Programmer Manual

SONY Tektronix

3026
Realtime Spectrum Analyzer

071-0419-00

This document supports firmware version 1.10.

Printed in Japan.

Sony/Tektronix Corporation, P.O.Box 5209, Tokyo Int'l, Tokyo 100–31 Japan

Tektronix, Inc., P.O. Box 1000, Wilsonville, OR 97070–1000

Table of Contents

List of Figures

List of Tables

Preface

This is the Programmer Manual for the 3026 Realtime Spectrum Analyzer. 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 for remote operation.
- *Syntax and Commands* defines the command syntax and processing conventions and describes each command in the realtime spectrum analyzer command set.
- *Status and Events* explains the status information and event messages reported by the realtime spectrum analyzer.
- *Appendices* contains various topics of use to the programmer.
- *Index* contains an index to this manual.

Related Manuals

Other documentation for the realtime spectrum analyzer includes:

- The *3026 User Manual* (Tektronix part number 071-0418-xx) describes the operation of the instrument.
- The *3026 Service Manual* (Tektronix part number 071-0420-xx) provides information for maintaining and servicing the Data Generator.

Getting Started

Getting Started

The 3026 Realtime Spectrum Analyzer has a GPIB interface. Almost all menu-controlled and front-panel controlled functions can be performed through the GPIB interface using the programming command set (described in *Command Syntax*).

The GPIB interface conforms to ANSI/IEEE Std 488.1-1987, which specifies the hardware interface, its basic functional protocol, and a set of interface messages (codes) that control the interface functions. This instrument also conforms to ANSI/IEEE Std 488.2-1987 which specifies Codes, Formats, Protocols, and Common Commands to support the system application. The functional layers of the GPIB system are shown in Figure 1-1.

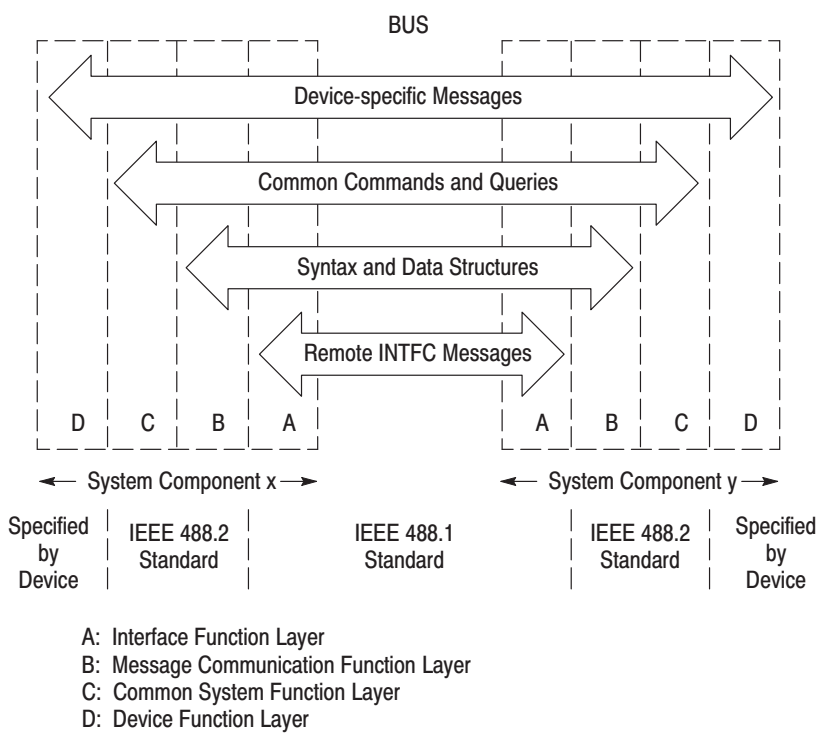


Figure 1-1: Functional layers in GPIB system

Installing for GPIB Communication

Connect a GPIB cable from the GPIB controller to the ANSI/IEEE Std 488 port (GPIB) connector on the rear panel of the realtime spectrum analyzer (see Figure 1-2). For example, when using an MS-DOS compatible controller, connect the GPIB cable between the National Instrument PC2A GPIB board and the realtime spectrum analyzer GPIB connector.

