Programmer Manual

SONY:
Tektronix

AWG510 & AWG520
Arbitrary Waveform Generator

071-0100-00

This document supports firmware version 1.0 and above.
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

## Preface

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>vii</td>
</tr>
</tbody>
</table>

## Getting Started

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started</td>
<td>1–1</td>
</tr>
<tr>
<td>Overview of the Manual</td>
<td>1–1</td>
</tr>
<tr>
<td>Setting Up Remote Communications</td>
<td>1–3</td>
</tr>
</tbody>
</table>

## Syntax and Commands

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Syntax</td>
<td>2–1</td>
</tr>
<tr>
<td>SCPI Commands and Queries</td>
<td>2–2</td>
</tr>
<tr>
<td>IEEE 488.2 Common Commands</td>
<td>2–9</td>
</tr>
<tr>
<td>Constructed Mnemonics</td>
<td>2–10</td>
</tr>
<tr>
<td>Syntax Diagrams</td>
<td>2–12</td>
</tr>
<tr>
<td>Command Groups</td>
<td>2–13</td>
</tr>
<tr>
<td>Functional Groups</td>
<td>2–13</td>
</tr>
<tr>
<td>Command Quick Reference</td>
<td>2–14</td>
</tr>
<tr>
<td>Command Summaries</td>
<td>2–16</td>
</tr>
<tr>
<td>Command Descriptions</td>
<td>2–25</td>
</tr>
<tr>
<td>Retrieving Response Messages</td>
<td>2–91</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>2–93</td>
</tr>
<tr>
<td>Data File</td>
<td>2–93</td>
</tr>
<tr>
<td>Data Transfer Procedures</td>
<td>2–99</td>
</tr>
</tbody>
</table>

## Status and Events

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status and Event Reporting</td>
<td>3–1</td>
</tr>
<tr>
<td>Status Reporting Structure</td>
<td>3–1</td>
</tr>
<tr>
<td>Registers</td>
<td>3–4</td>
</tr>
<tr>
<td>Status Registers</td>
<td>3–4</td>
</tr>
<tr>
<td>Enable Registers</td>
<td>3–8</td>
</tr>
<tr>
<td>Queues</td>
<td>3–10</td>
</tr>
<tr>
<td>Status and Event Processing Sequence</td>
<td>3–11</td>
</tr>
<tr>
<td>I/O Status and Event Screen</td>
<td>3–13</td>
</tr>
<tr>
<td>Synchronizing Execution</td>
<td>3–14</td>
</tr>
<tr>
<td>Messages</td>
<td>3–14</td>
</tr>
<tr>
<td>Messages and Codes</td>
<td>3–15</td>
</tr>
<tr>
<td>Command Errors</td>
<td>3–16</td>
</tr>
<tr>
<td>Execution Errors</td>
<td>3–18</td>
</tr>
<tr>
<td>Device Specific Errors</td>
<td>3–20</td>
</tr>
<tr>
<td>Query Errors</td>
<td>3–21</td>
</tr>
<tr>
<td>Power-On Events</td>
<td>3–21</td>
</tr>
<tr>
<td>Operation Complete Events</td>
<td>3–21</td>
</tr>
<tr>
<td>Device Errors</td>
<td>3–22</td>
</tr>
</tbody>
</table>
Table of Contents

Programming Examples

Programming Examples .................................................. 4-1

Appendices

Appendix A: Character Charts ......................................... A-1
Appendix B: Reserved Words .......................................... B-1
Appendix C: GPIB Interface Specification .......................... C-1
Appendix D: SCPI Conformance Information ...................... D-1
Appendix E: Factory Initialization Settings ....................... E-1

Glossary and Index
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2-1</td>
<td>BNF symbols and meanings</td>
<td>2-1</td>
</tr>
<tr>
<td>Table 2-2</td>
<td>Query response examples</td>
<td>2-3</td>
</tr>
<tr>
<td>Table 2-3</td>
<td>Parameter types used in syntax descriptions</td>
<td>2-4</td>
</tr>
<tr>
<td>Table 2-4</td>
<td>Functional groups in the command set</td>
<td>2-13</td>
</tr>
<tr>
<td>Table 2-5</td>
<td>AWG Control commands</td>
<td>2-16</td>
</tr>
<tr>
<td>Table 2-6</td>
<td>Calibration commands</td>
<td>2-16</td>
</tr>
<tr>
<td>Table 2-7</td>
<td>Diagnostic commands</td>
<td>2-17</td>
</tr>
<tr>
<td>Table 2-8</td>
<td>Display commands</td>
<td>2-17</td>
</tr>
<tr>
<td>Table 2-9</td>
<td>Hardcopy commands</td>
<td>2-18</td>
</tr>
<tr>
<td>Table 2-10</td>
<td>Mass storage in AWG500 series</td>
<td>2-19</td>
</tr>
<tr>
<td>Table 2-11</td>
<td>Mass Memory commands</td>
<td>2-19</td>
</tr>
<tr>
<td>Table 2-12</td>
<td>Output channel</td>
<td>2-20</td>
</tr>
<tr>
<td>Table 2-13</td>
<td>Output commands</td>
<td>2-20</td>
</tr>
<tr>
<td>Table 2-14</td>
<td>Available sources and markers</td>
<td>2-20</td>
</tr>
<tr>
<td>Table 2-15</td>
<td>Source commands</td>
<td>2-21</td>
</tr>
<tr>
<td>Table 2-16</td>
<td>Status commands</td>
<td>2-22</td>
</tr>
<tr>
<td>Table 2-17</td>
<td>Synchronization commands</td>
<td>2-22</td>
</tr>
<tr>
<td>Table 2-18</td>
<td>System commands</td>
<td>2-23</td>
</tr>
<tr>
<td>Table 2-19</td>
<td>Trigger commands</td>
<td>2-24</td>
</tr>
<tr>
<td>Table 2-20</td>
<td>Selecting run modes</td>
<td>2-28</td>
</tr>
<tr>
<td>Table 2-21</td>
<td>Self-Test Routines</td>
<td>2-36</td>
</tr>
<tr>
<td>Table 3-1</td>
<td>SBR bit functions</td>
<td>3-5</td>
</tr>
<tr>
<td>Table 3-2</td>
<td>SESR bit functions</td>
<td>3-6</td>
</tr>
<tr>
<td>Table 3-3</td>
<td>OCR bit functions</td>
<td>3-7</td>
</tr>
<tr>
<td>Table 3-4</td>
<td>QCR bit functions</td>
<td>3-7</td>
</tr>
<tr>
<td>Table 3-5</td>
<td>Definition of event codes</td>
<td>3-15</td>
</tr>
<tr>
<td>Table 3-6</td>
<td>Command errors</td>
<td>3-16</td>
</tr>
<tr>
<td>Table 3-7</td>
<td>Execution errors</td>
<td>3-18</td>
</tr>
<tr>
<td>Table 3-8</td>
<td>Device specific errors</td>
<td>3-20</td>
</tr>
<tr>
<td>Table 3-9</td>
<td>Query errors</td>
<td>3-21</td>
</tr>
<tr>
<td>Table 3-10</td>
<td>Power-on events</td>
<td>3-21</td>
</tr>
<tr>
<td>Table 3-11</td>
<td>Operation complete events</td>
<td>3-21</td>
</tr>
<tr>
<td>Table 3-12</td>
<td>Device errors</td>
<td>3-22</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>A-1</td>
<td>Table A-1: The AWG500 character set</td>
<td>A-1</td>
</tr>
<tr>
<td>A-2</td>
<td>Table A-2: ASCII &amp; GPIB code chart</td>
<td>A-2</td>
</tr>
<tr>
<td>C-1</td>
<td>Table C-1: GPIB interface function implementation</td>
<td>C-1</td>
</tr>
<tr>
<td>C-3</td>
<td>Table C-2: AWG500 standard interface message</td>
<td>C-3</td>
</tr>
<tr>
<td>D-1</td>
<td>Table D-1: SCPI conformance information</td>
<td>D-1</td>
</tr>
<tr>
<td>E-1</td>
<td>Table E-1: Factory initialization settings</td>
<td>E-1</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1–1: Common message elements ............................................. 1–1
Figure 1–2: Functional groupings and an alphabetical list of commands ......................................................... 1–2
Figure 1–3: The status reporting function provides for event driven programs ......................................................... 1–2
Figure 1–4: The floppy disk supplied with the waveform generator ................................................................. 1–3
Figure 1–5: GPIB connector location ................................................. 1–3
Figure 1–6: How to stack GPIB connectors ........................................ 1–4
Figure 1–7: Typical GPIB network configurations ........................................ 1–5
Figure 1–8: Selecting the GPIB configuration and address ............... 1–6
Figure 2–1: Example of SCPI subsystem hierarchy tree ................. 2–2
Figure 2–2: Example of abbreviating a command ........................ 2–5
Figure 2–3: Example of chaining commands and queries ............... 2–6
Figure 2–4: Example of omitting root and lower-level nodes in a chained message ........................................... 2–6
Figure 2–5: Typical syntax diagrams .............................................. 2–12
Figure 2–6: Retrieving response messages ....................................... 2–91
Figure 2–7: The Waveform File format ......................................... 2–94
Figure 2–8: The Pattern File format ............................................. 2–95
Figure 2–9: The Sequence File format ......................................... 2–96
Figure 2–10: The Code Convert File format ............................... 2–98
Figure 3–1: Error and Event handling process overview ............... 3–2
Figure 3–2: The Status Byte Register (SBR) ............................. 3–5
Figure 3–3: The Standard Event Status Register (SES) ................. 3–6
Figure 3–4: The Operation Condition Register (OCR) .............. 3–7
Figure 3–5: The Questionable Condition Register (QCR) ............ 3–7
Figure 3–6: The Event Status Enable Register (ESER) ............. 3–8
Figure 3–7: The Service Request Enable Register (SREP) .......... 3–9
Figure 3–8: The Operation Enable Register (OENR) ................. 3–9
Figure 3–9: The Questionable Enable Register (QREN) ............. 3–9
Figure 3–10: Status and Event processing sequence — Operation status block ............................................. 3–11
Figure 3–11: Status and Event processing sequence — Questionable status block ............................................. 3–11
Figure 3–12: Status and Event processing sequence — Standard/Event status block ............................................. 3–12
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-13</td>
<td>Status and Event screen</td>
<td>3-13</td>
</tr>
<tr>
<td>4-1</td>
<td>Equipment needed to run the example programs</td>
<td>4-1</td>
</tr>
</tbody>
</table>
Preface

This is the Programmer Manual for the AWG500 Series Arbitrary Waveform Generators. This manual provides information on operating the instrument over a General Purpose Interface Bus (GPIB) interface.

This manual provides the following information:

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

Related Manuals

Other documentation for the waveform generator includes:

- The *AWG510 & AWG520 Arbitrary Waveform Generator User Manual* (Tektronix part number 071-0099-XX) describes the operation of the instrument.
- The *AWG510 & AWG520 Arbitrary Waveform Generator Service Manual* (Tektronix part number 071-0101-XX) provides information for maintaining and servicing the waveform generator.
Getting Started

The AWG500 series waveform generators have GPIB 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 sections:

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

Overview of the Manual

The information contained in each major section of this manual is described below.

Syntax and Commands

The *Syntax and Commands* chapter 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 shows a syntax diagram and command parts as described in the *Command Syntax* subsection.

![Syntax diagram](image)

*Figure 1–1: Common message elements*
Chapter 2 also describes the effect of each command and provides examples of how you might use it. The Command Groups section provides a list by functional area. The Command Descriptions section arranges commands alphabetically (Figure 1–2).

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

**Status and Events**

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 illustrates the basic operation of this system.

The Status and Events chapter starting on page 3–1 describes how to use the status reporting function that conforms to the SCPI and IEEE–488.2 in your programs.

![Diagram of status reporting function](image)

**Figure 1–3: The status reporting function provides for event driven programs**
Programming Examples

The Programming Examples chapter starting on page 4–1 describes some example waveform generator programs. The floppy disk supplied with the waveform generator manual (Figure 1–4) has a Microsoft Visual C++ and Visual BASIC source-code version of each program.

![AWG Example Programs]

Figure 1–4: The floppy disk supplied with the waveform generator

Setting Up Remote Communications

For remote operations, the instrument must be connected to the controller. The following topics provide connection information.

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 (available from Tektronix as part number 012-0991-xx) to the GPIB connector.

![GPIB connector]

Figure 1–5: GPIB connector location
If needed, you can stack GPIB connectors as shown in Figure 1–6.

Figure 1–6: How to stack GPIB connectors
**GPIB Requirements**

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

- Assign a unique device address to each device on the bus. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices to a bus.
- Turn on at least two-thirds of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration as shown in Figure 1–7. Do not use loop or parallel configurations.

![GPIB Device Diagram](image)

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

*Appendix C: Interface Specifications* gives more information on 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. Once you have set these parameters, you can control the waveform generator through the GPIB interface.

1. Push the UTILITY button to display the Utility screen.

2. Push the Comm bottom menu button.

3. 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.

4. Move the cursor to the GPIB Address field using the down (↓) arrow button, then set the address using either the general purpose knob or the keypad.

![Figure 1–8: Selecting the GPIB configuration and address](image)

The waveform generator is set up for bidirectional communication with your controller. If you want to isolate the waveform generator from the bus, do the following:

- Select Off Bus in the GPIB Configuration field. This selection disables all communication with the controller.
Syntax and Commands
Command Syntax

This section contains general information on 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 the Backus-Naur Form (BNF) notation. Table 2–1 defines the standard BNF symbols.

Table 2–1: BNF symbols and meanings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;  &gt;</td>
<td>Defined element</td>
</tr>
<tr>
<td>: : =</td>
<td>Is defined as</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>{    }</td>
<td>Group; one element is required</td>
</tr>
<tr>
<td>[    ]</td>
<td>Optional; can be omitted</td>
</tr>
<tr>
<td>. . .</td>
<td>Previous element(s) may be repeated</td>
</tr>
<tr>
<td>(    )</td>
<td>Comment</td>
</tr>
</tbody>
</table>
SCPI Commands and Queries

The AWG500 Series Arbitrary Waveform Generator uses a command language based on the SCPI standard. The SCPI (Standard Commands for Programmable Instruments) standard was created by a consortium to provide guidelines for remote programming of instruments. These guidelines provide a consistent programming environment for instrument control and data transfer. This environment uses defined programming messages, instrument responses, and data formats that operate across all SCPI instruments, regardless of manufacturer.

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

```
        TRIGger
        /   \
Sequence \   / \\
        \ /  \
       LEvel POLarity SOURCE
```

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, TRIGger is the root node and SEQuence, LEvel, POLarity, and SOURce are lower-level nodes. To create a SCPI command, start with the root node TRIGger and move down the tree structure adding nodes until you reach the end of a branch. Most commands and some queries have parameters; you must include a value for these parameters. If you specify a parameter value that is out of range, the parameter will be set to a default value. The command descriptions, which start 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.

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.

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.

Table 2–2: Query response examples

<table>
<thead>
<tr>
<th>Query</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURce:VOLtage:AMPLitude?</td>
<td>1.000</td>
</tr>
<tr>
<td>AWGControl:RMODe?</td>
<td>CONT</td>
</tr>
</tbody>
</table>

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 AWG500 series command set and some are defined by ANSI/IEEE 488.2-1987.

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

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>arbitrary block</td>
<td>A block of data bytes</td>
<td>#512234xxxx... where 5 indicates that the following 5 digits (12234) specify the length of the data in bytes; xxxx... indicates the data or #0xxxx..&lt;LF&gt;&lt;&amp;EOI&gt;</td>
</tr>
<tr>
<td>boolean</td>
<td>Boolean numbers or values</td>
<td>ON or 1 OFF or 0</td>
</tr>
<tr>
<td>discrete</td>
<td>A list of specific values</td>
<td>MIN, MAX</td>
</tr>
<tr>
<td>binary</td>
<td>Binary numbers</td>
<td>#B0110</td>
</tr>
<tr>
<td>octal</td>
<td>Octal numbers</td>
<td>#Q75, #Q3</td>
</tr>
<tr>
<td>hexadecimal ²</td>
<td>Hexadecimal numbers (0-9, A-F)</td>
<td>#HAA, #H1</td>
</tr>
<tr>
<td>NR1 ²,³ numeric</td>
<td>Integers</td>
<td>0, 1, 15, -1</td>
</tr>
<tr>
<td>NR2 ² numeric</td>
<td>Decimal numbers</td>
<td>1.2, 3.141516, -6.5</td>
</tr>
<tr>
<td>NR3 ² numeric</td>
<td>Floating point numbers</td>
<td>3.1415E-9, -16.1E5</td>
</tr>
<tr>
<td>NRt ² numeric</td>
<td>Flexible decimal number that may be type NR1, NR2, or NR3</td>
<td>See NR1, NR2, NR3 examples</td>
</tr>
<tr>
<td>string ⁴</td>
<td>Alphanumeric characters (must be within quotation marks)</td>
<td>“Testing 1, 2, 3”</td>
</tr>
</tbody>
</table>

1. Defined in ANSI/IEEE 488.2 as “Arbitrary Block Program Data.”
3. Some commands and queries will accept an octal or hexadecimal value even though the parameter type is defined as NR1.
4. Defined in ANSI/IEEE 488.2 as “String Response Data” and “String Program Data.”
Special Characters

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

Abbreviating Commands, Queries, and Parameters

You can abbreviate most SCPI commands, queries, and parameters to an accepted short form. This manual shows these 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.

```
<table>
<thead>
<tr>
<th>Long form of a command</th>
<th>SOURce2:FREQuency 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum information needed for accepted short form</td>
<td></td>
</tr>
<tr>
<td>Accepted short form of a command</td>
<td>SOUR2:FREQ 100</td>
</tr>
</tbody>
</table>
```

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 AWG500 series will default to “1” if no suffix is used. In Figure 2–2, the “2” of “SOUR2” indicates that the command is directed to the second channel.
Chaining Commands and Queries

You can chain several commands or queries together into a single message. To create a chained message, first create a command or query, 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.

```
:SOUR:FREQ:FIX 100;:OUTP:STAT ON;:SOUR:VOLT:AMPL?;:TRIG:SEQ:LEV?
```

First command  Second command  First query  Second query

The response from this chained message might be:

```
100;1.2
```

Response from first query  Response from second query

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.

```
```

Identical root and lower-level nodes

```
:TRIG:SEQ:LEV 250;SLOP POS;SOUR EXT
```

First command  Additional commands (omitted the root nodes)

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

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

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

<table>
<thead>
<tr>
<th>SI prefix *</th>
<th>p/p</th>
<th>n/N</th>
<th>u/U</th>
<th>m/M</th>
<th>k/K</th>
<th>m/M</th>
<th>g/G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding power</td>
<td>$10^{-12}$</td>
<td>$10^{-9}$</td>
<td>$10^{-6}$</td>
<td>$10^{-3}$</td>
<td>$10^{3}$</td>
<td>$10^{6}$</td>
<td>$10^{9}$</td>
</tr>
</tbody>
</table>

* Note that the prefix m/M indicates $10^{-3}$ when the decimal numeric argument denotes voltage or time, but $10^{6}$ when it denotes frequency.
**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 250MHz

  is the same as
  
  output:filter:lpass:frequency 250mhz

  and
  
  OUTPUT:filter:LPASS:frequency 250MHz

**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 250MHz
  
  incorrect:  OUTPUT: FILTER: LPASS: FREQUENCY 250MHZ
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. 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 these mnemonic in the command just as you do any other mnemonic. Mnemonic ranges can be presented in either 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 [SOURce[1|2|5]]:FUNCTION:USER, the output source mnemonic could be any of the following: SOURce1, SOURce2, or SOURce5. Therefore, a valid usage of this command would be: SOURce5:FUNCTION:USER.

