Three–state driver |

- Acceptor Handshake (AH). Enables a listening device to coordinate data reception. The AH function delays data transfer initiation or termination until the listening device is ready to receive the next data byte.
- Source Handshake (SH). Enables a talking device to support the coordination of data transfer. The SH function controls the initiation and termination of data byte transfers.
- Talker (T). Enables a device to send device–dependent data over the interface. This capability is available only when the device is addressed to talk, and uses a one–byte address.
- Listener (L). Enables a device to receive device–dependent data over the interface. This capability is available only when the device is addressed to listen, and uses a one–byte address.
- Service Request (SR). Enables a device to request service from the controller.
- Remote/Local (RL). Enables a device to select between one of two sources for waveform generator control. It determines whether input information is controlled from the front panel (local control) or by GPIB commands (remote control).
- Device Clear (DC). Enables a device to be cleared or initialized, either individually, or as part of a group of devices.
- Controller (C). Enables a device that has this capability to send its address, universal commands, and addressed commands to other devices over the interface.
- Electrical Interface (E). Identifies the electrical interface driver type. The notation E1 means the electrical interface uses open collector drivers, E2 means the electrical interface uses three–state drivers.

## Interface Messages

Table B-2 shows the standard interface messages the waveform generator supports. Brief function descriptions are provided on the next page.

**Table B-2: AWG standard interface message**

| Message | GPIB |
|---|---|
| DCL | Yes |
| GET | Yes |
| GTL | Yes |
| LLO | Yes |
| PPC | No |
| PPD | No |
| PPE | No |
| PPU | No |
| SDC | Yes |
| SPD | Yes |
| SPE | Yes |
| TCT | No |
| UNL | Yes |
| UNT | Yes |
| Listen Addresses | Yes |
| Talk Addresses | Yes |

- Device Clear (DCL). Will clear (initialize) all devices on the bus that have a device clear function, whether or not the controller has addressed them.
- Group Execute Trigger (GET). Triggers all applicable devices and causes them to initiate their programmed actions.
- Go To Local (GTL). Causes the listen–addressed device to switch from remote to local (front–panel) control.
- Local Lockout (LLO). Disables the return to local function.
- Parallel Poll Configure (PPC). Causes the listen–addressed device to respond to the secondary commands Parallel Poll Enable (PPE) and Parallel Poll Disable (PPD), which are placed on the bus following the PPC command. PPE enables a device with parallel poll capability to respond on a particular data line. PPD disables the device from responding to the parallel poll.
- Select Device Clear (SDC). Clears or initializes all listen–addressed devices.
- Serial Poll Disable (SPD). Changes all devices on the bus from the serial poll state to the normal operating state.
- Serial Poll Enable (SPE). Puts all bus devices that have a service request function into the serial poll enabled state. In this state, each device sends the controller its status byte, instead of its normal output, after the device receives its talk address on the data lines. This function may be used to determine which device sent a service request.
- Take Control (TCT). Allows the controller in charge to pass control of the bus to another controller on the bus.

# Appendix C: Network Interface Specification

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

TCP/IP is used as the network protocol, and the port number is fixed as 4000. Commands can be sent from the application program through the TCP/IP socket interface, and queries can be received through the interface.

The following lists the differences between the GPIB interface and the Ethernet interface.

- The Line Feed (LF) code is needed as a terminator at the end of a message.
- The IEEE 488.1 standard (for instance, Device Clear, Service Request, etc.) is not supported.
- The Message Exchange Control Protocol in the IEEE 488.2 is not supported. However, common commands such as *ESE and the event handling features are supported.
- The Indefinite format (the block start at #0) in the <ARBITRARY BLOCK PROGRAM DATA> of the IEEE 488.2 is not supported.

For detailed information about the programming, refer to the *Sample Programs* disk supplied with this waveform generator.

# Appendix D: SCPI Conformance Information

All commands in the waveform generator are based on SCPI Version 1999.0. Table D-1lists the SCPI commands this waveform generator supports.