MNEMonic<\eta>. The value of <\eta> 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

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURce1</td>
<td>CH1 signal of the waveform generator</td>
</tr>
<tr>
<td>SOURce2</td>
<td>CH2 signal of the waveform generator (AWG520 only)</td>
</tr>
<tr>
<td>SOURce3</td>
<td>Not used</td>
</tr>
<tr>
<td>SOURce4</td>
<td>Not used</td>
</tr>
<tr>
<td>SOURce5</td>
<td>Digital data of the pattern generator (Option 03)</td>
</tr>
<tr>
<td>SOURce6</td>
<td>Not used</td>
</tr>
<tr>
<td>SOURce7</td>
<td>Internal noise generator signal</td>
</tr>
<tr>
<td>SOURce8</td>
<td>External input signal</td>
</tr>
</tbody>
</table>

### Output Channel Mnemonics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPut1</td>
<td>CH1 output from the waveform generator</td>
</tr>
<tr>
<td>OUTPut2</td>
<td>CH2 output from the waveform generator (AWG520 only)</td>
</tr>
<tr>
<td>OUTPut3</td>
<td>Not used</td>
</tr>
<tr>
<td>OUTPut4</td>
<td>Not used</td>
</tr>
<tr>
<td>Symbol</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OUTPUT5</td>
<td>Digital data output from the pattern generator (Option 03)</td>
</tr>
<tr>
<td>OUTPUT6</td>
<td>Not used</td>
</tr>
<tr>
<td>OUTPUT7</td>
<td>Output from the noise generator</td>
</tr>
</tbody>
</table>

**Direct D/A Output Mnemonics**

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOUTPUT1</td>
<td>Direct output from CH1 D/A converter</td>
</tr>
<tr>
<td>DOUTPUT2</td>
<td>Direct output from CH2 D/A converter (AWG520 only)</td>
</tr>
</tbody>
</table>

**Marker Mnemonics**

Commands specify the marker to use as a mnemonic in the header.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARKer1</td>
<td>The signal for the marker 1</td>
</tr>
<tr>
<td>MARKer2</td>
<td>The signal for the marker 2</td>
</tr>
</tbody>
</table>

**Remote Device Mnemonics**

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDEVice1</td>
<td>Network drive 1</td>
</tr>
<tr>
<td>RDEVice2</td>
<td>Network drive 2</td>
</tr>
<tr>
<td>RDEVice3</td>
<td>Network drive 3</td>
</tr>
</tbody>
</table>

**Gateway Mnemonics**

Commands specify the gateway to use as a mnemonic in the header.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATeway1</td>
<td>Gateway 1</td>
</tr>
<tr>
<td>GATeway2</td>
<td>Gateway 2</td>
</tr>
<tr>
<td>GATeway3</td>
<td>Gateway 3</td>
</tr>
</tbody>
</table>
Syntax Diagrams

The syntax of each command and query is explained by both syntax diagrams and BNF notation. Figure 2–5 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 one and only one of the paths must be taken in the command. (See the top diagram of Figure 2–5.)
- A loop around an element(s) shows the element can be repeated. (See the middle diagram.)
- A path around a group of elements shows that those elements are optional. (See the bottom diagram.)

**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 2–5.

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) 1995.0 and IEEE Std 488.2–1987 except where noted.

Functional Groups

Table 2-4 lists the functional groups into which the waveform generator commands are classified.

<table>
<thead>
<tr>
<th>Group</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWG Control</td>
<td>Control operating mode</td>
</tr>
<tr>
<td>Calibration</td>
<td>Perform calibration</td>
</tr>
<tr>
<td>Diagnostic</td>
<td>Control self-test routines</td>
</tr>
<tr>
<td>Display</td>
<td>Control the presentation of information on the front panel display</td>
</tr>
<tr>
<td>Hardcopy</td>
<td>Dump the whole display into the file on the mass storage</td>
</tr>
<tr>
<td>Mass Memory</td>
<td>Control file operations on the mass storage</td>
</tr>
<tr>
<td>Output</td>
<td>Control the characteristics of the waveform output port</td>
</tr>
<tr>
<td>Source</td>
<td>Set waveform and marker output parameters, such as frequency and level</td>
</tr>
<tr>
<td>Status</td>
<td>Set and query the registers and queues of the reporting system</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Control operation complete and pending command execution</td>
</tr>
<tr>
<td>System</td>
<td>Control miscellaneous instrument functions such as LAN, security, and time</td>
</tr>
<tr>
<td>Trigger</td>
<td>Synchronize the waveform generator actions with events</td>
</tr>
</tbody>
</table>
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:DOU TigOutput-xx:STATE (7)
AWGControl:EVENTLOGic:IMMediate (7)
AWGControl:RMODE (7)
AWGControl:RSTate (7)
AWGControl:RUN:IMMediate (7)
AWGControl:SRSStore (7)
AWGControl:SSAvE (7)
AWGControl:STOP:IMMediate (7)

Calibration commands:
*CAL? (7)
CALibration:ALL (7)

Diagnostic commands
DIAGnostic:DATA? (7)
DIAGnostic:IMMediate (7)
DIAGnostic:SELect (7)
*TST? (7)

Display commands
DISPlay:BRIGHTness (7)

Hardcopy commands
HCOPY:DESTination (7)
HCOPY:DEVice:LANguage (7)
HCOPY:IMMediate (7)
HCOPY:SDUMp:IMMediate (7)

Mass memory commands
MEMory:CATalog? (7)
MEMory:CDIRectory (7)
MEMory:CLOSE (7)
MEMory:COPY (7)
MEMory:DATA (7)
MEMory:DELete (7)
MEMory:FEED (7)
MEMory:INItialize (7)
MEMory:MDIRectory (7)
MEMory:MSIS (7)
MEMory:MOVE (7)
MEMory:NAME (7)
MEMory:OPEN (7)

Output commands
OUTPut-xx:FLTer:LPASs:FREQuency (7)
OUTPut-xx:STATe (7)
OUTPut1:STATe (7)

Source commands
SOURce:COMBine:FEED (7)
SOURce:COMBine:FREQuency:CMpl:FIXed (7)
SOURce:COMBine:FUnction:USER (7)
SOURce:COMBine:MARKer<y>:DELay (7)
SOURce:COMBine:MARKer<y>:VOLTage:LEVEL:IMMediate:HIG (7)
SOURce:COMBine:MARKer<y>:VOLTage:LEVEL:IMMediate:LOW (7)
SOURce7:POWer:VOLTage:LEVEL:IMMediate:AMPLitude (7)
SOURce:MEASure:RESistance:SOURce (7)
SOURce:MEASure:VOLTage:LEVEL:IMMediate:AMPLitude (7)
SOURce6:VOLTage:LEVEL:IMMediate:HIG (7)

SOURce6:VOLTage:LEVel:IMMediate:LOW (7)
SOURce6:VOLTage:LEVel:IMMediate:OFFSet (7)

Status commands
*CLS (7)
*ESE (7)
*ESR? (7)
*PSC (7)
*SRE (7)
STATus:OPERation:EVENT? (7)
STATus:OPERation:CONDition? (7)
STATus:OPERation:ENABLE (7)
STATus:QUEStionable:EVENT? (7)
STATus:QUEStionable:CONDition? (7)
STATus:QUEStionable:ENABLE (7)
STATus:PRESet (7)
STATus:QUEue:NEXT? (7)
*STB? (7)

Synchronization commands
*OPC (7)
*WAI (7)

System commands
*IDN? (7)
*OPT? (7)
*RST (7)
SYSTem:BEep:IMMediate (7)
SYSTem:COMMunicate:LAN:FTP:SERVer:STATe (7)
SYSTem:COMMunicate:LAN:GATEway-xx:ADDRess (7)
SYSTem:COMMunicate:LAN:PING? (7)
SYSTem:COMMunicate:LAN:RDEVice-xx:ADDRess (7)
SYSTem:COMMunicate:LAN:RDEVice-xx:FSIZEt (7)
SYSTem:COMMunicate:LAN:RDEVice-xx:NAME (7)
SYSTem:COMMunicate:LAN:RDEVice-xx:PROTocol (7)
SYSTem:COMMunicate:LAN:RDEVice-xx:STATe (7)
SYSTem:COMMunicate:LAN:SELf:ADDRess (7)
SYSTem:COMMunicate:LAN:SELf:SMASK (7)
SYSTem:DATE (7)
SYSTem:ERRor? (7)
SYSTem:KDIRection (7)
SYSTem:KEYBoard:TYPE (7)
SYSTem:LOCK (7)
SYSTem:SECunity:IMMediate (7)
SYSTem:TIME (7)
SYSTem:UPTIme? (7)
SYSTem:VERSIon? (7)

Trigger commands
ABORT (7)
*TRG (7)
TRIGger:SEQUence:IMMediate (7)
TRIGger:SEQUence:IMPedance (7)
TRIGger:SEQUences:LEVEL (7)
TRIGger:SEQUence:POLarity (7)
TRIGger:SEQUence:SLOPe (7)
TRIGger:SEQUence:SOURce (7)
TRIGger:SEQUence:TIMer (7)
Command Summaries

Tables 2–6 through 2–19 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**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGControl:DOU[Tput[1</td>
<td>2] [:STAtE] (?)</td>
</tr>
<tr>
<td>AWGControl:EVENT[:LOGic][:IMMediate]</td>
<td>Generate the event signal for logic jump</td>
</tr>
<tr>
<td>AWGControl:RMODE (?)</td>
<td>Select the run mode, such as triggered or gated</td>
</tr>
<tr>
<td>AWGControl:RSTate?</td>
<td>Query the current running status</td>
</tr>
<tr>
<td>AWGControl:RUN[:IMMediate]</td>
<td>Enable the output from CH&lt;&gt;&lt;&gt;</td>
</tr>
<tr>
<td>AWGControl:SREStore</td>
<td>Restore the settings from the specified file</td>
</tr>
<tr>
<td>AWGControl:SSAVe</td>
<td>Store the settings to the specified file</td>
</tr>
<tr>
<td>AWGControl:STOP[:IMMediate]</td>
<td>Stop the output from CH&lt;&gt;&lt;&gt;</td>
</tr>
</tbody>
</table>

**Calibration Commands**

The Calibration commands calibrate the waveform generator.

**Table 2–6: Calibration commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CAL?</td>
<td>Perform calibration</td>
</tr>
<tr>
<td>CALibration[:ALL] (?)</td>
<td>Perform calibration</td>
</tr>
</tbody>
</table>
Diagnostic Commands

The Diagnostic commands control self-test diagnostic routines.

Table 2-7: Diagnostic commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAGnostic:DATA?</td>
<td>Query results of self-test</td>
</tr>
<tr>
<td>DIAGnostic[:IMMediate] (?)</td>
<td>Start the self-test</td>
</tr>
<tr>
<td>DIAGnostic:SElect (?)</td>
<td>Select the self-test routine</td>
</tr>
<tr>
<td>*TST?</td>
<td>Perform self-test</td>
</tr>
</tbody>
</table>

Display Commands

The Display commands control the presentation of textual information on the front panel display.

Table 2-8: Display commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY:BRIGHTness (?)</td>
<td>Control brightness of the display</td>
</tr>
</tbody>
</table>
Hardcopy Commands

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

The hardcopy commands used in this application do not conform to the 1995 SCPI hardcopy standard. (The 1995 SCPI standards state that the `MMEMory:OPEN` and `MMEMory:CLOSE` commands are to be 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 "SAMPLE1.BMP"
MMEMory:OPEN
HCOPy:DESTination "MMEM"
HCOPy
MMEM:CLOSE
```

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

```
MMEMory:NAME "SAMPLE1.BMP"
HCOPy
```

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

Table 2-9: Hardcopy commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCOPy:DESTination</td>
<td>Set the destination</td>
</tr>
<tr>
<td>HCOPy:Device:LANGUAGE (?)</td>
<td>Select the data format</td>
</tr>
<tr>
<td>HCOPy[:IMMediate]</td>
<td>Initiate the plot or print immediately</td>
</tr>
<tr>
<td>HCOPy:SDUMP[:IMMediate]</td>
<td>Plot or print the whole display</td>
</tr>
</tbody>
</table>
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 AWG500 series

<table>
<thead>
<tr>
<th>String argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>Internal hard disk drive</td>
</tr>
<tr>
<td>FLOP or FLOPPY</td>
<td>Internal floppy disk drive</td>
</tr>
<tr>
<td>NET1</td>
<td>Network drive 1</td>
</tr>
<tr>
<td>NET2</td>
<td>Network drive 2</td>
</tr>
<tr>
<td>NET3</td>
<td>Network drive 3</td>
</tr>
</tbody>
</table>

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 extension.

Table 2-11: Mass Memory commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMEMory:CATalog?</td>
<td>Query information on the mass storage media</td>
</tr>
<tr>
<td>MMEMory:CDIRectory (?)</td>
<td>Change the default directory for a file system</td>
</tr>
<tr>
<td>MMEMory:CLOSE</td>
<td>Close the file specified in NAME</td>
</tr>
<tr>
<td>MMEMory:COPY</td>
<td>Copy an existing file to a new file</td>
</tr>
<tr>
<td>MMEMory:DATA</td>
<td>Load data into the file</td>
</tr>
<tr>
<td>MMEMory:DELETE</td>
<td>Remove a file</td>
</tr>
<tr>
<td>MMEMory:FEED (?)</td>
<td>Feed data into the file specified in NAME</td>
</tr>
<tr>
<td>MMEMory:INITialize</td>
<td>Initialize the specified mass storage</td>
</tr>
<tr>
<td>MMEMory:MDIRectory</td>
<td>Make a directory</td>
</tr>
<tr>
<td>MMEMory:MSIS (?)</td>
<td>Select the current mass storage</td>
</tr>
<tr>
<td>MMEMory:MOVE</td>
<td>Move an existing file to another file</td>
</tr>
<tr>
<td>MMEMory:NAME (?)</td>
<td>Set the file name to be opened or closed</td>
</tr>
<tr>
<td>MMEMory:OPEN</td>
<td>Open the file specified in NAME</td>
</tr>
</tbody>
</table>
Output Commands

The Output commands control the characteristics of the waveform output port. In Table 2-13, OUTPut<x> refers to the waveform output channel, where <x> represents related channel number as shown in Table 2-12.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPut1</td>
<td>CH1 output of the waveform generator</td>
</tr>
<tr>
<td>OUTPut2</td>
<td>CH2 output of the waveform generator</td>
</tr>
<tr>
<td>OUTPut3</td>
<td>Not used</td>
</tr>
<tr>
<td>OUTPut4</td>
<td>Not used</td>
</tr>
<tr>
<td>OUTPut5</td>
<td>Output of the pattern generator</td>
</tr>
<tr>
<td>OUTPut6</td>
<td>Not used</td>
</tr>
<tr>
<td>OUTPut7</td>
<td>Output of the noise generator</td>
</tr>
</tbody>
</table>

Table 2-13: Output commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPut[1</td>
<td>2]:FILTER[::LPASs]:FREQuency (?)</td>
</tr>
<tr>
<td>OUTPut[1</td>
<td>2</td>
</tr>
<tr>
<td>OUTPut[1]:ISTate (?) (AWG510)</td>
<td>Set the inverted output on or off</td>
</tr>
</tbody>
</table>

Source Commands

The Source commands set waveform and marker output parameters, such as frequency and level. SOURce<x> and MARKer<y> in these commands have the meanings as shown in Table 2-14 below.

Table 2-14: Available sources and markers

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURce1</td>
<td>CH1 waveform</td>
</tr>
<tr>
<td>SOURce2</td>
<td>CH2 waveform</td>
</tr>
<tr>
<td>SOURce3</td>
<td>Not used</td>
</tr>
<tr>
<td>SOURce4</td>
<td>Not used</td>
</tr>
</tbody>
</table>
### Table 2-14: Available sources and markers (Cont.)

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURce5</td>
<td>Pattern generator signal</td>
</tr>
<tr>
<td>SOURce6</td>
<td>Not used</td>
</tr>
<tr>
<td>SOURce7</td>
<td>Noise generator signal</td>
</tr>
<tr>
<td>SOURce8</td>
<td>External input signal</td>
</tr>
<tr>
<td>MARKer1</td>
<td>Marker 1 signal</td>
</tr>
<tr>
<td>MARKer2</td>
<td>Marker 2 signal</td>
</tr>
</tbody>
</table>

### Table 2-15: Source commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SOURce[1]:]COMBine:FEED (?)</td>
<td>Add noise or external signal to the output</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>SOURce7:POWer[:LEVEL] [:IMMediate][:AMPLitude] (?)</td>
<td>Set the level for the noise generator output</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2]:]VOLTage[:LEVEL] [:IMMediate][:AMPLitude] (?)</td>
</tr>
<tr>
<td>SOURCe5:VOLTage[:LEVEL] [:IMMediate]:HIGH (?)</td>
<td>Set the high level of a digital pattern signal</td>
</tr>
<tr>
<td>SOURCe5:VOLTage[:LEVEL] [:IMMediate]:LOW (?)</td>
<td>Set the low level of a digital pattern signal</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2]:]VOLTage[:LEVEL] [:IMMediate]:OFFSet (?)</td>
</tr>
</tbody>
</table>
**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 2–16, refer to *Status and Event Reporting* on page 3–1.

**Table 2–16: Status commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CLS</td>
<td>Clear all the event registers and queues</td>
</tr>
<tr>
<td>*ESE (?)</td>
<td>Set and query ESER</td>
</tr>
<tr>
<td>*ESR?</td>
<td>Query SESR</td>
</tr>
<tr>
<td>*PSC (?)</td>
<td>Set power-on status clear flag</td>
</tr>
<tr>
<td>*SRE (?)</td>
<td>Set and query SRER</td>
</tr>
<tr>
<td>STATus:OPERation[:EVENT]?</td>
<td>Query the contents of OEVr</td>
</tr>
<tr>
<td>STATus:OPERation:CONDITION?</td>
<td>Query the contents of OCR</td>
</tr>
<tr>
<td>STATus:OPERation:ENABLE (?)</td>
<td>Set the enable mask of OENR</td>
</tr>
<tr>
<td>STATus:QUESTionable[:EVENT]?</td>
<td>Query the contents of QEVr</td>
</tr>
<tr>
<td>STATus:QUESTionable:CONDITION?</td>
<td>Query the contents of OCR</td>
</tr>
<tr>
<td>STATus:QUESTionable:ENABLE (?)</td>
<td>Set the enable mask of QENR</td>
</tr>
<tr>
<td>STATus:PRESet</td>
<td>Preset OENR and QENR</td>
</tr>
<tr>
<td>STATus:QUEue[:NEXT]?</td>
<td>Query the next item from the error/event queue</td>
</tr>
<tr>
<td>*STB?</td>
<td>Query SBR</td>
</tr>
</tbody>
</table>

**Synchronization Commands**

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

**Table 2–17: Synchronization commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*OPC (?)</td>
<td>Generate or return the operation complete message</td>
</tr>
<tr>
<td>*WAI</td>
<td>Hold off all commands until all pending operations complete</td>
</tr>
</tbody>
</table>
System Commands

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

Table 2-18: System commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*IDN?</td>
<td>Query ID information about the waveform generator</td>
</tr>
<tr>
<td>*OPT?</td>
<td>Query installed options</td>
</tr>
<tr>
<td>*RST</td>
<td>Reset the waveform generator</td>
</tr>
<tr>
<td>SYSTEM:BEEP[:IMMediate]</td>
<td>Generate an audible tone</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN:FTP [:SERVer][:STATE] (?)</td>
<td>Control the FTP server function</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :GATeway[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :PING?</td>
<td>Execute PING test for the specified IP address</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :RDEVICE[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :RDEVICE[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :RDEVICE[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :RDEVICE[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN :RDEVICE[1</td>
<td>2</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN[:SELF]:ADDRess (?)</td>
<td>Set IP address of the waveform generator</td>
</tr>
<tr>
<td>SYSTEM:COMMunicate:LAN[:SELF]:SMASK (?)</td>
<td>Set the subnet mask of the waveform generator</td>
</tr>
<tr>
<td>SYSTEM:DATE (?)</td>
<td>Set the internal calendar</td>
</tr>
<tr>
<td>SYSTEM:ERROR?</td>
<td>Query the next entry from the waveform generator's error/event queue</td>
</tr>
<tr>
<td>SYSTEM:KDIRection (?)</td>
<td>Set the direction of cursor movement controlled by the general purpose knob</td>
</tr>
<tr>
<td>SYSTEM:KEYBoard[:TYPE] (?)</td>
<td>Select the keyboard type</td>
</tr>
<tr>
<td>SYSTEM:KLOCK (?)</td>
<td>Lock the front panel and keyboard</td>
</tr>
<tr>
<td>SYSTEM:SECurity:IMMediate</td>
<td>Destroy all data and settings for security</td>
</tr>
<tr>
<td>SYSTEM:TIME (?)</td>
<td>Set the internal clock</td>
</tr>
<tr>
<td>SYSTEM:UPTime?</td>
<td>Query elapsed time from the power-on</td>
</tr>
<tr>
<td>SYSTEM:VERSION?</td>
<td>Query the SCPI version number</td>
</tr>
</tbody>
</table>
**Trigger Commands**

The Trigger commands synchronize the waveform generator actions with events.

**Table 2–19: Trigger commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td>Reset the trigger system</td>
</tr>
<tr>
<td>*TRG</td>
<td>Generate the trigger event</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence][:IMMediate]</td>
<td>Immediately trigger the sequence operation</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:IMPedance (?)</td>
<td>Select the input impedance of the external trigger</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:LEVEL (?)</td>
<td>Set the trigger level</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:POLarity (?)</td>
<td>Select the polarity of the trigger signal</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:SLOPe (?)</td>
<td>Select the slope of the trigger signal</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:SOURCE (?)</td>
<td>Select the source for the event detector</td>
</tr>
<tr>
<td>TRIGGER[:SEQuence]:TIMer (?)</td>
<td>Set the period of the internal clock</td>
</tr>
</tbody>
</table>
Command Descriptions

This subsection lists each command and query in the AWG500 series Arbitrary Waveform Generator command set alphabetically. 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 subsection fully spells out headers, mnemonics, and arguments with the minimal 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 those commands that can be used as either a command or a query; the symbol "?" follows those commands that can only be a query. Commands that are command-only or query-only are noted as such.
**ABORT (No Query Form)**

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.

**AWGControl:DOUTput[1|2][:STATe] (?)**

Supplies raw output of the waveform generator D/A converter for the specified channel. The settings of SOURce:VOLTage and OUTPut:FILTer commands are ignored.

- **Group**: AWG Control
- **Related Commands**: SOURce:VOLTage command group, OUTPut:FILTer command group
- **Syntax**: `AWGControl:DOUTput[1|2][:STATe] { OFF | ON | <NRI> }`

`AWGControl:DOUTput[1|2][:STATe]?`
Arguments  OFF or <NR1> = 0 provides the D/A converter output normally.
            ON or <NR1> ≠ 0 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:EVENT[:LOGic][:IMMediate] (No Query Form)

Generates a trigger event for the “logic jump” specified in the sequence file. This
has the same effect as pushing 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:RMODE (?)

Selects the mode used to output waveforms or sequences.

Group  AWG Control

Related Commands  ABORT,TRIGger[:SEQUence][:IMMediate],*TRG

Syntax  AWGControl:RMODE { CONTinuous | TRIGgered | GATed | ENHanced }
          AWGControl:RMODE?
Arguments

You can select the modes listed in Table 2–20.

Table 2–20: Selecting run modes

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTinuous</td>
<td>Sets the continuous mode, which continuously outputs the waveform. The external trigger, including FORCE TRIGGER button and the corresponding remote commands, have no effect.</td>
</tr>
<tr>
<td>TRIGgered</td>
<td>Sets the triggered mode, which outputs one waveform cycle for each trigger.</td>
</tr>
<tr>
<td>GATed</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>- The FORCE TRIGGER button remains pressed</td>
</tr>
<tr>
<td></td>
<td>- A valid external gate signal remains input</td>
</tr>
<tr>
<td></td>
<td>- The TRIGGER[:SEQUENCE][:IMMediate] or *TRG command has been executed but a ABORT command has not yet been issued</td>
</tr>
<tr>
<td>ENHanced</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:RSTate? (Query Only)**

Returns the current running status.

**Group**

AWG Control

**Syntax**

AWGControl:RSTate?
Command Descriptions

**AWGControl**

**Arguments** None

**Returns** `<NR>`

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

**Examples**

```
AWGControl::RSTate?
```

might return the following response:

1

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

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

**Group** AWG Control

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

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

```
AWGControl::RUN[:IMMediate]
```

**Arguments** None

**Examples**

```
AWGControl::RUN[:IMMediate]
```

starts the output of a waveform or a sequence.

**AWGControl::SREStore (No Query Form)**

Restores the settings from the specified file.

**Group** AWG Control
Related Commands

AWGControl:SSAVE, MMEMory:CDIRector, 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
- FLOPy: 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","FLOPy"
restores the settings from the file SAMPLE1.SET on the floppy disk.

AWGControl:SSAVE (No Query Form)

Stores the current settings to the specified file.

Group

AWG Control

Related Commands

AWGControl:SREStore, MMEMory:CDIRector, 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
- FLOPy: Internal floppy disk drive
- NET1, NET2, or NET3: Network drive 1, 2, or 3 (specified with the SYSTem:COMMunicate:LAN commands)
Examples

\texttt{AWGControl:SAVE "SAMPLE1.SET","FLOppy"}

stores the current settings to the file SAMPLE1.SET on the floppy disk.

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

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**

\texttt{AWGControl:RUN[:IMMediate], *TRG}

**Syntax**

\texttt{AWGControl:STOP[:IMMediate]}

![Diagram of AWGControl:STOP[:IMMediate]]

**Arguments**

None

**Examples**

\texttt{AWGControl:STOP[:IMMediate]}

stops the output of a waveform.

**CAL? (Query Only)**

The \texttt{*CAL?} query performs an internal calibration and returns status that indicates whether the waveform generator completes 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 \texttt{CALibration [:ALL]?} query.

**NOTE. Up to 15 seconds are 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 detected
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.** Up to 15 seconds are 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
<NRI>
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 detected.

*CLS (No Query Form)

Clears all the event registers and queues, which are used in the waveform generator status and event reporting system. For more details, refer to Section 3, Status and Events.

Group
Status

Syntax
*CLS

Arguments
None

Examples
*CLS
clears all the event registers and queues.
**DIAGnostic:DATA? (Query Only)**

Returns the results of self-test.

**Group**
Diagostic

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

**Syntax**
DIAGnostic:DATA?

**Arguments**
None

**Returns**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NR1&gt;</td>
<td>Termination code</td>
</tr>
<tr>
<td>0</td>
<td>Terminated without error.</td>
</tr>
<tr>
<td>-330</td>
<td>Self-test failed.</td>
</tr>
</tbody>
</table>

**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**
Diagostic

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

**Syntax**
DIAGnostic[:IMMediate]

DIAGnostic[:IMMediate]?
DIAGnostic

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 (?)

Selects the self-test routine(s).

Group

- Diagnostic

Related Commands

- DIAGnostic[::IMMediate]

Syntax

DIAGnostic:SELect { ALL | OUTPut | RMODE | ROSe illator | SMEMory | SYSTem | WMEMory }

DIAGnostic:SELect?
Arguments
You can select the following self-test routines:

Table 2-21: Self-Test Routines

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Checks all routines that follow</td>
</tr>
<tr>
<td>AOUTput</td>
<td>Checks the analog output unit</td>
</tr>
<tr>
<td>ROSEIlator</td>
<td>Checks the reference oscillator unit</td>
</tr>
<tr>
<td>SMEMory</td>
<td>Checks the sequence memory</td>
</tr>
<tr>
<td>SYStem</td>
<td>Checks the system unit, such as the system memory</td>
</tr>
<tr>
<td>TRIGger</td>
<td>Checks the trigger unit</td>
</tr>
<tr>
<td>WMEMory</td>
<td>Checks the waveform memory</td>
</tr>
</tbody>
</table>

At *RST, this parameter is set to ALL.

Examples
DIAGnostic:SELECT WMEMory; IMMEDIATE
executes the waveform memory self-test routine.

DISPlay:BRIghtness (?)

Controls the intensity of the display.

Group  Display

Syntax
DISPlay:BRIghtness <NRf>
DISPlay:BRIghtness?

Arguments
<NRF> ranges from 0 to 1, where 1 is full intensity and 0 is fully blanked.

Examples
DISPlay:BRIghtness 0.8
sets the intensity of the display to 80% of maximum intensity.
**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 Section 3 *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)

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?

!*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)

Sets the hardcopy destination. For the waveform generator, the destination is fixed to MMEMory (mass memory), and 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>

!*COPY :DESTination <space> <data_handle>

Arguments <data_handle>::=<string>
where <string> is fixed to "MMEMory" for the waveform generator.
Examples  HCOPY:DESTination "MMEM"
           sets the hardcopy destination to a file specified with the MMEMory:NAME command.

HCOPY:DEVice:LANGuage (?)

Sets the hardcopy data format.

Group  Hardcopy

Related Commands  HCOPY[:IMMEDIATE]

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)

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
Command Descriptions

Related Commands
HCOPY:DESTination, HCOPY:SDUMP[:IMMEDIATE]

Syntax
HCOPY[:IMMEDIATE]

Arguments
None

Examples
HCOPY:IMMEDIATE
starts hardcopy output.

HCOPY:SDUMP[:IMMEDIATE] (No Query Form)
Initiates a screen dump of the whole display. For the waveform generator, this is the same as the HCOPY[:IMMEDIATE] command.

Group
Hardcopy

Syntax
HCOPY:SDUMP[:IMMEDIATE]

Arguments
None

Examples
MMEMory:NAME "SAMPLE1.BMP"; HCOPY:SDUMP:IMMEDIATE
prints the whole display to the file SAMPLE1.BMP.

*IDN? (Query Only)
Returns identification information for the waveform generator.

Group
System
**Syntax**  
*IDN?*

- **Arguments** None
- **Returns**  
  `<manufacturer>,<model>,<serial_number>,<firmware_level>`  
  where  
  `<manufacturer>::=SONY/TEK`  
  `<model>::={( AWG510 | AWG520 ) }`  
  `<serial_number>::=0`  
  `<firmware_level>::=SCPI:95.0 OS:x.y USR:x'.y'`

- **Examples**  
  *IDN?* might return SONY/TEK,AWG520,0,SCPI:95.0 OS:1.0 USR:1.0

**MMEMory:CATalog? (Query Only)**

- Returns information on the current contents and state of the mass storage media.
- **Group** Mass Memory
- **Related Commands** MMEMory: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 \(<NR1>, <NR1>[,<file_entry>]...\)

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, \(<NR1> = 0\).

\(<file_entry>::= "<file_name>, <file_type>, <file_size>"\)

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

\(\text{MMEMory:CATalog? "MAIN"}\)

might return the following response:

\(546322, 3878625, "\text{SAMPLE1.WFM,}, 2948\)
Examples

MME\textsc{mory:CDIRectory} "/AWG/WORK1"
changes the default directory to /AWG/WORK1.

\textbf{MME\textsc{mory:CLOSE} (No Query Form)}

Closes the file specified in the \textsc{MME}\textsc{mory:NAME} command. This command is included only for compatibility with the SCPI standard and may not be used.

\textbf{Group} Mass Memory

\textbf{Related Commands} \textsc{MME\textsc{mory:NAME, MME\textsc{mory:OPEN}}}

\textbf{Syntax} \textsc{MME\textsc{mory:CLOSE}}

\begin{center}
\begin{tikzpicture}
\node (memory) {\textsc{memory}};
\node (close) at (memory -| close) {\textsc{CLOSE}};
\node (space) at (memory -| space) {\text{space}};
\node (source) at (space -| source) {\text{<file\_source>}};
\node (destination) at (source -| destination) {\text{<file\_destination>}};
\draw (memory) -- (close);
\draw (space) -- (source);
\end{tikzpicture}
\end{center}

\textbf{Arguments} None

\textbf{Examples} \textsc{MME\textsc{mory:NAME "TEST1.WFM";CLOSE}}
closes the file \textsc{TEST1.WFM}.

\textbf{MME\textsc{mory:COPY} (No Query Form)}

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

\textbf{Group} Mass Memory

\textbf{Related Commands} \textsc{MME\textsc{mory:CDIRectory, MME\textsc{mory:DELete, MME\textsc{mory:MSIS}}}

\textbf{Syntax} \textsc{MME\textsc{mory:COPY <file\_source>,<file\_destination>}}

\begin{center}
\begin{tikzpicture}
\node (memory) {\textsc{memory}};
\node (copy) at (memory -| copy) {\textsc{COPY}};
\node (space) at (memory -| space) {\text{space}};
\node (source) at (space -| source) {\text{<file\_source>}};
\node (destination) at (source -| destination) {\text{<file\_destination>}};
\draw (memory) -- (copy);
\draw (space) -- (source);
\end{tikzpicture}
\end{center}
Arguments: `<file_source>` and `<file_destination>` ::= { `<file_name>` | `<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
- **FLOPy** 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","FLOPy"` copies the file `FILE1.WFM` on the waveform generator hard disk to the file `FILE2.WFM` on the floppy disk.

**MMEMory:DATA (No Query Form)**

Loads block data into the file on the default mass storage device.

Group: Mass Memory

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

Syntax: `MMEMory:DATA <file_name>,<data>`

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)**

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 (?)**

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>` specifies the data handle to be used.
**Arguments**

<data_handle>::="HCOPy" is the data handle for the waveform generator. At *RST, this parameter is set to "HCOP".

**Examples**

MMEMory:FEED "HCOPy"
sets the data handle.

**MMEMory:INITialize (No Query Form)**

Initializes the specified mass storage media. In this application, you can initialize the internal hard disk or floppy disk.

*NOTE. The initializing process erases all information already on the disk.*

**Group**  
Mass Memory

**Related Commands**  
MMEMory:MSIS

**Syntax**

MMEMory:INITialize[ <msus> [,DOS[,<NRF>]] ]

**Arguments**

<msus> (mass storage unit specifier)::={ "MAIN" | "FLOPy" }
where MAIN means the internal hard disk, and FLOPy means the floppy disk.
The media is initialized in DOS format.

<NRF> 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 the communication parameters (see Appendix E: Factory Initialization Settings).
Examples

MMEMory:INITialize "FLOPy"
initializes a floppy disk in DOS format.

MMEMory:MDIREctory (No Query Form)

Makes a directory on the specified mass storage.

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","FLOPy"
makes the directory “WAVEFORM” 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**

```
MMEemory:MSIS [( "MAIN" | "FLOPy" | "NET1" | "NET2" | "NET3" )]
```

MMEemory:MSIS?

**Arguments**

- **MAIN** selects the internal hard disk drive. This is the default value.
- **FLOPy** selects the internal floppy disk drive.
- **NET1**, **NET2**, or **NET3** selects the network drive 1, 2, or 3. (The network drive is specified by the SYSTEM:COMMunicate:LAN command.)

At *RST, this parameter is set to MAIN.

**Examples**

```
MMEemory:MSIS "FLOPy"
```

selects the floppy disk drive as the default mass storage device.

---

**MMEemory:MOVE (No Query Form)**

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

**Group**

Mass Memory

**Related Commands**

```
MMEemory:CDRectory, MMEemory:COPY, MMEemory:DELeTe, MMEemory:MSIS
```

**Syntax**

```
MMEemory: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.WMF","MAIN","FILE2.WFM","FLOPPy".
moves the file FILE1.WMF on the waveform generator hard disk to FILE2.WFM on the floppy disk.

MMEMory:NAME (?)

Sets the name of the file specification used by MMEMory:OPEN or CLOSE commands.

Group
Mass Memory

Related Commands
MMEMory:OPEN, MMEMory:CLOSE

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)

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.

Examples  SOURce:FUNCTion:USER "SAMPLE1.WFM";*OPC completes the SOURce:FUNCTion:USER "SAMPLE1.WFM" command before proceeding to the next command.

*OPT? (Query Only)

Returns the implemented options of the waveform generator.

Group  System

Syntax  *OPT?

Arguments  None

Returns  <string>

where:

0  the waveform generator has no options installed.
DD0  the waveform generator has Option 03 (Digital Data Out) installed.

Examples  *OPT?
might return DD0 indicating that the option 03 (Digital Data Out) installed in the instrument.
**OUTPut[1|2]:FILTer[:LPASs]:FREQuency (?)**

Determines the cutoff frequency of the low pass filter for the specified channel.