**Table D-1: SCPI conformance information**

| Command | | | | | Defined in SCPI 1999.0 | Not defined in SCPI 1999.0 |
|---|---|---|---|---|---|---|
| ABORt | | | | | ✓ | |
| ABSTouch | | | | | | ✓ |
| AWGcontrol | CLOCk | SOURCe(?) | | | | ✓ |
| | DOUTput | [STATe](?) | | | | ✓ |
| | ENHanced | SEQuence | [JMODe](?) | | | ✓ |
| | EVENt | LOGic | [IMMediate] | | | ✓ |
| | | SOFTware | [IMMediate] | | | ✓ |
| | | TABLe | [IMMediate] | | | ✓ |
| | FG | FREQuency | [CW\|FIXed] | (?) | | ✓ |
| | | FUNCtion | [SHAPe] | (?) | | ✓ |
| | | POLarity | (?) | | | ✓ |
| | | PULSe | DCYCle | (?) | | ✓ |
| | | [STATe] | (?) | | | ✓ |
| | | VOLTage [LEVel] | [IMMediate] | [AMPLitude](?) | | ✓ |
| | | VOLTage [LEVel] | [IMMediate] | OFFSet(?) | | ✓ |
| | MIX | [STATe](?) | | | | ✓ |
| | RMODe(?) | | | | | ✓ |
| | RSTATe? | | | | | ✓ |
| | RUN | [IMMediate] | | | | ✓ |
| | SRESTore | | | | | ✓ |
| | SSAVe | | | | | ✓ |
| | STOP | [IMMediate] | | | | ✓ |
| | SYNChronous | ADDRess(?) | | | | ✓ |
| | SYNChronous | CALibration | | | | ✓ |
| | SYNChronous | CONNect(?) | | | | ✓ |
| | SYNChronous | MASTer | [STATe](?) | | | ✓ |
| | SYNChronous | SLAVe | [STATe](?) | | | ✓ |

**Table D-1: SCPI conformance information (cont.)**

| Command | | | | | Defined in SCPI 1999.0 | Not defined in SCPI 1999.0 |
|---|---|---|---|---|---|---|
| CALibration | [ALL](?) | | | | ✓ | |
| DIAGnostic | DATA? | | | | | ✓ |
| | [IMMediate](?) | | | | | ✓ |
| | SELect(?) | | | | | ✓ |
| DISPlay | ENABle(?) | | | | ✓ | |
| | HILight | COLor(?) | | | | ✓ |
| HCOPy | DESTination | | | | ✓ | |
| | DEVice | COLor(?) | | | ✓ | |
| | | LANGuage(?) | | | ✓ | |
| | [IMMediate] | | | | ✓ | |
| | SDUMp | [IMMediate] | | | ✓ | |
| MMEMory | CATalog? | | | | ✓ | |
| | CDIRectory(?) | | | | ✓ | |
| | CLOSe | | | | ✓ | |
| | COPY | | | | ✓ | |
| | DATA | | | | ✓ | |
| | DELete | | | | ✓ | |
| | FEED(?) | | | | ✓ | |
| | INITialize | | | | ✓ | |
| | MDIRectory | | | | ✓ | |
| | MSIS(?) | | | | | ✓ |
| | MOVE | | | | ✓ | |
| | NAME(?) | | | | ✓ | |
| | OPEN | | | | ✓ | |
| OUTPut | FILTer | [LPASs] | FREQuency(?) | | ✓ | |
| | ISTate(?) | | | | | ✓ |
| | MARKer | [STSTe](?) | | | | ✓ |
| | [STATe](?) | | | | ✓ | |
| SOURce | FREQuency | [CW\|FIXed](?) | | | ✓ | |
| | FUNCtion | USER(?) | | | | ✓ |
| | MARKer | VOLTage | [LEVel] [IMMediate] | HIGH(?) | | ✓ |
| | | VOLTage | [LEVel] [IMMediate] | LOW(?) | | ✓ |
| | ROSCillator | SOURce(?) | | | ✓ | |
| | VOLTage | [LEVel] | [IMMediate] | [AMPLitude](?) | ✓ | |
| | | | | OFFSet(?) | ✓ | |

**Table D-1: SCPI conformance information (cont.)**

| Command | | | | | | Defined in SCPI 1999.0 | Not defined in SCPI 1999.0 |
|---|---|---|---|---|---|---|---|
| STATus | OPERation | [EVENt]? | | | | ✓ | |
| | | CONDition? | | | | ✓ | |
| | | ENABle(?) | | | | ✓ | |
| | PRESet | | | | | ✓ | |
| | QUEStionable | [EVENt]? | | | | ✓ | |
| | | CONDition? | | | | ✓ | |
| | | ENABle(?) | | | | ✓ | |
| SYSTem | BEEPer | [IMMediate] | | | | ✓ | |
| | COMMunicate | LAN DHCP [CLIent] LEASe TIME(?) | | | | | ✓ |
| | | LAN DHCP [CLIent] [STATe](?) | | | | | ✓ |
| | | LAN | FTP [SERVer] | [STATe](?) | | | ✓ |
| | | | | VERSion(?) | | | ✓ |
| | | | GATeway | ADDRess(?) | | | ✓ |
| | | | NFS | TLIMit(?) | | | ✓ |
| | | | PING? | | | | ✓ |
| | | | RDEVice | ADDRess(?) | | | ✓ |
| | | | | FSYStem(?) | | | ✓ |
| | | | | NAME(?) | | | ✓ |
| | | | | PROTocol(?) | | | ✓ |
| | | | | [STATe](?) | | | ✓ |
| | | | [SELF] | ADDRess(?) | | | ✓ |
| | | | | MADDress(?) | | | ✓ |
| | | | | SMASk(?) | | | ✓ |
| | DATE(?) | | | | | ✓ | |
| | ERRor | [NEXT]? | | | | ✓ | |
| | KDIRection(?) | | | | | | ✓ |
| | KEYBoard | [TYPE](?) | | | | | ✓ |
| | KLOCk(?) | | | | | ✓ | |
| | SECurity | IMMediate | | | | ✓ | |
| | TIME(?) | | | | | ✓ | |
| | UPTime? | | | | | | ✓ |
| | VERSion? | | | | | ✓ | |