**Group**  
Output

**Syntax**  
OUTPut[1|2]:FILTer[:LPASs]:FREQuency { <NRf> | INFinity }  
OUTPut[1|2]:FILTer[:LPASs]:FREQuency?

**Arguments**  
<NRf> is the cutoff frequency of the low pass filter, in Hz. You can select 10e6 (10MHz), 20e6 (20MHz), 50e6 (50MHz), 100e6 (100MHz), or 9.9e37 (INFinity, that means “through”).

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

**Examples**  
OUTPut1:FILTer:LPASs:FREQuency 10e6  
sets the cutoff frequency of the low pass filter for CH 1 to 10 MHz.

**OUTPut[1|2|5|7][:STATe] (?)**

Controls whether the output terminal is open or closed. When the function is OFF, the terminal is at maximum isolation from the signal.

**Group**  
Output

**Related Commands**  
SOURce1:COMBine:FEED

**Syntax**  
OUTPut[1|2|5|7][:STATe] { ON | OFF | <NR1> }  
OUTPut[1|2|5|7][:STATe]?
Arguments

<ON> or <NR1> ≠ 0 turns the output on.

<OFF> or <NR1> = 0 turns the output off.

OUTPut7[:STATE] cannot be set to ON when SOURce1:COMBine:FEED is specified as SOURce7 (the noise generator).

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

Examples

OUTPut1:STATE 1
turns the CH 1 output on.

OUTPut[1]:ISTate (?)
AWG510 Only

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

Group

Output

Syntax

OUTPut[1]:ISTate { ON | OFF | <NR1> }

OUTPut[1]:ISTate?

Arguments

<ON> or <NR1> ≠ 0 enables the CH1 output.

<OFF> or <NR1> = 0 disables the CH1 output.

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

Example: OUTPUT1:ISTate 1
enables the CH1 output.

*PSC (?)

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 are
restored at power-on. For a complete discussion of the use of these registers,
refer to Section 3, Status and Event.

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)**

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 Initialization Settings.

**Group**  
System

**Related Commands**  
SYSTem:SECurity:IMMediate

**Syntax**  
*RST

**Arguments**  
None

**Examples**  
*RST

resets the instrument.
**[SOURce[1]:COMBine:FEED (?)**

Adds the signal from the noise generator or the external input to the CH 1 output.

**Group**
Source

**Related Commands**
OUTPut7:STATE

**Syntax**
[SOURce[1]:COMBine:FEED { "SOURce7" | "SOURce8" | "" }
[SOURce[1]:COMBine:FEED?

**Arguments**
SOURce7 adds the signal from the noise generator to the CH 1 output.
When you set the OUTPut7:STATE command to 1 (ON), you can not specify SOURce7 in this command.
SOURce8 adds the signal from the external input to the CH 1 output.
"" (null) adds no signal.
At *RST, the parameter is set to "" (null).

**Examples**
SOURce1:COMBine:FEED "SOURce7"
adds noise to the CH 1 output.

**[SOURce[1|2|5]:FREQuency[:CW|FIXed] (?)**

Sets the sampling frequency to output a waveform or pattern file. The file is specified by the SOURce<x>:FUNCTION:USER command. For the waveform generator, SOURce1 (CH 1), SOURce2 (CH 2), and SOURce5 (pattern) have the same frequency.

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

Related Commands  
[SOURce<x>]:FUNCTion:USER

Syntax  
[SOURce[1|2|5]:]FREQuency[:CW|FIXed]  <NRF>
[SOURce[1|2|5]:]FREQuency[:CW|FIXed]?

Arguments  
<NRF> is the sampling frequency. The range is 50 kHz to 1 GHz. At *RST, this value is set to 100 MHz.

Examples  
SOURce1:FREQuency 500MHz sets the sampling frequency to 500 MHz.

[SOURce[1|2|5]:]FUNCTION:USER (?)  
Specifies a waveform or pattern file that you have created as the output source. This command causes the file to be loaded into the waveform generator’s RAM prior to output.

Group  
Source

Related Commands  
[SOURce<x>]:FREQuency[:CW|FIXed]

Syntax  
SOURce[1|2|5]:FUNCTION:USER <file_name>[,<msus>]
SOURce[1|2|5]:FUNCTION:USER?
Command Descriptions

[SOURce[1|2|5]:]MARKer[1|2]:DElay (?)

Sets the marker output delay referenced to the continuous clock output for the specified channel.

**Group**  
Source

**Syntax**  
[SOURce[1|2|5]:]MARKer[1|2]:DElay <Nrf>

[SOURce[1|2|5]:]MARKer[1|2]:DElay?

**Arguments**  
<Nrf> is the delay time in seconds. The range is -1.0 ns to +1.0 ns with a resolution of 20 ps.

At *RST, this value is set to 0.
Examples SOURce1:MARKer1:DElay 500ps
sets the delay of marker1 to 500 ps for CH 1 output.

[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:HIGH (?)
Sets the high level for the marker output.

Group Source

Related Commands
[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW

Syntax
[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:HIGH <NRF>
[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:HIGH?

Arguments <NRF> is the high level voltage of the marker output. The range is -2.0 V to 2.0 V (into 50 Ω) with a resolution of 0.05 V. Note that the high level must be larger than the low level.

At *RST, this value is set to 2 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|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW (?)
Sets the low level for the marker output.

Group Source

Related Commands
[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:HIGH
Command Descriptions

**Syntax**

[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW

[SOURce[1|2|5]:]MARKer[1|2]:VOLTage[:LEVEL][:IMMediate]:LOW?

**Arguments**

<NRF> is the low level voltage of the marker output. The range is -2.0 V to 2.0 V (into 50 Ω) with a resolution of 0.05 V. Note that the low level must be smaller than the high level. At *RST, this value is set to 0.

**Examples**

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

**SOURce7:POWer[:LEVel][:IMMediate][:AMPLitude] (?)**

Sets the level of the noise generator output, in dBm/Hz.

**Group**

Source

**Syntax**

SOURce7:POWer[:LEVel][:IMMediate][:AMPLitude] <NRF>

SOURce7:POWer[:LEVel][:IMMediate][:AMPLitude]?

**Arguments**

<NRF> is the noise output level.
The range is -145 to -105 dBm/Hz in 1 dB step.
At *RST, this value is set to -105 dBm/Hz.
Examples SOURce7:POWer:LEVe1:IMMediate:AMPLitude -120
sets the level of the noise generator output to -120 dBm/Hz.

[SOURce[1|2|5]:]ROScillator:SOURce (?)

Selects the reference oscillator. For the waveform generator, SOURce1 (CH 1), SOURce2 (CH 2), and SOURce5 (pattern) have the same oscillator.

Group Source

Syntax [SOURce[1|2|5]:]ROScillator:SOURce { INTernal | EXTernal }
[SOURce[1|2|5]:]ROScillator:SOURce?

Arguments INTernal means that the reference frequency is derived from the internal precision oscillator.

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

At *RST, this parameter is set to INTernal.

Examples SOURce1:ROScillator:SOURce EXTernal selects the external clock source.

[SOURce[1|2]:]VOLTage[:LEVe1][:IMMediate][:AMPLitude] (?)

Sets the actual magnitude of the output signal from SOURce1 (CH 1), or SOURce2 (CH 2).

Group Source

Related Commands [SOURce[1|2]:]VOLTage[:LEVe1][:IMMediate]:OFFSet
**Syntax**

[SOURce[1|2]:]VOLTage[:LEVEL][:IMMEDIATE][:AMPLitude] <NRf>

[SOURce[1|2]:]VOLTage[:LEVEL][:IMMEDIATE][:AMPLitude]? 

**Arguments**

<NRf> is the amplitude:
SOURce1, 2  
The range is 20 mV to 2.0 V (into 50 Ω), in 1 mV steps.

At *RST, this value is set to 1 V for SOURce1 and SOURce2.

**Examples**

SOURce1:VOLTage:LEVEL:IMMEDIATE:AMPLitude 230mV

sets the amplitude of CH 1 waveform to 230 mV.

**SOURce5:VOLTage[:LEVEL][:IMMEDIATE]:HIGH (?)**

Sets the high level of the signal from the pattern generator (SOURce5, Option 03). This command is used in conjunction with LOW.

**Group**

Source

**Related Commands**

SOURce5:VOLTage[:LEVEL][:IMMEDIATE]:LOW

**Syntax**

SOURce5:VOLTage[:LEVEL][:IMMEDIATE]:HIGH <NRf>

SOURce5:VOLTage[:LEVEL][:IMMEDIATE]:HIGH?
Arguments  

<NRF> is the high level of the pattern generator output. The range is –2.0 V to +2.0 V in 1 mV steps.

At *RST, this value is set to 2 V.

Examples  

SOURce5:VOLTage:LEVe1:IMMEDIATE:HIGH 230mV
sets the high level of the pattern generator output to 230 mV.

SOURce5:VOLTage[:LEVe1][:IMMEDIATE]:LOW (?)

Sets the low level of the signal from the pattern generator (SOURce5, Option 03). This command is used in conjunction with HIGH.

Group  
Source

Related Commands  
SOURce5:VOLTage[:LEVe1][:IMMEDIATE]:HIGH

Syntax  
SOURce5:VOLTage[:LEVe1][:IMMEDIATE]:LOW <NRF>
SOURce5:VOLTage[:LEVe1][:IMMEDIATE]:LOW?

Arguments  

<NRF> is the low level of the pattern generator output in volts. The range is –2.0 V to +2.0 V in 1 mV steps.

At *RST, this value is set to 0.

Examples  

SOURce5:VOLTage:LEVe1:IMMEDIATE:LOW –230mV
sets the low level of the pattern generator output to –230 mV.
**[SOURce[1|2]:]VOLTage[:LEVEL][:IMMediate]:OFFSet (?)**

Sets the non-time-varying component of the signal that is added to SOURce1 (CH 1) or SOURce2 (CH 2).

**Group**  
Source

**Related Commands**  
[SOURce[1|2|7]:]VOLTage[:LEVEL][:IMMediate][:AMPLitude]

**Syntax**  
[SOURce[1|2]:]VOLTage[:LEVEL][:IMMediate]:OFFSet <NRf>  
[SOURce[1|2]:]VOLTage[:LEVEL][:IMMediate]:OFFSet?

**Arguments**  
<Nrf> is the offset voltage. The range is -1.000 V to +1.000 V, in 1 mV steps.

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

**SRE (?)**

Sets and queries the bits in the Service Request Enable Register (SRER). For a complete discussion of the use of these registers, refer to Section 3 Status and Events.

**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 00110000 binary.

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

STATus:OPERation:CONDition? (Query Only)

Returns the contents of the Operation Condition Register (OCR). For more information on registers, refer to Section 3, Status and Events.

Group : Status

Related Commands STATus:OPERation:ENABLE, STATus:OPERation[:EVENT]?

Syntax STATus:OPERation:CONDition?

Arguments None

Returns <NR1> indicates 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 (?)

Sets the enable mask for the Operation Enable Register (OENR). For more information on registers, refer to Section 3, Status and Events.

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 32767.

Returns    <NR1> indicates 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)

Returns the contents of the Operation Event Register (OEVR) and clears it. For more information on registers, refer to Section 3, Status and Events.

Group    Status

Related Commands   STATus:OPERation:CONDition?, STATus:OPERation:ENABLE
Syntax

$\text{STATus:OPERation[:EVENT]}?, \text{STATus:OPERation:ENABLE}$

Returns

$\langle\text{NR1}\rangle$ indicates the content of the OEVR in a decimal number.

Examples

$\text{STATus:OPERation:EVENT?}$
might return 1 which indicates that the OEVR contains the binary number 00000000 00000001 and the CAL bit is set.

$\text{STATus:PRESet (No Query Form)}$

Presets the SCPI enable registers OENR and QENR. For more information on registers, refer to Section 3, $\textit{Status and Events}$.

Group

Status

Syntax

$\text{STATus:PRESet}$

Arguments

None

Examples

$\text{STATus:PRESet}$

presets the SCPI enable registers.

$\text{STATus:QUEStionable:CONDition? (Query Only)}$

Returns the contents of the Questionable Condition Register (QCR). For more information on registers, refer to Section 3, $\textit{Status and Events}$.

Group

Status

Related Commands

$\text{STATus:QUEStionable:ENABLE}, \text{STATus:QUEStionable[:EVENT]}?$
**Syntax**

```
STATUS:QUESTIONable:CONDition?
```

**Returns**

<NR1> indicates 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 (?)**

Sets the enable mask for the Questionable Enable Register (QENR). For more information on registers, refer to Section 3, *Status and Events*.

**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 32767.

**Returns**

<NR1> indicates 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)

Returns the contents of the Questionable Event Register (QEV) and clears it. For more information on registers, refer to Section 3, Status and Events.

**Group**  
Status

**Related Commands**  
STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABLE

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

\[
\text{STATus} \rightarrow \cdot \rightarrow \text{QUEStionable} \rightarrow \cdot \rightarrow \text{EVENt} \rightarrow ?
\]

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

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

STATus:QUEue[:NEXT]? (Query Only)

Returns the next item from the error/event queue and removes that item from the queue. Operation is identical to that of the SYSTem:ERRor? query. Refer to Section 3, Status and Events for more details.

**Group**  
Status

**Related Commands**  
SYSTem:ERRor?

**Syntax**  
STATus:QUEue[:NEXT]?

\[
\text{STATus} \rightarrow \cdot \rightarrow \text{EVENt} \rightarrow \cdot \rightarrow \text{NEXT} \rightarrow ?
\]
Command Descriptions

**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**

```
STATus:QUEue:NEXT?
```

might return the following response:

```
-102,"Syntax error; possible invalid suffix - :SOUR:FREQ 2V"
```

In this case, the unit is invalid.

**STB? (Query Only)**

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 Section 3, Status and Events.

**Group**

Status

**Related Commands**

*CLS, *ESE, *ESR?, *SRE

**Syntax**

```
*STB?
```

**Arguments**

None

**Returns**

- `<NR1>` indicates 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)**

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 ignored:

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

**Examples** SYSTem:BEEPer:IMMediate turn on the beep sound.

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

Turns on or off the FTP (File Transfer Protocol) server function.

**Group** System

**Syntax** SYSTem:COMMunicate:LAN:FTP[:SERVer][:STATe] { ON | OFF |<NR1>}
SYSTem:COMMunicate:LAN[:SERVer][:STATe]?
Command Descriptions

Arguments
OFF or \(<NR1> = 0\) turns off the FTP server function.
ON or \(<NR1> \neq 0\) 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:GATeway[1|2|3]:ADDRes (?)
Sets the IP address of the gateway when you communicate with the waveform
generator from anywhere other than the local network segment.

Group
System

Syntax
SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRes
<net_address>,<ip_address>
SYSTem:COMMunicate:LAN:GATeway[1|2|3]:ADDRes?

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:GATEway: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:PING? (Query Only)

Executes the ping test, sending the ICMP ECHO_REQUEST packet to the 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 1, indicating that there was a response from the host 2.199.55.1.

SYSTEM:COMMunicate:LAN:RDEVice[1|2|3]:ADDRESS (?)

Sets the IP address of the remote host. The host corresponds to "NET<x>" in the menu display. (You can change this name by 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
SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:ADDRes<ip_address>

Arguments <ip_address>::=<string> is the IP address of the remote host.
*RST has no effect on the value.

Examples SYSTem:COMMunicate:LAN:RDEVic1:ADDResce "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]:FSYSte<directory_name>

Sets the mount directory on the specified remote host.

Group System

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

Syntax SYSTem:COMMunicate:LAN:RDEVice[1|2|3]:FSYSte<directory_name>

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

Examples SYSTem:COMMunicate:LAN:RDEVic1:FSYSte"/AWG/SAMPLE"
sets the mount directory to /AWG/SAMPLE on the remote host 1 (NET1).
SYStem:COMMunicate:LAN:RDEVice[1|2|3]:NAME (?)

Sets the name of the 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 (?)

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 for compatibility only.

**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] (?)

Turns on or off the LAN communication with the remote host.

Group  System

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

Syntax  
SYSTem:COMMunicate:LAN:RDEVice[1|2|3][::STATE] { ON | OFF | <NR1> }
SYSTem:COMMunicate:LAN:RDEVice[1|2|3][::STATE]?

Arguments  OFF or <NR1> = 0 turns off the LAN communication with the remote host.
ON or <NR1> ≠ 0 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]:ADDRes ( ? )

Sets the IP address of the waveform generator.

**NOTE.** You must set the IP address of the waveform generator to use its LAN functions. If you specify "" (null) for the IP address, the LAN functions will not work.

**Group**  
System

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

**Syntax**  
SYSTem:COMMunicate:LAN[:SELF]:ADDRes <ip_address>
SYSTem:COMMunicate:LAN[:SELF]:ADDRes?

**Arguments**  
<ip_address> ::= <string> is the IP address of the waveform generator.
*RST has no effect on the value.

**Examples**  
SYSTem:COMMunicate:LAN:SELF:ADDRes "2.199.55.1"  
sets the IP address of the waveform generator.

SYSTem:COMMunicate:LAN[:SELF]:SMask ( ? )

Sets the subnet mask of the waveform generator.

**Group**  
System

**Related Commands**  
SYSTem:COMMunicate:LAN[:SELF]:ADDRes
**Syntax**

SYSTem:COMMunicate:LAN[:SELF]:SMASK <ip_mask>
SYSTem:COMMunicate:LAN[:SELF]:SMASK?

**Arguments**

<ip_mask>::=<string> is the subnet mask of the 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 waveform generator.

**SYSTem:DATE (?)**

Sets the date for the waveform generator operating system.

**Group** System

**Syntax**

SYSTem:DATE <year>,<month>,<day>
SYSTem:DATE?

**Arguments**

<year>::=<NRf> must be entered as a four-digit number.

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

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

<NRf> is rounded to the nearest integer.

*RST has no effect on the value.
Examples

SYSTem:DATE 1998,10,31
sets the date.

SYSTem:ERRor? (Query Only)

Retrieves and returns error data from the Error and Event Queue. It has the same
function as the STATus:QUEue[:NEXT]? query. For more details, refer to
Section 3 Status and Event.

Group

System

Related Commands

SYSTem:QUEue[:NEXT]?

Syntax

SYSTem:ERRor?

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?
might return the following response:

-102,"Syntax error;possible invalid suffix - :SOUR:FREQ 2V"

This response indicates that the unit is invalid.
**SYS**Te*m:*KDIRrection (\?)

Determines the direction the cursor moves in response to the general purpose knob.

**Group**  
System

**Syntax**  
SYS*Te*m:*KDI*Re*ction  \{  FORWARD  |  BACKward  \}
SYS*Te*m:*KDI*Re*ction?

**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**  
SYS*Te*m:*KDI*Re*ction  BACKward  
makes the cursor move to the left when the general purpose knob turns clockwise.

**SYS**Te*m:*KEYBoard[:TYPE] (\?)

Selects the type of keyboard that connects to the waveform generator.

**Group**  
System

**Syntax**  
SYS*Te*m:*KE*Y*Board[:TYPE]  \{  ASCII  |  JIS  \}
SYS*Te*m:*KE*Y*Board[: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 (?)
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.

Group
System

Syntax
SYSTem:KLOCK { ON | OFF | <NR1> }  
SYSTem:KLOCK?

Arguments
- OFF or <NR1> = 0 unlocks controls of the front panel and keyboard.
- ON or <NR1> ≠ 0 locks controls of the front panel and keyboard.
- *RST has no effect on the parameter.
Returns

\(<\text{NR1}\) = 0 indicates the front panel and keyboard are unlocked.
\(<\text{NR1}\) = 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)

Immediately destroys all waveform generator data and settings. Current settings are initialized to their *RST values.

**NOTE.** This command erases all information already 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 (?)

Sets the internal clock.

Group

System
Related Commands

**SYSTem:DATE**

**Syntax**

SYSTem:TIME <hour>, <minute>, <second>
SYSTem:TIME?

**Arguments**

<hour>, <minute>, <second>
<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)**

Query elapsed time 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, which indicates 3 hours 18 minutes and 52 seconds have elapsed after turning on the waveform generator.

SYSTem:VERSion? (Query Only)

Returns the SCPI version number for which 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 1995.0.

*TRG (No Query Form)

Generates a trigger event. This command is equivalent to the TRIGger[:SEQUence][:IMMediate] command or pushing 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)
Generates a trigger event. This command is equivalent to the *TRG command or
pushing 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 (?)
Selects the impedance of the external trigger input.

Group       Trigger

Syntax       TRIGger[::SEQUence]:IMPedance <NRF>
TRIGger[::SEQUence]:IMPedance?

Arguments    <NRF> is 50 (50 Ω) or 1e3 (1 kΩ).
At *RST, the value is set to 50 Ω.
Examples
TRIGger:SEQUence:IMPedance 1k
selects 1 kΩ impedance for the external trigger input.

TRIGger[:SEQUence]:LEVel (?)

Sets the trigger level on the selected SOURce.

Group Trigger

Related Commands
TRIGger[:SEQUence]:SOURce

Syntax
TRIGger[:SEQUence]:LEVel <NRf>
TRIGger[:SEQUence]:LEVel?

Arguments
<NrF> 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 (?)

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 (?)

Determines whether the event occurs on 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 (?)

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 (?)

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 <NRf>

TRIGger[:SEQUence]:TIMer?

Arguments

<NR3> is the internal trigger rate. The range is 1.0 μs 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)

Performs the self test and returns the results. If an error is detected during self test, execution is immediately stopped.

NOTE. This command takes several minutes to run the self test, and the waveform generator will not respond to any commands and queries while it runs.

Group

Diagnostic

Related Commands

*CAL?, CALibration[:ALL], DIAGnostic[:IMMediate]

Syntax

*TST?

Arguments

None

Returns

<NR1>

0 Terminated without error.

−330 Self-test failed.
Examples  *TST?
          might return -330 indicating the self-test failed.

*WAI (No Query Form)

Prevents the waveform generator from executing further commands or queries until all pending operations finish.

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 puts the response message on the output queue. This response message cannot be retrieved unless you perform a retrieval operation through the external controller. For example, call IBREAD subroutine with the National Instruments drivers.

![Diagram of the process]

**Figure 2-6: 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 is lost.

The SBR (status byte register) MAV bit can be used to check the response message queuing state. Refer to Chapter 3, *Status and Events*, for more information on 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 discusses the data format along with the data transfer procedures.

Data File

The waveform generator handles these types of files:

- 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 Code Convert File contains the Code Convert Table.

The waveform generator creates these files automatically during front panel operation. However, when you operate the waveform generator remotely, you must create the files by editing or programming according to the specified format described in the following topics.
Waveform File

The Waveform File contains waveform data in single precision floating-point numbers and marker data.

```
MAGIC 1000<CR><LF>#<Num_digits><Num_bytes><Data(1)><Data(2)> ... <Data(n)><CLOCK<Clock><CR><LF>
```

Figure 2–7: The Waveform File format

**File Format.** The Waveform File consists of three main parts. See Figure 2–7.

```
<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>` means the number of digits in `<Num_bytes>`.
- `<Num_bytes>` means 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
00 00 00 00 00 00 00 00 00 43 4C 43 4F 43 4B 20
31 2B 30 30 30 30 30 30 30 30 30 30 65 2B 30 38
0D 0A
```

```
MAGIC 1000..#210

.........CLOCK

1.0000000000E+08

..
Pattern File

The Pattern File contains waveform data in the binary format.

```
MAGIC2000<CR><LF>\#<Num_digits><Num_bytes><Data(1)><Data(2)>...<Data(n)>[CLOCK<Clock><CR><LF>
```

Figure 2-8: The Pattern File format

**File Format.** The data consists of three main parts. See Figure 2–8:

```
<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>` means the number of digits in `<Num_bytes>`.
- `<Num_bytes>` means the byte count of the data that follows.
- `<Data(n)>` represents each data point in two bytes (16 bits). The low byte is transferred first.

Bit 0 (LSB) – Bit 9 corresponds to D0 – D9 on the rear panel (Digital Data Out, Option 03).