**Table D-1: SCPI conformance information (cont.)**

| Command | | | Defined in SCPI 1999.0 | Not defined in SCPI 1999.0 |
|---|---|---|---|---|
| TRIGger | [SEQuence] | [IMMediate] | ✓ | |
| | | IMPedance(?) | | ✓ |
| | | LEVel(?) | ✓ | |
| | | POLarity(?) | | ✓ |
| | | SLOPe(?) | ✓ | |
| | | SOURce(?) | ✓ | |
| | | TIMer(?) | ✓ | |
| *CLS | | | ✓ | |
| *ESR(?) | | | ✓ | |
| *ESR? | | | ✓ | |
| *IDN? | | | ✓ | |
| *OPC(?) | | | ✓ | |
| *OPT? | | | | ✓ |
| *RST | | | ✓ | |
| *SRE(?) | | | ✓ | |
| *STB? | | | ✓ | |
| *TST? | | | ✓ | |
| *WAI? | | | ✓ | |

# Appendix E: Factory Initialization Settings

The following tables lists the commands affected by factory initialization. Table E-1 on page E-1 lists commands for the AWG710.

The SYStem:SECurity:IMMediate command initializes all the settings as shown below; the *RST command has no effect on the Status commands and the SYSTem:COMMunicate:LAN commands.

**Table E-1: Factory initialization settings**

| Header | Default settings |
|---|---|
| **AWGcontrol commands** | |
| AWGControl:CLOCk:SOURce | INTernal |
| AWGControl:DOUTput[1] | 0 |
| AWGControl:ENHanced:SEQuence[:JMODe] | TABLe |
| AWGControl:FG:FREQuency[:CW\|:FIXed] | 20.0MHz |
| AWGControl:FG1:FUNCtion[:SHAPe] | SINusoid |
| AWGControl:FG1:POLarity | POSitive |
| AWGControl:FG1:PULSe:DCYCle | 10.0 |
| AWGControl:FG[:STATe] | 0 |
| AWGControl:FG1:VOLTage[:LEVel][:IMMediate][:AMPLitude] | 1.000 |
| AWGControl:FG1:VOLTage[:LEVel][:IMMediate]:OFFSet | 0.000 |
| AWGControl:MIX[:STATe] | 0 |
| AWGControl:RMODe | CONTinuous |
| AWGControl:SYNChronous:ADDRess | no affect |
| AWGControl:SYNChronous:CONNect | 0 |
| AWGControl:SYNChronous:MASTer[:STATe] | 0 |
| AWGControl:SYNChronous:SLAVe[:STATe] | 0 |
| **Diagnostic commands** | |
| DIAGnostic:SELect | ALL |
| **Display commands** | |
| DISPlay:ENABle | 1 |
| DISPlay:HILight:COLor | 0 |
| **Hardcopy commands** | |
| HCOPy:DEVice:COLor | 0 |