- Bits 13 and 14 are used for Markers 1 and 2, respectively.

```
<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.

![Sequence File format diagram](image)

**Figure 2-9: The Sequence File format**

**File Format.** The data consists of three main parts. See Figure 2-9.

<Sequence File>
:::=<Header><Sequence Definition>[[Optional Information>]]

where:

<Header>::=MAGIC<space>300x<CR><LF>

x=1 or 2 represents the number of channels for which sequences are defined in the file.

<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>]]<CR><LF>

<CHx_file_name>::=string is the waveform or pattern file name for the specified channel.

<Repeat_count>::=<NR1> is the repeat count for the line. 0 means infinity.

<Wait_trigger>::=<NR1> specifies whether or not to wait for a trigger.
<NR1>=0 means Off, ≠0 for On.

[Goto-1]:=<NR1> specifies whether or not to go to the next line.
<NR1>=0 means Off, ≠0 for On.

<Logic_jump_target>::=<NR1> is line number for the Logic-Jump. 0 means Off, -1 for Next, and -2 for Table-Jump. The default is Off.
<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(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(3)> , <Jump_on/off(4)> <CR> <LF>

 <Jump_on/off(n)> ::= <NR1> sets the Logic-Jump on or off.

 <NR1> = 0 means Off, >0 for On, and <0 for Ignore. The default is

 <Jump_mode> ::= JUMP_MODE <space> { LOGIC | TABLE } <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 to use the
 STROBE signal from the EVENT IN connector on the rear panel. <NR1> = 0
 means Off, ≠0 for On. The default is Off.

Example. This Sequence File contains two lines of sequence definitions for CH 1
 and CH 2.

MAGIC 3002
LINES 2
"SAMPLE1.wfm", "SAMPLE2.wfm", 1, 0, 0, 0
"SAMPLE3.wfm", "SAMPLE4.wfm", 1, 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
The Code Convert File is an ASCII text file that describes the Code Convert Table as displayed in the Edit menu.

\[ \text{Sequence of bit pattern definitions} \]

\[ \langle \text{Bit\_pattern}(1) \rangle \langle \text{Bit\_pattern}(2) \rangle \ldots \langle \text{Bit\_pattern}(n) \rangle \]

\[ \text{Transfer direction} \]

**Figure 2-10: The Code Convert File format**

**File Format.** The Code Convert File consists of bit pattern definitions. See Figure 2-10.

\[ \langle \text{Code\_Convert\_File} \rangle \::= \langle \text{Bit\_pattern}(1) \rangle \langle \text{Bit\_pattern}(2) \rangle \ldots \langle \text{Bit\_pattern}(n) \rangle \]

where:

\[ \langle \text{Bit\_pattern}(n) \rangle \::= \langle \text{Past\_Source} \rangle , \langle \text{Current\_Source} \rangle , \langle \text{Next\_Source} \rangle , \langle \text{Past\_Output} \rangle , \langle \text{Output\_Code} \rangle \langle \text{CR} \rangle \langle \text{LF} \rangle \]

\[ \langle \text{Past\_Source} \rangle , \langle \text{Current\_Source} \rangle , \langle \text{Next\_Source} \rangle , \langle \text{Past\_Output} \rangle , \langle \text{Output\_Code} \rangle \]

and \langle \text{Output\_Code} \rangle 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 the AWG510 & AWG520 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. When the waveform generator is connected to the Ethernet LAN, you can also transfer files to and from up to three network drives.

External Device to Waveform Generator

There are two ways to transfer data from the external device to the waveform generator: through the GPIB interface and through the Ethernet interface.

**Through GPIB interface only.** Use the MME\text{Mory:DATA} command.

\[ \text{MME\text{Mory:DATA} }<\text{file\_name}>,<\text{data}> \]

This command loads \(<\text{data}>\) into the file \(<\text{file\_name}>\) on the internal hard disk, floppy disk, or the network drive. The default directory and mass memory device are specified by the \text{MME\text{Mory:CDIrectory}} and \text{MME\text{Mory:MSI}}S commands respectively. The \(<\text{data}>\) is in IEEE488.2 block format.

For example, the following command string will load 2048 bytes of data to the file AWG1.

\[ \text{MME\text{Mory:DATA} } "AWG1",\#42048<\text{data}(1)<\text{data}(2)>\ldots<\text{data}(2048)> \]
Data Transfer

Through Ethernet Interface. Use the MMeMory:COpy command.

MMeMory:COpy <file_source>,<file_destination>

The file <file_source> is on the network device (NET1, NET2, or NET3) specified by the SYStem:COMMunicate:LAN:RDEVice commands.

For example, the following command string will copy the file FILE-PC on the network drive (defined as NET1) to the file FILE-AWG on the waveform generator hard disk.

MMeMory:COpy "FILE-PC","NET1","FILE-AWG","MAIN"

Waveform Generator to External Device

You can transfer file data from the waveform generator to an external device through Ethernet interface.

Through Ethernet Interface. Use the MMeMory:COpy command.

MMeMory:COpy <file_source>,<file_destination>

The file <file_destination> is on the network device (NET1, NET2, or NET3) specified by the SYStem:COMMunicate:LAN:RDEVice command.

For example, the following command string will copy the file FILE-AWG on the waveform generator floppy disk to the file FILE-PC on the network drive (defined as NET1).

MMeMory:COpy "FILE-AWG","FLOPy","FILE-PC","NET1"
Status and Events
Status and Event Reporting

This section provides details on status information and events reported by the waveform generator.

Status Reporting Structure

The waveform generator status reporting function conforms to the IEEE-488.2 and SCPI standards. The status reporting function is used to check for instrument errors and to identify the types of events that have occurred on the instrument.

Figure 3–1 shows an outline of the instrument’s status reporting function. The status reporting function is divided 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 on power on/off, command error, and command execution status.

This block is made up of two registers: the Standard Event Status Register (SESR) and the Event Status Enable Register (ESER). See the Standard/Event Status Block shown at the bottom of Figure 3–1.

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. This register cannot be written to by the user. The ESER is an eight-bit enable register that masks the SESR. This mask can be set by the user and can 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.

This block is made up of three registers: the Operation Condition Register (OCR), the Operation Event Register (OEV) and the Operation Enable Register (OEN). See the Operation Status Block shown in the middle of Figure 3–1.

When the instrument achieves a certain status, the corresponding bit is set to the OCR. This register cannot be written to by the user. OCR bits that have changed from false (reset) to true (set) status are set in the OEV. The function of the OEN is to mask the OEV. This mask can be set by the user and can take AND with the OEV 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 (OEV) on page 3–7, and Operation Enable Register (OEN) 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 (QEV) on page 3–8, and Questionable Enable Register (QEN) 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. This type of register can be set by the user.

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 (OEV)
- 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.

![Status Byte Register (SBR)](image)

Figure 3–2: The Status Byte Register (SBR)

Table 3–1: SBR bit functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Operation Summary Status (OSS).</td>
</tr>
<tr>
<td>6</td>
<td>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. 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.</td>
</tr>
<tr>
<td>5</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved.</td>
</tr>
<tr>
<td>3</td>
<td>Questionable Summary Status (QSS).</td>
</tr>
<tr>
<td>2</td>
<td>Event Queue Available (EAV).</td>
</tr>
<tr>
<td>1–0</td>
<td>Not used</td>
</tr>
</tbody>
</table>
Standard Event Status Register (SESR)

The SESR is made up of 8 bits. Each bit records the occurrence of a different type of event, as shown in Figure 3–3 and Table 3–2. The contents of this register are returned when the *ESR? query is used.

![SESR bit locations](image)

**Figure 3–3: The Standard Event Status Register (SESR)**

### Table 3–2: SESR bit functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Power On (PON). Indicates that the power to the instrument is on.</td>
</tr>
<tr>
<td>6</td>
<td>Not used.</td>
</tr>
<tr>
<td>5</td>
<td>Command Error (CME). Indicates that a command error has occurred while parsing by the command parser was in progress.</td>
</tr>
</tbody>
</table>
| 4 | Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons:  
  - A value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument  
  - 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:  
  - An attempt was made to retrieve messages from the output queue, despite the fact that the output queue is empty or in pending status.  
  - 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 note the occurrence of three different types of events as indicated in Figure 3-4 and Table 3-3.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TRIG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CAL</td>
</tr>
</tbody>
</table>

Figure 3-4: The Operation Condition Register (OCR)

Table 3-3: OCR bit functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 - 6</td>
<td>Not used.</td>
</tr>
<tr>
<td>5</td>
<td>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.</td>
</tr>
<tr>
<td>4 - 1</td>
<td>Not used.</td>
</tr>
<tr>
<td>0</td>
<td>Calibration (CAL): Indicates whether the instrument is being calibrated. This bit is set when calibration is in progress and is reset when calibration ends.</td>
</tr>
</tbody>
</table>

Operation Event Register (OEVR)

In this instrument, this register has the same content as the Operation Condition Register 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.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>FREQ</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Figure 3-5: The Questionable Condition Register (QCR)

Table 3-4: QCR bit functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 - 6</td>
<td>Not used. Must be set to zero for the waveform generator operation.</td>
</tr>
<tr>
<td>5</td>
<td>Frequency (FREQ). Indicates whether frequency accuracy of the signal is of questionable quality.</td>
</tr>
<tr>
<td>4 - 0</td>
<td>Not used. Must be set to zero for the waveform generator operation.</td>
</tr>
</tbody>
</table>
**Questionable Event Register (QEVR)**

In this instrument, this register has the same content as the Questionable Condition Register described above.

## 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 these 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 (see Figure 3–6). You can use this register to designate whether the SBR ESB bit should be set when an event has occurred and to determine whether the corresponding SESR bit has been 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.

![Figure 3–6: The Event Status Enable Register (ESER)](image)

### 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 determine 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 normally be set to 0.

![Figure 3-7: The Service Request Enable Register (SRER)]

The OENR is made up of bits that are defined exactly the same as bits 0 through 15 in the OEV (see Figure 3-8). This register is used for the operator to define whether the OSB bit in the SBR is set when an event occurs and the corresponding OEV 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)]

The QENR is made up of bits that are defined exactly the same as bits 0 through 15 in the QEV (see Figure 3-9). You can use this register to define whether the QSB bit in the SBR is set when an event occurs and the corresponding QEV 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 used in the waveform generator: output queues and error/event queues.

Output Queue

The output queue is a FIFO (first-in, first-out) queue and holds response messages to queries, where they await 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 and stores events as they occur in the instrument. If more than 64 events occur, the 64th event will be replaced with event code -350 (“Queue Overflow”). The oldest error code and text are retrieved using one of the following queries:

- SYSTem:ERRor?
- STATus:QUEue[:NEXT]?

First, issue the *ESR? query to read the contents of the SESR. The contents of the SESR are cleared after the 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? or STATus:QUEue[: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

See Figure 3–10. When an event occurs, a signal is sent to the OEV (1). If the corresponding bit in the OENR is also enabled (2), then the OSS bit in the SBR is set to one (3). See Figure 3–12.

Figure 3–10: Status and Event processing sequence — Operation status block

Questionable Status Block

See Figure 3–11. When an event occurs, a signal is sent to the QEV (1). If the corresponding bit in the QENR is also enabled (2), then the QSS bit in the SBR is set to one (3). See Figure 3–12.

Figure 3–11: Status and Event processing sequence — Questionable status block
See 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), then 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).

\[ \begin{align*}
1 & \quad \left( \begin{array}{c}
\text{Standard Event Status Register (SESR)} \\
\text{Read using *ESR?} \\
\text{Cannot be written}
\end{array} \right) \\
2 & \quad \left( \begin{array}{c}
\text{Event Status Enable Register (ESER)} \\
\text{Read using *ESE?} \\
\text{Write using *ESE}
\end{array} \right) \\
3 & \quad \left( \begin{array}{c}
\text{Status Byte Register (SBR)} \\
\text{Read using *STB?} \\
\text{Cannot be written}
\end{array} \right) \\
4 & \quad \left( \begin{array}{c}
\text{Service Request Enable Register (SRER)} \\
\text{Read using *SRE?} \\
\text{Write using *SRE}
\end{array} \right)
\end{align*} \]

\[ \begin{align*}
\text{From the Operation status block} & \quad \left( \begin{array}{c}
\text{From the Questionable status block}
\end{array} \right)
\end{align*} \]

\[ \begin{align*}
\text{Event queue} & \quad \left( \begin{array}{c}
\text{Output queue}
\end{array} \right)
\end{align*} \]

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 screen. Use the following procedure to display the status and event screen.

1. Push the **UTILITY** menu button on the front panel. The **UTILITY** menu appears on the screen.

2. Push the **Status** bottom menu button to display the Status submenu.

3. Push 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 part of the display.

![Status and Event Screen Diagram](image)

Figure 3–13: Status and Event screen
Synchronizing Execution

All commands used in this 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 3–12 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?` and `STATus:QUEue[:NEXT]?`. These are returned in the following format:

 `<event code>,"<event message>"`
Messages and Codes

Error and event codes with a negative value are SCPI standard codes; errors and events with a positive value are unique to the waveform generator.

Table 3–5 lists the definition of event codes. When an error has occurred, it is possible to find out what class of error has occurred by checking the code range. See Tables 3–6 through 3–12 for more information on events used by the waveform generator; events are organized by class in these tables.

### Table 3–5: Definition of event codes

<table>
<thead>
<tr>
<th>Event class</th>
<th>Code ranges</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No error</td>
<td>0</td>
<td>No event nor status</td>
</tr>
<tr>
<td>Command errors</td>
<td>−100 to −199</td>
<td>Command syntax errors</td>
</tr>
<tr>
<td>Execution errors</td>
<td>−200 to −299</td>
<td>Command execution errors</td>
</tr>
<tr>
<td>Device-specific errors</td>
<td>−300 to −399</td>
<td>Internal device errors</td>
</tr>
<tr>
<td>Query errors</td>
<td>−400 to −499</td>
<td>System event and query errors</td>
</tr>
<tr>
<td>Power-on events</td>
<td>−500 to −599</td>
<td>Power-on events</td>
</tr>
<tr>
<td>Operation complete events</td>
<td>−800 to −899</td>
<td>Operation complete events</td>
</tr>
<tr>
<td>Extended device-specific errors</td>
<td>1 to 32767</td>
<td>Device dependent device errors</td>
</tr>
<tr>
<td>Reserved</td>
<td>other than above</td>
<td>not used</td>
</tr>
</tbody>
</table>
Command Errors

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-100</td>
<td>Command error</td>
</tr>
<tr>
<td>-101</td>
<td>Invalid character</td>
</tr>
<tr>
<td>-102</td>
<td>Syntax error</td>
</tr>
<tr>
<td>-103</td>
<td>Invalid separator</td>
</tr>
<tr>
<td>-104</td>
<td>Data type error</td>
</tr>
<tr>
<td>-105</td>
<td>GET not allowed</td>
</tr>
<tr>
<td>-108</td>
<td>Parameter not allowed</td>
</tr>
<tr>
<td>-109</td>
<td>Missing parameter</td>
</tr>
<tr>
<td>-110</td>
<td>Command header error</td>
</tr>
<tr>
<td>-111</td>
<td>Header separator error</td>
</tr>
<tr>
<td>-112</td>
<td>Program mnemonic too long</td>
</tr>
<tr>
<td>-113</td>
<td>Undefined header</td>
</tr>
<tr>
<td>-114</td>
<td>Header suffix out of range</td>
</tr>
<tr>
<td>-120</td>
<td>Numeric data error</td>
</tr>
<tr>
<td>-121</td>
<td>Invalid character in number</td>
</tr>
<tr>
<td>-123</td>
<td>Exponent too large</td>
</tr>
<tr>
<td>-124</td>
<td>Too many digits</td>
</tr>
<tr>
<td>-128</td>
<td>Numeric data not allowed</td>
</tr>
<tr>
<td>Error code</td>
<td>Error message</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>-130</td>
<td>Suffix error</td>
</tr>
<tr>
<td>-131</td>
<td>Invalid suffix</td>
</tr>
<tr>
<td>-134</td>
<td>Suffix too long</td>
</tr>
<tr>
<td>-138</td>
<td>Suffix not allowed</td>
</tr>
<tr>
<td>-140</td>
<td>Character data error</td>
</tr>
<tr>
<td>-141</td>
<td>Invalid character data</td>
</tr>
<tr>
<td>-144</td>
<td>Character data too long</td>
</tr>
<tr>
<td>-148</td>
<td>Character data not allowed</td>
</tr>
<tr>
<td>-150</td>
<td>String data error</td>
</tr>
<tr>
<td>-151</td>
<td>invalid string data</td>
</tr>
<tr>
<td>-158</td>
<td>String data not allowed</td>
</tr>
<tr>
<td>-160</td>
<td>Block data error</td>
</tr>
<tr>
<td>-161</td>
<td>Invalid block data</td>
</tr>
<tr>
<td>-168</td>
<td>Block data not allowed</td>
</tr>
<tr>
<td>-170</td>
<td>Expression error</td>
</tr>
<tr>
<td>-171</td>
<td>Invalid expression</td>
</tr>
<tr>
<td>-178</td>
<td>Expression data not allowed</td>
</tr>
<tr>
<td>-180</td>
<td>Macro error</td>
</tr>
<tr>
<td>-181</td>
<td>Invalid outside macro definition</td>
</tr>
<tr>
<td>-183</td>
<td>Invalide inside macro definition</td>
</tr>
<tr>
<td>-184</td>
<td>Macro parameter error</td>
</tr>
</tbody>
</table>
Execution Errors

These error codes are returned when an error is detected during command execution.

Table 3-7: Execution errors

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-200</td>
<td>Execution error</td>
</tr>
<tr>
<td>-201</td>
<td>Invalid while in local</td>
</tr>
<tr>
<td>-202</td>
<td>Settings lost due to RTL</td>
</tr>
<tr>
<td>-203</td>
<td>Command protected</td>
</tr>
<tr>
<td>-210</td>
<td>Trigger error</td>
</tr>
<tr>
<td>-211</td>
<td>Trigger ignored</td>
</tr>
<tr>
<td>-212</td>
<td>Arm ignored</td>
</tr>
<tr>
<td>-213</td>
<td>Init Ignored</td>
</tr>
<tr>
<td>-214</td>
<td>Trigger deadlock</td>
</tr>
<tr>
<td>-215</td>
<td>Arm deadlock</td>
</tr>
<tr>
<td>-220</td>
<td>Parameter error</td>
</tr>
<tr>
<td>-221</td>
<td>Settings conflict</td>
</tr>
<tr>
<td>-222</td>
<td>Data out of range</td>
</tr>
<tr>
<td>-223</td>
<td>Too much data</td>
</tr>
<tr>
<td>-224</td>
<td>Illegal parameter value</td>
</tr>
<tr>
<td>-225</td>
<td>Out of memory</td>
</tr>
<tr>
<td>-226</td>
<td>Lists not same length</td>
</tr>
<tr>
<td>-230</td>
<td>Data corrupt or stale</td>
</tr>
<tr>
<td>-231</td>
<td>Data questionable</td>
</tr>
<tr>
<td>-232</td>
<td>Invalid format</td>
</tr>
<tr>
<td>-233</td>
<td>Invalid version</td>
</tr>
<tr>
<td>-240</td>
<td>Hardware error</td>
</tr>
<tr>
<td>-241</td>
<td>Hardware missing</td>
</tr>
<tr>
<td>-250</td>
<td>Mass storage error</td>
</tr>
<tr>
<td>-251</td>
<td>Missing mass storage</td>
</tr>
<tr>
<td>-252</td>
<td>Missing media</td>
</tr>
<tr>
<td>Error code</td>
<td>Error message</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>-253</td>
<td>Corrupt media</td>
</tr>
<tr>
<td>-254</td>
<td>Media full</td>
</tr>
<tr>
<td>-255</td>
<td>Directory full</td>
</tr>
<tr>
<td>-256</td>
<td>File name not found</td>
</tr>
<tr>
<td>-257</td>
<td>File name error</td>
</tr>
<tr>
<td>-258</td>
<td>Media protected</td>
</tr>
<tr>
<td>-260</td>
<td>Expression error</td>
</tr>
<tr>
<td>-261</td>
<td>Math error in expression</td>
</tr>
<tr>
<td>-270</td>
<td>Macro error</td>
</tr>
<tr>
<td>-271</td>
<td>Macro syntax error</td>
</tr>
<tr>
<td>-272</td>
<td>Macro execution error</td>
</tr>
<tr>
<td>-273</td>
<td>Illegal macro label</td>
</tr>
<tr>
<td>-274</td>
<td>Macro parameter error</td>
</tr>
<tr>
<td>-275</td>
<td>Macro definition too long</td>
</tr>
<tr>
<td>-276</td>
<td>Macro recursion error</td>
</tr>
<tr>
<td>-277</td>
<td>Macro redefinition not allowed</td>
</tr>
<tr>
<td>-278</td>
<td>Macro header not found</td>
</tr>
<tr>
<td>-280</td>
<td>Program error</td>
</tr>
<tr>
<td>-281</td>
<td>Cannot create program</td>
</tr>
<tr>
<td>-282</td>
<td>Illegal program name</td>
</tr>
<tr>
<td>-283</td>
<td>Illegal variable name</td>
</tr>
<tr>
<td>-284</td>
<td>Program currently running</td>
</tr>
<tr>
<td>-285</td>
<td>Program syntax error</td>
</tr>
<tr>
<td>-286</td>
<td>Program runtime error</td>
</tr>
<tr>
<td>-290</td>
<td>Memory use error</td>
</tr>
<tr>
<td>-291</td>
<td>Out of memory</td>
</tr>
<tr>
<td>-292</td>
<td>Referenced name does not exist</td>
</tr>
<tr>
<td>-293</td>
<td>Referenced name already exists</td>
</tr>
<tr>
<td>-294</td>
<td>Incompatible type</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-300</td>
<td>Device specific error</td>
</tr>
<tr>
<td>-310</td>
<td>System error</td>
</tr>
<tr>
<td>-311</td>
<td>Memory error</td>
</tr>
<tr>
<td>-312</td>
<td>PUD memory lost</td>
</tr>
<tr>
<td>-313</td>
<td>Calibration memory lost</td>
</tr>
<tr>
<td>-314</td>
<td>Save/recal memory lost</td>
</tr>
<tr>
<td>-315</td>
<td>Configuration memory lost</td>
</tr>
<tr>
<td>-320</td>
<td>Storage fault</td>
</tr>
<tr>
<td>-321</td>
<td>Out of memory</td>
</tr>
<tr>
<td>-330</td>
<td>Self-test failed</td>
</tr>
<tr>
<td>-340</td>
<td>Calibration failed</td>
</tr>
<tr>
<td>-350</td>
<td>Queue overflow</td>
</tr>
<tr>
<td>-360</td>
<td>Communication error</td>
</tr>
<tr>
<td>-361</td>
<td>Parity error in program message</td>
</tr>
<tr>
<td>-362</td>
<td>Framing error in program message</td>
</tr>
<tr>
<td>-363</td>
<td>Input buffer overrun</td>
</tr>
</tbody>
</table>
Query Errors

These error codes are returned in response to an unanswered query.

Table 3-9: Query errors

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>query error</td>
</tr>
<tr>
<td>-410</td>
<td>query INTERRUPTED</td>
</tr>
<tr>
<td>-420</td>
<td>query UNTERMINATED</td>
</tr>
<tr>
<td>-430</td>
<td>query DEADLOCKED</td>
</tr>
<tr>
<td>-440</td>
<td>query UNTERMINATED after indefinite response</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-500</td>
<td>Power on</td>
</tr>
</tbody>
</table>

Operation Complete Events

These events occur when the instrument’s synchronization protocol, having been enabled by an *OPC command, completes all selected pending operations.

Table 3-11: Operation complete events

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>-800</td>
<td>Operation complete</td>
</tr>
</tbody>
</table>
Device Errors

These error codes are unique to the waveform generator.

Table 3-12: Device errors

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1101</td>
<td>CH1 internal offset</td>
</tr>
<tr>
<td>1102</td>
<td>CH1 output offset</td>
</tr>
<tr>
<td>1103</td>
<td>CH1 gain</td>
</tr>
<tr>
<td>1111</td>
<td>CH1 x3dB attenuator</td>
</tr>
<tr>
<td>1112</td>
<td>CH1 x6dB attenuator</td>
</tr>
<tr>
<td>1113</td>
<td>CH1 x12dB attenuator</td>
</tr>
<tr>
<td>1114</td>
<td>CH1 x20dB attenuator</td>
</tr>
<tr>
<td>1121</td>
<td>CH1 10MHz filter</td>
</tr>
<tr>
<td>1122</td>
<td>CH1 20MHz filter</td>
</tr>
<tr>
<td>1123</td>
<td>CH1 50MHz filter</td>
</tr>
<tr>
<td>1124</td>
<td>CH1 100MHz filter</td>
</tr>
<tr>
<td>1201</td>
<td>CH2 internal offset</td>
</tr>
<tr>
<td>1202</td>
<td>CH2 output offset</td>
</tr>
<tr>
<td>1203</td>
<td>CH2 gain</td>
</tr>
<tr>
<td>1211</td>
<td>CH2 x3dB attenuator</td>
</tr>
<tr>
<td>1212</td>
<td>CH2 x6dB attenuator</td>
</tr>
<tr>
<td>1213</td>
<td>CH2 x12dB attenuator</td>
</tr>
<tr>
<td>1214</td>
<td>CH2 x20dB attenuator</td>
</tr>
<tr>
<td>1221</td>
<td>CH2 10MHz filter</td>
</tr>
<tr>
<td>1222</td>
<td>CH2 20MHz filter</td>
</tr>
<tr>
<td>1223</td>
<td>CH2 50MHz filter</td>
</tr>
<tr>
<td>1224</td>
<td>CH2 100MHz filter</td>
</tr>
<tr>
<td>2100</td>
<td>System failure</td>
</tr>
<tr>
<td>2101</td>
<td>Real-time clock power</td>
</tr>
<tr>
<td>2102</td>
<td>Configuration record and checksum status</td>
</tr>
<tr>
<td>2103</td>
<td>Incorrect configuration</td>
</tr>
<tr>
<td>2104</td>
<td>Memory size miscompare</td>
</tr>
<tr>
<td>2105</td>
<td>Fixed-disk drive initialization status</td>
</tr>
<tr>
<td>2106</td>
<td>Time status</td>
</tr>
<tr>
<td>Error code</td>
<td>Error message</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>2110</td>
<td>Front panel failure</td>
</tr>
<tr>
<td>2111</td>
<td>Front panel configuration</td>
</tr>
<tr>
<td>2112</td>
<td>Front panel communication</td>
</tr>
<tr>
<td>2113</td>
<td>Front panel RAM</td>
</tr>
<tr>
<td>2114</td>
<td>Front panel ROM</td>
</tr>
<tr>
<td>2115</td>
<td>Front panel A/D</td>
</tr>
<tr>
<td>2116</td>
<td>Front panel timer</td>
</tr>
<tr>
<td>2301</td>
<td>A30 board failure</td>
</tr>
<tr>
<td>2401</td>
<td>Clock delay data not found</td>
</tr>
<tr>
<td>2402</td>
<td>Clock delay data checksum</td>
</tr>
<tr>
<td>2700</td>
<td>Calibration data failure</td>
</tr>
<tr>
<td>2701</td>
<td>Calibration data not found</td>
</tr>
<tr>
<td>2702</td>
<td>Calibration data checksum</td>
</tr>
<tr>
<td>3100</td>
<td>Control1 register failure</td>
</tr>
<tr>
<td>3101</td>
<td>Control1 register bit0 to bit3</td>
</tr>
<tr>
<td>3200</td>
<td>Event table data bus failure</td>
</tr>
<tr>
<td>3201</td>
<td>Event table data bus bit0 to bit15</td>
</tr>
<tr>
<td>3250</td>
<td>Event table address bus failure</td>
</tr>
<tr>
<td>3251</td>
<td>Event table address bus bit0 to bit23</td>
</tr>
<tr>
<td>3300</td>
<td>Event table memory chip select failure</td>
</tr>
<tr>
<td>3301</td>
<td>Event table memory chip select 0</td>
</tr>
<tr>
<td>3302</td>
<td>Event table memory chip select 1</td>
</tr>
<tr>
<td>3350</td>
<td>Event table memory chip cell failure</td>
</tr>
<tr>
<td>3351</td>
<td>Event table memory chip 0</td>
</tr>
<tr>
<td>3352</td>
<td>Event table memory chip 1</td>
</tr>
<tr>
<td>4100</td>
<td>A40 board failure</td>
</tr>
<tr>
<td>4101</td>
<td>PLL 500MHz locked</td>
</tr>
<tr>
<td>4102</td>
<td>PLL 600MHz unlocked</td>
</tr>
<tr>
<td>4103</td>
<td>PLL 1350MHz unlocked</td>
</tr>
<tr>
<td>4104</td>
<td>PLL 1500MHz locked</td>
</tr>
<tr>
<td>5100</td>
<td>Sequence memory data bus failure</td>
</tr>
<tr>
<td>5101</td>
<td>Sequence memory data bus bit0 to bit15</td>
</tr>
</tbody>
</table>
### Table 3-12: Device errors (Cont.)

<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>5150</td>
<td>Sequence memory address bus failure</td>
</tr>
<tr>
<td>5151 to 5174</td>
<td>Sequence memory address bus bit0 to bit23</td>
</tr>
<tr>
<td>5200</td>
<td>Sequence memory chip select failure</td>
</tr>
<tr>
<td>5201 to 5206</td>
<td>Sequence memory chip select 0 to select 5</td>
</tr>
<tr>
<td>5250</td>
<td>Sequence memory chip cell failure</td>
</tr>
<tr>
<td>5251 to 5256</td>
<td>Sequence memory chip 0 to chip 5</td>
</tr>
<tr>
<td>5300</td>
<td>CH1 Waveform memory data bus failure</td>
</tr>
<tr>
<td>5301 to 5316</td>
<td>CH1 Waveform memory data bus bit0 to bit15</td>
</tr>
<tr>
<td>5350</td>
<td>CH1 Waveform memory address bus failure</td>
</tr>
<tr>
<td>5351 to 5374</td>
<td>CH1 Waveform memory address bus bit0 to bit23</td>
</tr>
<tr>
<td>5400</td>
<td>CH1 Waveform memory chip select failure</td>
</tr>
<tr>
<td>5401 to 5449</td>
<td>CH1 Waveform memory chip select 0 to select 48</td>
</tr>
<tr>
<td>5500</td>
<td>CH1 Waveform memory chip cell failure</td>
</tr>
<tr>
<td>5501 to 5549</td>
<td>CH1 Waveform memory chip 0 to chip 48</td>
</tr>
<tr>
<td>5600</td>
<td>CH2 or CH1 Waveform memory data bus failure</td>
</tr>
<tr>
<td>5601 to 5616</td>
<td>CH2 or CH1 Waveform memory data bus bit0 to bit15</td>
</tr>
<tr>
<td>5650</td>
<td>CH2 or CH1 Waveform memory address bus failure</td>
</tr>
<tr>
<td>5651 to 5674</td>
<td>CH2 or CH1 Waveform memory address bus bit0 to bit23</td>
</tr>
<tr>
<td>5700</td>
<td>CH2 or CH1 Waveform memory chip select failure</td>
</tr>
<tr>
<td>5701 to 5748</td>
<td>CH2 or CH1 Waveform memory chip select 0 to select 47</td>
</tr>
<tr>
<td>5800</td>
<td>CH2 or CH1 Waveform memory chip cell failure</td>
</tr>
<tr>
<td>5801 to 5848</td>
<td>CH2 or CH1 Waveform memory chip 0 to chip 47</td>
</tr>
<tr>
<td>5900</td>
<td>CH1 Arb D/A failure</td>
</tr>
<tr>
<td>5901 to 5912</td>
<td>CH1 Arb D/A data bit0 to bit11</td>
</tr>
<tr>
<td>5950</td>
<td>CH2 Arb D/A failure</td>
</tr>
<tr>
<td>5951 to 5962</td>
<td>CH2 Arb D/A data bit0 to bit11</td>
</tr>
<tr>
<td>7110</td>
<td>CH1 output offset failure</td>
</tr>
<tr>
<td>7111</td>
<td>CH1 output offset</td>
</tr>
<tr>
<td>7120</td>
<td>CH1 internal offset failure</td>
</tr>
<tr>
<td>7121</td>
<td>CH1 internal offset</td>
</tr>
<tr>
<td>7130</td>
<td>CH1 Arb gain failure</td>
</tr>
<tr>
<td>7131</td>
<td>CH1 Arb gain</td>
</tr>
</tbody>
</table>

AWG510 & AWG520 Arbitrary Waveform Generator Programmer Manual
<table>
<thead>
<tr>
<th>Error code</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>7140</td>
<td>CH1 attenuator failure</td>
</tr>
<tr>
<td>7141</td>
<td>CH1 3dB attenuator</td>
</tr>
<tr>
<td>7142</td>
<td>CH1 6dB attenuator</td>
</tr>
<tr>
<td>7143</td>
<td>CH1 12dB attenuator</td>
</tr>
<tr>
<td>7144</td>
<td>CH1 20dB attenuator</td>
</tr>
<tr>
<td>7150</td>
<td>CH1 filter failure</td>
</tr>
<tr>
<td>7151</td>
<td>CH1 10MHz filter</td>
</tr>
<tr>
<td>7152</td>
<td>CH1 20MHz filter</td>
</tr>
<tr>
<td>7153</td>
<td>CH1 50MHz filter</td>
</tr>
<tr>
<td>7150</td>
<td>CH1 100MHz filter</td>
</tr>
<tr>
<td>7170</td>
<td>CH1 output key failure</td>
</tr>
<tr>
<td>7171</td>
<td>CH1 output key</td>
</tr>
<tr>
<td>7210</td>
<td>CH2 or CH1 output offset failure</td>
</tr>
<tr>
<td>7211</td>
<td>CH2 or CH1 output offset</td>
</tr>
<tr>
<td>7220</td>
<td>CH2 or CH1 internal offset failure</td>
</tr>
<tr>
<td>7221</td>
<td>CH2 or CH1 internal offset</td>
</tr>
<tr>
<td>7230</td>
<td>CH2 or CH1 Arb gain failure</td>
</tr>
<tr>
<td>7231</td>
<td>CH2 or CH1 Arb gain</td>
</tr>
<tr>
<td>7240</td>
<td>CH2 or CH1 attenuator failure</td>
</tr>
<tr>
<td>7241</td>
<td>CH2 or CH1 3dB attenuator</td>
</tr>
<tr>
<td>7242</td>
<td>CH2 or CH1 6dB attenuator</td>
</tr>
<tr>
<td>7243</td>
<td>CH2 or CH1 12dB attenuator</td>
</tr>
<tr>
<td>7244</td>
<td>CH2 or CH1 20dB attenuator</td>
</tr>
<tr>
<td>7250</td>
<td>CH2 or CH1 filter failure</td>
</tr>
<tr>
<td>7251</td>
<td>CH2 or CH1 10MHz filter</td>
</tr>
<tr>
<td>7252</td>
<td>CH2 or CH1 20MHz filter</td>
</tr>
<tr>
<td>7253</td>
<td>CH2 or CH1 50MHz filter</td>
</tr>
<tr>
<td>7254</td>
<td>CH2 or CH1 100MHz filter</td>
</tr>
<tr>
<td>7270</td>
<td>CH2 or CH1 output key failure</td>
</tr>
<tr>
<td>7271</td>
<td>CH2 or CH1 output key</td>
</tr>
</tbody>
</table>
Programming Examples
Programming Examples

The example programs illustrate methods you can use to control the waveform generator from the GPIB interface. The floppy disk supplied with the waveform generator contains listing for these programs written in Microsoft Visual C++ and Visual BASIC. The programs will also work with National Instruments LabVIEW.

The diskette also contains the file README.TXT. Refer to the file for details about how to run the programs.

The programs run on a PC-compatible system equipped with a National Instruments GPIB board and associated drivers.

Figure 4-1: Equipment needed to run the example programs
Appendices
Appendix A: Character Charts

Table A-1: The AWG500 character set

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NUL</td>
<td>0</td>
<td>16</td>
<td>32</td>
<td>0</td>
<td>48</td>
<td>64</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>!</td>
<td>1</td>
<td>17</td>
<td>33</td>
<td>1</td>
<td>49</td>
<td>65</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>&quot;</td>
<td>2</td>
<td>18</td>
<td>34</td>
<td>2</td>
<td>50</td>
<td>66</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>#</td>
<td>3</td>
<td>19</td>
<td>35</td>
<td>3</td>
<td>51</td>
<td>67</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>4</td>
<td>20</td>
<td>36</td>
<td>4</td>
<td>52</td>
<td>68</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>5</td>
<td>21</td>
<td>37</td>
<td>5</td>
<td>53</td>
<td>69</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>&amp;</td>
<td>6</td>
<td>22</td>
<td>38</td>
<td>6</td>
<td>54</td>
<td>70</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>,</td>
<td>7</td>
<td>23</td>
<td>39</td>
<td>7</td>
<td>55</td>
<td>71</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>(</td>
<td>8</td>
<td>24</td>
<td>40</td>
<td>8</td>
<td>56</td>
<td>72</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>)</td>
<td>9</td>
<td>25</td>
<td>41</td>
<td>9</td>
<td>57</td>
<td>73</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>LF</td>
<td>10</td>
<td>26</td>
<td>42</td>
<td>10</td>
<td>58</td>
<td>74</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>ESC</td>
<td>11</td>
<td>27</td>
<td>43</td>
<td>11</td>
<td>59</td>
<td>75</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>28</td>
<td>44</td>
<td>12</td>
<td>60</td>
<td>76</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>CR</td>
<td>13</td>
<td>29</td>
<td>45</td>
<td>13</td>
<td>61</td>
<td>77</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>30</td>
<td>46</td>
<td>14</td>
<td>62</td>
<td>78</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>31</td>
<td>47</td>
<td>15</td>
<td>63</td>
<td>79</td>
<td>95</td>
</tr>
</tbody>
</table>

AWG510 & AWG520 Arbitrary Waveform Generator Programmer Manual
<table>
<thead>
<tr>
<th>B7 B6 B5</th>
</tr>
</thead>
<tbody>
<tr>
<td>B4 B3 B2 B1</td>
</tr>
<tr>
<td>CONTROL NUMBERS SYMBOLS UPPER CASE LOWER CASE</td>
</tr>
<tr>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
</tr>
<tr>
<td>0 0 1 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
</tr>
<tr>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
</tr>
<tr>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
</tr>
<tr>
<td>1 1 0 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

**KEY**
- **ocetal**: 5
- **hex**: 5
- **ENQ**: ASCII character
- **LGIB code (with ATN asserted)**
- **decimal**

---

**Appendix A: Character Charts**

**Table A-2: ASCII & GPIB code chart**

<table>
<thead>
<tr>
<th>ADDRESSABLE COMMANDS</th>
<th>UNIVERSAL COMMANDS</th>
<th>LISTEN ADDRESSES</th>
<th>TALK ADDRESSES</th>
<th>SECONDARY ADDRESSES OR COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>octal 5</td>
<td>PP 20</td>
<td>DC1 40</td>
<td>TA6 100</td>
<td>SA0 140</td>
</tr>
<tr>
<td>hex 5</td>
<td>ENQ 11</td>
<td>DC2 41</td>
<td>TA1 101</td>
<td>SA1 141</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Tektronix**

REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 644-2073

**AWG510 & AWG520 Arbitrary Waveform Generator Programmer Manual**
Appendix B: Reserved Words

The words in the following list are reserved words for use with the AWG500 Series Arbitrary Waveform Generator.

*CAL
*CLS
*ESE
*ESR
*IDN
*OPC
*OPT
*PSC
*RST
*SRE
*STB
*TRG
*TST
*WAI
ABORt
ADDRes
ALL
AMPLitude
AOuTput
ASCII
AWGControl
BACKward
BEEPer
BMP
BRIGHTness
CALibration
CATalog
CDIRectory
CH<<
CLOCK
CLOSE
COMMunicate
CONDition
CONTInuous
COPY
COUNT
CW
DATA
DATE
DELay
DELeTe
DESTInation
DEVice
DIAgnostic
DISPlay
DOUTput
ENABle
ENHanced
ERRor
EVENT
EXTERNAL
FEED
FILTER
FIXed
FLOPy
FORtward
FREQuency
FSYStem
FUNCTION
GATed
GATeway<<
HCOPY
HIGH
INTernal
IMMediate
IMPedance
INITialize
ISTS
JIS
KDIREction
KEYBoard
KLOCK
LAN
LANGuage
LEVEL
LOGic
LOW
LPASS
MAIN
MARKer
MDIREctory
MMEMory
MSIS
NAME
NEGative
NET<<
NFS
NORMAL
OFFSet
OPEN
OPERation
OUTPUT
PING
POLarity
POSITive
POWER
PRESet
PROtocol
QUEstionable
QUEu
RDEVice<<
RMODE
ROSCillator
RSTate
RUN
SDUMp
SECurity
SELect
SELF
SEQUence
SLOPe
SMASK
SMEMemory
SREStore
SOURce
SSAVe
STATe
STOP
SYSTem
TEXT
TIFF
TIME
TIMer
TRIGger
TYPE
UNLock
UPTime
USER
VERSion
VOLTage
WMEMory
Appendix C: GPIB Interface Specification

This appendix lists and describes the GPIB functions and messages that the AWG500 Series Arbitrary Waveform Generator implements.

Interface Functions

Table C−1 shows which GPIB interface functions are implemented in this instrument. Following the table is a brief description of each function.

<table>
<thead>
<tr>
<th>Interface Function</th>
<th>Implemented Subset</th>
<th>Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceptor Handshake (AH)</td>
<td>AH1</td>
<td>Complete</td>
</tr>
<tr>
<td>Source Handshake (SH)</td>
<td>SH1</td>
<td>Complete</td>
</tr>
<tr>
<td>Talker (T)</td>
<td>T6</td>
<td>Basic Talker, Serial Poll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unaddress if my-listen-address (MLA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No Talk Only mode</td>
</tr>
<tr>
<td>Listener (L)</td>
<td>L4</td>
<td>Basic Listener</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unaddress if my talk address (MTA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No Listen Only mode</td>
</tr>
<tr>
<td>Service Request (SR)</td>
<td>SR1</td>
<td>Complete</td>
</tr>
<tr>
<td>Remote/Local (RL)</td>
<td>RL1</td>
<td>Complete</td>
</tr>
<tr>
<td>Parallel Poll (PP)</td>
<td>PP0</td>
<td>None</td>
</tr>
<tr>
<td>Device Clear (DC)</td>
<td>DC1</td>
<td>Complete</td>
</tr>
<tr>
<td>Device Trigger (DT)</td>
<td>DT1</td>
<td>Complete</td>
</tr>
<tr>
<td>Controller (C)</td>
<td>C0</td>
<td>None</td>
</tr>
<tr>
<td>Electrical Interface</td>
<td>E2</td>
<td>Three-state driver</td>
</tr>
</tbody>
</table>
Appendix C: GPIB Interface Specification

- Acceptor Handshake (AH). Allows a listening device to help coordinate the proper reception of data. The AH function holds off initiation or termination of a data transfer until the listening device is ready to receive the next data byte.

- Source Handshake (SH). Allows a talking device to help coordinate the proper transfer of data. The SH function controls the initiation and termination of the data byte transfer.

- Talker (T). Allows a device to send device-dependent data over the interface. This capability exists only when the device is addressed to talk. The function uses a one-byte address.

- Listener (L). Allows a device to receive device-dependent data over the interface. This capability exists only when the device is addressed to listen. This function uses a one-byte address.

- Service Request (SR). Allows a device to request service from the controller.

- Remote/Local (RL). Allows a device to select between two sources for operating control. This function determines whether input information from the front panel controls (local) or GPIB commands (remote) control the waveform generator.

- Device Clear (DC). Allows a device to be cleared or initialized, either individually or as part of a group of devices.

- Controller (C). Allows a device with the capability to send the device address, universal commands, and addressed commands to other devices over the interface to do so.

- Electrical Interface (E). Identifies the type of electrical interface. The notation E1 indicates the electrical interface uses open collector drivers, while E2 indicates the electrical interface uses three-state drivers.
Interface Messages

Table C–2 shows the standard interface messages that are supported by the waveform generator.

<table>
<thead>
<tr>
<th>Message</th>
<th>GPIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCL</td>
<td>Yes</td>
</tr>
<tr>
<td>GET</td>
<td>Yes</td>
</tr>
<tr>
<td>GTL</td>
<td>Yes</td>
</tr>
<tr>
<td>LLO</td>
<td>Yes</td>
</tr>
<tr>
<td>PPC</td>
<td>No</td>
</tr>
<tr>
<td>PPD</td>
<td>No</td>
</tr>
<tr>
<td>PPE</td>
<td>No</td>
</tr>
<tr>
<td>PPU</td>
<td>No</td>
</tr>
<tr>
<td>SDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SPD</td>
<td>Yes</td>
</tr>
<tr>
<td>SPE</td>
<td>Yes</td>
</tr>
<tr>
<td>TCT</td>
<td>No</td>
</tr>
<tr>
<td>UNL</td>
<td>Yes</td>
</tr>
<tr>
<td>UNT</td>
<td>Yes</td>
</tr>
<tr>
<td>Listen Addresses</td>
<td>Yes</td>
</tr>
<tr>
<td>Talk Addresses</td>
<td>Yes</td>
</tr>
</tbody>
</table>
- Device Clear (DCL). Clears (initializes) all devices on the bus that have a device clear function, whether the controller has addressed them or not.

- 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 devices on the bus, 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 D: SCPI Conformance Information

All commands in the AWG500 Series Arbitrary Waveform Generator are based on SCPI Version 1995.0. Table D-1 lists the commands supported by the generator.

Table D-1: SCPI conformance information

<table>
<thead>
<tr>
<th>Command</th>
<th>Defined in SCPI 1995.0</th>