**Table E-1: Factory initialization settings (cont.)**

| Header | Default settings |
|---|---|
| HCOPy:DEVice:LANGuage | BMP |
| **MMemory commands** | |
| MMEMory:CDIRectory | "/" |
| MMEMory:FEED | "HCOP" |
| MMEMory:MSIS | "MAIN" |
| MMEMory:NAME | "HARDCOPY","MAIN" |
| **Output commands** | |
| OUTPut[1]:FILTer[:LPASs]:FREQuency | 9.9E+37 |
| OUTPut[1]:ISTAte | 0 |
| OUTPut[1][:STATe] | 0 |
| OUTPut[1]:MARKer[:STATe] | 0 |
| **Source commands** | |
| [SOURce[1]]:COMBine:FEED | "" (null) |
| [SOURce[1]]:FREQuency[:CW\|:FIXed] | 1.0000000E+8 |
| [SOURce[1]]:FUNCtion:USER | "" (null),"MAIN" |
| [SOURce[1]]:MARKer[1\|2]:VOLTage[:LEVel] [:IMMediate]:HIGH | 2.00 (AWG710)<br>1.00 (AWG710B) |
| [SOURce[1]]:MARKer[1\|2]:VOLTage[:LEVel] [:IMMediate]:LOW | 0.00 |
| [SOURce[1]]:ROSCillator:SOURce | INTernal |
| [SOURce[1]]:VOLTage[:LEVel][:IMMediate] [:AMPLitude] | 1.000 |
| [SOURce[1]]:VOLTage[:LEVel][:IMMediate]: OFFSet | 0.000 |
| **Status commands** | |
| *ESE [1] | 0 |
| *PSC [1] | 1 |
| *SRE [1] | 0 |
| STATus:OPERation:ENABle [1] | 0 |
| STATus:QUEStionable:ENABle [1] | 0 |

**Table E-1: Factory initialization settings (cont.)**

| Header | Default settings |
|---|---|
| **System commands** | |
| SYSTem:COMMunicate:LAN:DHCP[:CLIent]:LEASe:TIME | 28800 |
| SYSTem:COMMunicate:LAN:DHCP[:CLIent][:STATe] [1] | 0 |
| SYSTem:COMMunicate:LAN:FTP[:SERVer][:STATe] [1] | 0 |
| SYSTem:COMMunicate:LAN:FTP[:SERVer]:VERSion [1] | STANdard |
| SYSTem:COMMunicate:LAN:GATeway[1\|2\|3]:ADDRess [1] | "" (null) |
| SYSTem:COMMunicate:LAN:NFS:TLIMit | 300 |
| SYSTem:COMMunicate:LAN:RDEVice<x>:ADDRess [1] | "" (null) |
| SYSTem:COMMunicate:LAN:RDEVice<x>:FSYStem [1] | "" (null) |
| SYSTem:COMMunicate:LAN:RDEVice<x>:NAME [1] | "NET<x>" |
| SYSTem:COMMunicate:LAN:RDEVice<x>:PROTocol [1] | NFS |
| SYSTem:COMMunicate:LAN:RDEVice<x>[:STATe] [1] | 0 |
| SYSTem:COMMunicate:LAN[:SELF]:ADDRess [1] | "" (null) |
| SYSTem:COMMunicate:LAN[:SELF]:SMASk [1] | "" (null) |
| SYSTem:KDIRection | FORWard |
| SYSTem:KEYBoard[:TYPE] | ASCii |
| SYSTem:KLOCK | 0 |
| **Trigger commands** | |
| TRIGger[:SEQuence]:IMPedance | 5.0E+1 |
| TRIGger[:SEQuence]:LEVel | 1.4 |
| TRIGger[:SEQuence]:POLarity | POSitive |
| TRIGger[:SEQuence]:SLOPe | POSitive |
| TRIGger[:SEQuence]:SOURce | EXTernal |
| TRIGger[:SEQuence]:TIMer | 1.00E-1 |
| **GPIB** | |
| Remote Control [1] | GPIB |
| GPIB Address [1] | 1 |
| GPIB Confoguration [1] | Talk/Listen |

**1 These commands are not affected by the *RST command.**

# Glossary and Index

# Glossary

**ASCII**

Acronym for the American Standard Code for Information Interchange. Controllers transmit commands to the instrument using ASCII character encoding.

**Address**

A 7–bit code that identifies an instrument on the communication bus. The instrument must have a unique address for the controller to recognize and transmit commands.

**BNF (Backus–Naur Form)**

A standard notation system for command syntax diagrams. The syntax diagrams in this manual use BNF notation.

**Controller**

A computer or other device that sends commands to and accepts responses from the digitizing oscilloscope.

**EOI**

A mnemonic referring to the control line “End or Identify” on the GPIB interface bus. One of the two possible end–of–message terminators.

**EOM**

A generic acronym referring to the end–of–message terminator. The end–of–message terminator can be either an EOI or the ASCII code for line feed (LF).

**GPIB**

Acronym for General Purpose Interface Bus, the common name for the communications interface system defined in IEEE Std 488.

**IEEE**

Acronym for the Institute for Electrical and Electronic Engineers.

**QuickC**

A computer language (distributed by Microsoft) that is based on C.

**SCPI**

Acronym for Standard Commands for Programmable Instruments.

# Index

## A



## B



## C

## D



## E



## F



## G

## H



## I



## L



## M



## N



## O



## P

## Q



## R



## S

## T



## U



## W