<th>Not defined in SCPI 1995.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWGcontrol</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOUTput</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="?">STATE</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EVENT</td>
<td>[IMMediate]</td>
</tr>
<tr>
<td></td>
<td>LOGic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RMODE(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSTATE?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RUN</td>
<td>[IMMediate]</td>
</tr>
<tr>
<td></td>
<td>SRESTore</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSAVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[IMMediate]</td>
<td></td>
</tr>
<tr>
<td>CALibration</td>
<td><a href="?">ALL</a></td>
<td></td>
</tr>
<tr>
<td>DIAGnostic</td>
<td>DATA?</td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="?">IMMediate</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SElect(?)</td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>BRIGHTness(?)</td>
<td></td>
</tr>
<tr>
<td>HCOPY</td>
<td>DESTination</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEViCE</td>
<td>LANGUAGE(?)</td>
</tr>
<tr>
<td></td>
<td>[IMMediate]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDUMP</td>
<td>[IMMediate]</td>
</tr>
</tbody>
</table>
### Table D-1: SCPI conformance information (Cont.)

<table>
<thead>
<tr>
<th>Command</th>
<th>Defined in SCPI 1995.0</th>
<th>Not defined in SCPI 1995.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MMEMory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATalog?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CDIrectory(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>COPY</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>DELetE</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FEED(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>INITIALize</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MDIRectory</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MSIS(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MOVE</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>NAME(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>OPEN</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>OUTPut</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER</td>
<td>[LPASS]</td>
<td>FREQUncy(?)</td>
</tr>
<tr>
<td>ISTate(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><a href="?">STATE</a></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>SOURce</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMBine</td>
<td>FEED(?)</td>
<td></td>
</tr>
<tr>
<td>FREQuncy</td>
<td>[CW</td>
<td>FIXed](?)</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>USER(?)</td>
<td></td>
</tr>
<tr>
<td>MARKer</td>
<td>DELay(?)</td>
<td></td>
</tr>
<tr>
<td>VOLTage</td>
<td>[LEVEL] JMMEDIATE]</td>
<td>HIGH(?)</td>
</tr>
<tr>
<td></td>
<td>VOLTage [LEVEL] JMMEDIATE]</td>
<td>LOW(?)</td>
</tr>
<tr>
<td>POWER</td>
<td>[LEVEL] JMMEDIATE]</td>
<td><a href="?">AMPLitude</a></td>
</tr>
<tr>
<td>ROSScillator</td>
<td>SOURce(?)</td>
<td></td>
</tr>
<tr>
<td>VOLTage</td>
<td>[LEVEL] JMMEDIATE]</td>
<td><a href="?">AMPLitude</a></td>
</tr>
<tr>
<td></td>
<td>OFFSET(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HIGH(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOW(?)</td>
<td></td>
</tr>
</tbody>
</table>
### Table D-1: SCPI conformance information (Cont.)

<table>
<thead>
<tr>
<th>Command</th>
<th>Defined in SCPI 1995.0</th>
<th>Not defined in SCPI 1995.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPERation</td>
<td>[EVENT]?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CONDITION?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>ENABLE(?)</td>
<td>✓</td>
</tr>
<tr>
<td>QUESTIONable</td>
<td>[EVENT]?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CONDITION?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>ENABLE(?)</td>
<td>✓</td>
</tr>
<tr>
<td>PREset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUEue</td>
<td>[NEXT]?</td>
<td></td>
</tr>
<tr>
<td>SYSTem</td>
<td>BEEper [IMMediate]</td>
<td>✓</td>
</tr>
<tr>
<td>COMMunicate</td>
<td>LAN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FTP [SERVER] <a href="?">STATE</a></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>GATEway ADDRESS(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PING?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>RDEVice ADDRESS(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>FSYSTEM(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>NAME(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>PROTOCOL(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td><a href="?">STATE</a></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>[SELF] ADDRESS(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>SMASK(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>DATE(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERRor?</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>KDIREction(?)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>KEYBoard <a href="?">TYPE</a></td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>KLOCK(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SECURITY IMMediate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TIME(?)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UPTime?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VERSION?</td>
<td></td>
</tr>
</tbody>
</table>
Table D-1: SCPI conformance information (Cont.)

<table>
<thead>
<tr>
<th>Command</th>
<th>Defined in SCPI 1995.0</th>
<th>Not defined in SCPI 1995.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIGger</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[SEQUence]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[IMMEDIATE]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMPedance(?)</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>LEVel(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>POLarity(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SLOPe(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SOURce(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TIMer(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*CLS</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*ESE(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*ESR?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*IDN?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*OPC(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*OPT?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*RST</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*SRE(?)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*STB?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*TST?</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>*WA1?</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Appendix E: Factory Initialization Settings

The following table lists the commands affected by a factory initialization.

The SYSTem:SECurity:IMMEDIATE command initializes all the settings as shown below, while the *RST command has no effect on the Status commands and the SYSTem:COMMunicate:LAN commands.

Table E-1: Factory initialization settings

<table>
<thead>
<tr>
<th>Header</th>
<th>Default settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWGcontrol commands</td>
<td></td>
</tr>
<tr>
<td>AWGControl: DOUTput[1</td>
<td>2]</td>
</tr>
<tr>
<td>AWGControl: RMODE</td>
<td>CONTInuous</td>
</tr>
<tr>
<td>Diagnostic commands</td>
<td></td>
</tr>
<tr>
<td>DIAGnostic: SELECT</td>
<td>ALL</td>
</tr>
<tr>
<td>Display commands</td>
<td></td>
</tr>
<tr>
<td>DISPLAY: BRIGHTness</td>
<td>0.7</td>
</tr>
<tr>
<td>Hardcopy commands</td>
<td></td>
</tr>
<tr>
<td>HCopy: DEVeice: LANguage</td>
<td>BMP</td>
</tr>
<tr>
<td>Memory commands</td>
<td></td>
</tr>
<tr>
<td>MMemory: CDIRectory</td>
<td>&quot;/&quot;</td>
</tr>
<tr>
<td>MMemory: FEED</td>
<td>&quot;HCOP&quot;</td>
</tr>
<tr>
<td>MMemory: MSIS</td>
<td>&quot;MAIN&quot;</td>
</tr>
<tr>
<td>MMemory: NAME</td>
<td>&quot;HARDCOPY&quot;,&quot;MAIN&quot;</td>
</tr>
<tr>
<td>Output commands</td>
<td></td>
</tr>
<tr>
<td>OUTPut[1</td>
<td>2]: FILTers: LPASs: FRequency</td>
</tr>
<tr>
<td>OUTPut[1]: ISTATe</td>
<td>0</td>
</tr>
<tr>
<td>OUTPut[1</td>
<td>2</td>
</tr>
</tbody>
</table>
### Table E-1: Factory initialization settings (Cont.)

<table>
<thead>
<tr>
<th>Header</th>
<th>Default settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Commands</strong></td>
<td></td>
</tr>
<tr>
<td>[SOURce1]:COMBine:FEED</td>
<td>&quot; &quot; (null)</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>SOURCe7:POWER[:LEVel][:IMMediate][:AMPLitude]</td>
<td>-105</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2]][:VOLTage[:LEVel][:IMMediate][:AMPlitude]]</td>
</tr>
<tr>
<td>SOURCe5:VOLTage[:LEVel][:IMMediate]:HIGH</td>
<td>2.00</td>
</tr>
<tr>
<td>SOURCe5:VOLTage[:LEVel][:IMMediate]:LOW</td>
<td>0.00</td>
</tr>
<tr>
<td>[SOURce[1</td>
<td>2]][:VOLTage[:LEVel][:IMMediate]:OFFSet]</td>
</tr>
<tr>
<td><strong>Status Commands</strong></td>
<td></td>
</tr>
<tr>
<td>*ESE^1</td>
<td>0</td>
</tr>
<tr>
<td>*PSC^1</td>
<td>1</td>
</tr>
<tr>
<td>*SRE^1</td>
<td>0</td>
</tr>
<tr>
<td>STATus:OPERation:ENABLE^1</td>
<td>0</td>
</tr>
<tr>
<td>STATus:QUESTionable:ENABLE^1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table E-1: Factory initialization settings (Cont.)

<table>
<thead>
<tr>
<th>Header</th>
<th>Default settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Commands</strong></td>
<td></td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:GATeway&lt;x&gt;:ADDReSS1</td>
<td>&quot;) (null)</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:RDEVice&lt;x&gt;:ADDReSS1</td>
<td>&quot;) (null)</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:RDEVice&lt;x&gt;:FSYSteM1</td>
<td>&quot;) (null)</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:RDEVice&lt;x&gt;:NAME1</td>
<td>&quot;NET&lt;x&gt;&quot;</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:RDEVice&lt;x&gt;:PROTocol1</td>
<td>NFS</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN:RDEVice&lt;x&gt;[:STATe]1</td>
<td>0</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN[:SELF]:ADDReSS1</td>
<td>&quot;) (null)</td>
</tr>
<tr>
<td>SYSTem:COMMunicate:LAN[:SELF]:SMASK1</td>
<td>&quot;) (null)</td>
</tr>
<tr>
<td>SYSTem:KDIRection</td>
<td>FORWa rd</td>
</tr>
<tr>
<td>SYSTem:KEYBoard[:TYPE]</td>
<td>ASCII</td>
</tr>
<tr>
<td>SYSTem:KLOCK</td>
<td>0</td>
</tr>
<tr>
<td><strong>Trigger Commands</strong></td>
<td></td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:IMPedance</td>
<td>5.0E+1</td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:LEVEL1</td>
<td>1.4</td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:POLarity</td>
<td>POSitive</td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:SLOPe</td>
<td>POSitive</td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:SOURce</td>
<td>EXternal</td>
</tr>
<tr>
<td>TRIGger[:SEQUence]:TIMER</td>
<td>1.00E-1</td>
</tr>
</tbody>
</table>

1 These commands are not affected by the *RST command.
Glossary

ASCII
Acronym for the American Standard Code for Information Interchângé. 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 to it.

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.
Symbols

*CAL?, 2–31
*CLS, 2–33
*ESE, 2–37
*ESR, 2–38
*IDN?, 2–40
*OPC, 2–50
*OPT?, 2–51
*PSC, 2–54
*RST, 2–55
*SRE, 2–64
*STB?, 2–70
*TRG, 2–84
*TST?, 2–89
*WAI, 2–90

A
Abbreviations, commands, queries, and parameters, 2–5
ABORt, 2–26
Arguments, parameters, 2–4
AWG control command group, 2–16
AWG control commands
   AWGControl:DOUTput[12];[:STATe], 2–26
   AWGControl:EVENt[:LOGic][:IMMediate], 2–27
   AWGControl:RMONi, 2–27
   AWGControl:RSTate?, 2–28
   AWGControl:RUN[:IMMediate], 2–29
   AWGControl:SREStore, 2–29
   AWGControl:SSAve, 2–30
   AWGControl:STOP[:IMMediate], 2–31
AWGControl:DOUTput[12];[:STATe], 2–26
AWGControl:EVENt[:LOGic][:IMMediate], 2–27
AWGControl:RMONi, 2–27
AWGControl:RSTate?, 2–28
AWGControl:RUN[:IMMediate], 2–29
AWGControl:STOP[:IMMediate], 2–31

B
Backus-Naur Form, 2–1
BNF (Backus-Naur form), 2–1

C
Calibration command group, 2–16
Calibration commands
   *CAL?, 2–31
   CALibration[:ALL], 2–32
   CALibration[:ALL], 2–32
   Case sensitivity, 2–8
   Character chart, A–1
   Code, error, 3–15
   Code Convert File, 2–98
   Command, summaries, 2–16
   Command group
      AWG control, 2–16
      Calibration, 2–16
      Diagnostic, 2–17
      Display, 2–17
      Hardcopy, 2–18
      Mass memory, 2–19
      Output, 2–20
      Source, 2–20
      Status, 2–22
      Synchronization, 2–22
      System, 2–23
      Trigger, 2–24
   Command syntax, 2–1
   Commands
      chaining, 2–6
      Parts of, 1–1
      structure of IEEE 488.2 commands, 2–9
   Creating commands, 2–3

D
Data transfer, 2–93
DCL, C–3
Device Clear, C–3
Diagnostic command group, 2–17
Diagnostic commands
   *TST?, 2–89
   DIAgnostic:DATA?, 2–34
   DIAgnostic:SELECT, 2–35
   DIAgnostic[:IMMediate], 2–34
   DIAgnostic:DATA?, 2–34
   DIAgnostic:SELECT, 2–35
   DIAgnostic[:IMMediate], 2–34
   Diagram, syntax, 2–12
   Disks included with this manual, 1–3
   Display command group, 2–17
   Display commands, DISPlay:BRIghtness, 2–36
   DISPlay:BRIghtness, 2–36
Index

E
Enable registers, 3–8
Error code, 3–15
Error codes
   AWG 500 unique, 3–22
   commands, 3–16
   device specific, 3–20
   execution, 3–18
   hardware, 3–20, 3–22
   query, 3–21
Error message, 3–15
Example programs, 1–3

F
Factory initialization settings, E–1
File
   Code Convert, 2–98
   Pattern, 2–95
   Sequence, 2–96
   Waveform, 2–94

G
GET, C–3
Go to local, C–3
GPIB
   Configurations, 1–5
   Connection rules, 1–5
   interface messages, C–3
   interface specification, C–1
   Setting parameters, 1–6
Group execute trigger, C–3
GTL, C–3

H
Hardcopy command group, 2–18
Hardcopy commands
   HCOPy:DESTination, 2–38
   HCOPy:DEVice:LANGuage, 2–39
   HCOPy:SDUMp:[IMMediate], 2–40
   HCOPy:[IMMediate], 2–39
HCOPy:DESTination, 2–38
HCOPy:DEVice:LANGuage, 2–39
HCOPy:SDUMp:[IMMediate], 2–40
HCOPy:[IMMediate], 2–39

I
IEEE 488.2 common commands, 2–9
IEEE Std 488.2-1987, 1–3
Instrument setup, 1–3
Interface message, C–3

L
LLO, C–3
Local lock out, C–3

M
Mass memory command group, 2–19
Mass memory commands
   MMEMory:CATalog?, 2–41
   MMEMory:CDIRectory, 2–42
   MMEMory:CLOSE, 2–43
   MMEMory:COPY, 2–43
   MMEMory:DATA, 2–44
   MMEMory:DELETE, 2–44
   MMEMory:FEED, 2–45
   MMEMory:INITialize, 2–46
   MMEMory:MDIREctory, 2–47
   MMEMory:MOVE, 2–48
   MMEMory:MSIS, 2–47
   MMEMory:NAME, 2–49
   MMEMory:OPEN, 2–50
Message, error, 3–15
MMEMory:CATalog?, 2–41
MMEMory:CDIREctory, 2–42
MMEMory:CLOSE, 2–43
MMEMory:COPY, 2–43
MMEMory:DATA, 2–44
MMEMory:DELETE, 2–44
MMEMory:FEED, 2–45
MMEMory:INITialize, 2–46
MMEMory:MDIREctory, 2–47
MMEMory:MOVE, 2–48
MMEMory:MSIS, 2–47
MMEMory:NAME, 2–49
MMEMory:OPEN, 2–50
Mnemonics, Constructed, 2–10

O
Output command group, 2–20
Output commands
  OUTPut[1]:INSTate, 2–53
  OUTPut[12]:FILTer[.LPASS]:FREQuency, 2–52
  OUTPut[12]:[STATE], 2–52
  OUTPut[1]:ISTate, 2–53
  OUTPut[12]:FILTer[.LPASS]:FREQuency, 2–52
  OUTPut[12]:[STATE], 2–52

P

Parallel poll, C–3
Parameter setting, GPIB, 1–6
Parameter types used in syntax descriptions, 2–4
Parts of commands, 1–1
Pattern File, 2–95
PPC, C–3
PPD, C–3
PPE, C–3
PPU, C–3
Programming Examples, 4–1

Q

Queries, 2–3
Query Responses, 2–3
Queues, 3–10
event, 3–10
output, 3–10
Quotes, 2–8

R

Registers, 3–4
  Event Status Enable Register (ESER), 3–8
  Operation Condition Register (OCR), 3–7
  Operation Enable Register (OENR), 3–9
  Operation Event Register (OEVr), 3–7
  Questionable Condition Register (OCR), 3–7
  Questionable Enable Register (QENR), 3–9
  Questionable Event Register (QEVr), 3–8
  Service Request Enable Register (SREr), 3–8
  Standard Event Status Register (SERS), 3–6
  Status Byte Register (SBR), 3–5
Reserved words, B–1
Rules, for using SCPI commands, 2–8

S

SCPI
  abbreviating, 2–5
  chaining commands, 2–6
  commands, 2–2
  conformation information, D–1
  general rules, 2–8
  parameter types, 2–4
  subsystem hierarchy tree, 2–2
  SCPI commands and queries syntax, 2–2–2–8
  creating commands, 2–3
  creating queries, 2–3
  SDC, C–3
  Selected device clear, C–3
  Sequence File, 2–96
  Serial poll
    Disable, C–3
    Enable, C–3
  Setup, Instrument preparation, 1–3
  SI prefix and unit, 2–7
  Source command group, 2–20
  Source commands
    [SOURce[1]:COMBine:FEED, 2–56
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      :OFFSET, 2–64
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:AMPLitude], 2–61
    [SOURce[125]:FREQuency[:CWFIXEd], 2–56
    [SOURce[125]:FUNCtion:USER, 2–57
    [SOURce[125]:MARKer[12]:DELay, 2–58
    [SOURce[125]:MARKer[12]:VOLTage[:LEVel]
      [:IMMediate]:HIGH, 2–59
    [SOURce[125]:MARKer[12]:VOLTage[:LEVel]
      [:IMMediate]:LOW, 2–59
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:IMMediate]:HIGH, 2–62
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]:LOW, 2–63
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:AMPlitude], 2–60
    [SOURce[1]:COMBine:FEED, 2–56
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      :OFFSET, 2–64
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:AMPlitude], 2–61
    [SOURce[1]:FREQuency[:CWFIXEd], 2–56
    [SOURce[1]:FUNCtion:USER, 2–57
    [SOURce[1]:MARKer[12]:DELay, 2–58
    [SOURce[1]:MARKer[12]:VOLTage[:LEVel]
      [:IMMediate]:HIGH, 2–59
    [SOURce[1]:MARKer[12]:VOLTage[:LEVel]
      [:IMMediate]:LOW, 2–59
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:IMMediate]:HIGH, 2–62
    [SOURce[1]:VOLTage[:LEVel][:IMMediate]
      [:IMMediate]:LOW, 2–63

AWG510 & AWG520 Arbitrary Waveform Generator Programmer Manual
Index

SOURce:POWer[:LEVel][:IMMediate][:AMPlitude], 2–60
SPD, C-3
SPE, C-3
Special characters, 2–5
Status and error commands
  STATUS:QUEStionable:ENABle, 3–9
Status and events, displaying on screen, 3–13
Status command group, 2–22
Status commands
  *CLS, 2–33
  *ESE, 2–37
  *ESR?, 2–38
  *PSC, 2–54
  *SRE, 2–64
  *STB?, 2–70
  STATUS:OPERation:CONDition?, 2–65
  STATUS:OPERation:ENABle, 2–66
  STATUS:OPERation[:EVENT]?, 2–66
  STATUS:PRESet, 2–67
  STATUS:QUEStionable:CONDition?, 2–67
  STATUS:QUEStionable:ENABle, 2–68
  STATUS:QUEStionable][:EVENT]?, 2–69
  STATUS:QUEUe[:NEXT]?, 2–69
Status registers, 3–1
Status reporting, 3–1
  STATUS:OPERation:CONDition?, 2–65
  STATUS:OPERation:ENABle, 2–66, 3–9
  STATUS:OPERation[:EVENT]?, 2–66
  STATUS:PRESet, 2–67
  STATUS:QUEStionable:CONDition?, 2–67
  STATUS:QUEStionable:ENABle, 2–68, 3–9
  STATUS:QUEStionable][:EVENT]?, 2–69
  STATUS:QUEUe[:NEXT]?, 2–69
Synchronization command group, 2–22
Synchronization commands
  *OPC, 2–50
  *WAI, 2–90
Synchronizing execution, 3–14
Syntax
  Command, 2–1
  diagrams, 2–12
Syntax diagrams, 1–1
System command group, 2–23
System commands
  *IDN?, 2–40
  *OPT?, 2–51
  *RST, 2–55
  SYSTEM:BEEPer[:IMMediate], 2–71
  SYSTEM:COMMunicate:LAN:FTP[:SERVer][:STATe], 2–71
  SYSTEM:COMMunicate:LAN:GATeway:ADDRess, 2–72
  SYSTEM:COMMunicate:LAN:PING?, 2–73
  SYSTEM:COMMunicate:LAN:RDEVice[12][3] :PROTocol, 2–75
  SYSTEM:COMMunicate:LAN:RDEVice[12][3] [:STATe], 2–76
  SYSTEM:COMMunicate:LAN[:SELF]:ADDRess, 2–77
  SYSTEM:COMMunicate:LAN[:SELF]:SMASK, 2–77
  SYSTEM:DATE, 2–78
  SYSTEM:ERRor?, 2–79
  SYSTEM:KDIRection, 2–80
  SYSTEM:KEYBoard[:TYPE], 2–80
  SYSTEM:KLOCK, 2–81
  SYSTEM:SECurity:IMMediate, 2–83
  SYSTEM:TIME, 2–82
  SYSTEM:UPTime, 2–83
  SYSTEM:VERsion?, 2–84
  SYSTEM:BEEPer[:IMMediate], 2–71
  SYSTEM:COMMunicate:LAN:FTP[:SERVer][:STATe], 2–71
  SYSTEM:COMMunicate:LAN:GATeway:ADDRess, 2–71
  SYSTEM:COMMunicate:LAN:PING?, 2–73
  SYSTEM:COMMunicate:LAN:RDEVice[12][3] [:STATe], 2–76
  SYSTEM:COMMunicate:LAN:RDEVice[12][3] :PROTocol, 2–75
  SYSTEM:COMMunicate:LAN:RDEVice[12][3] [:STATe], 2–76
  SYSTEM:COMMunicate:LAN[:SELF]:ADDRess, 2–77
  SYSTEM:COMMunicate:LAN[:SELF]:SMASK, 2–77
  SYSTEM:DATE, 2–78
  SYSTEM:ERRor?, 2–79
  SYSTEM:KDIRection, 2–80
  SYSTEM:KEYBoard[:TYPE], 2–80
  SYSTEM:KLOCK, 2–81
  SYSTEM:SECurity:IMMediate, 2–82
  SYSTEM:TIME, 2–82

Index-4

AWG510 & AWG520 Arbitrary Waveform Generator Programmer Manual
SYSTem:UPTime, 2–83
SYSTem:VERSion?, 2–84

T
TCT, C–3
Transfer, data, 2–93
Trigger command group, 2–24
Trigger commands
  *TRG, 2–84
  ABORt, 2–26
  TRIGger[:SEQuence]:IMPedance, 2–85
  TRIGger[:SEQuence]:LEVEL, 2–86
  TRIGger[:SEQuence]:POLarity, 2–86
  TRIGger[:SEQuence]:SLOPe, 2–87
  TRIGger[:SEQuence]:SOURce, 2–88
  TRIGger[:SEQuence]:TIMer, 2–88
  TRIGger[:SEQuence][:IMMediate], 2–85
  TRIGger[:SEQuence]:IMPedance, 2–85
  TRIGger[:SEQuence]:LEVEL, 2–86
  TRIGger[:SEQuence]:POLarity, 2–86
  TRIGger[:SEQuence]:SLOPe, 2–87
  TRIGger[:SEQuence]:SOURce, 2–88
  TRIGger[:SEQuence]:TIMer, 2–88
  TRIGger[:SEQuence][:IMMediate], 2–85

U
Unit and SI prefix, 2–7
UNL, C–3
Unlisten, C–3
UNT, C–3
Untalk, C–3

W
Waveform File, 2–94