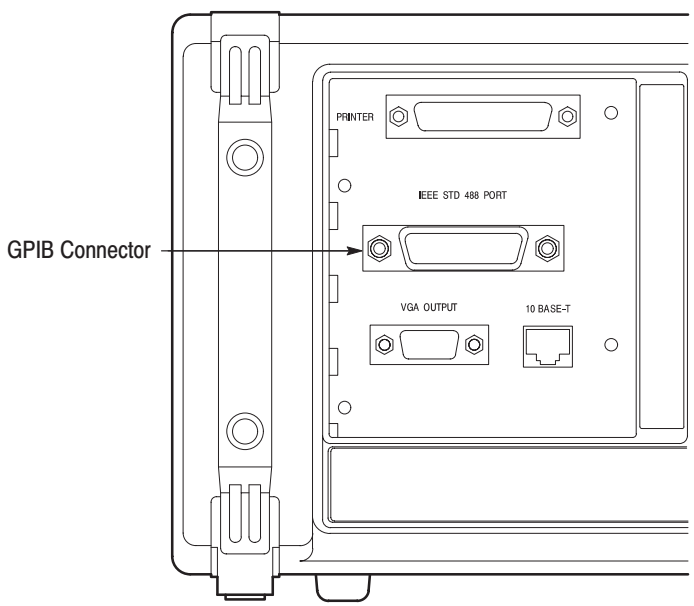


Figure 1-2: GPIB connector

Instruments can be connected to the GPIB in linear or star configurations or in a combination of both configurations. A linear hookup is one where a GPIB cable is used to string one device to a second, and then another GPIB cable is used to string from a second to a third, and so on until all devices in the system are connected. A star setup is one where one end of all the GPIB cables in the system are attached to one device. Refer to Figure 1-3 for these GPIB system configurations.

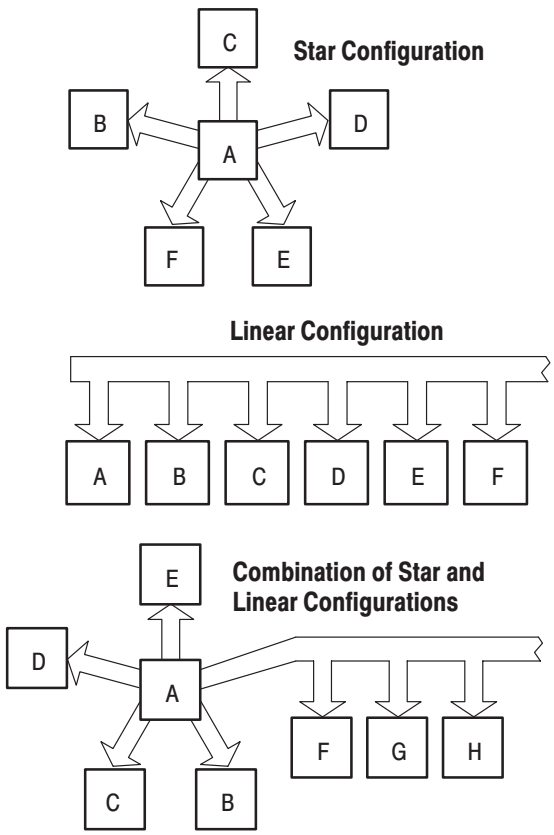


Figure 1-3: GPIB system configurations

Restrictions

Consider the following restrictions when distributing instruments on the GPIB bus:

1. No more than 15 total devices (including the controller) can be included on a signal bus.
2. In order to maintain the electrical characteristics of the bus, one device load must be connected for every two meters of cable (most often, each device represents one device load to the bus).
3. The total cable length (cumulative) must not exceed 66 feet (20 meters).
4. At least two-thirds of the device loads must be powered on.

Setting the GPIB Parameters

To set the GPIB parameters, proceed as follows:

1. Press the **UTILITY** button on the front panel.
2. Press the **GPIB** bottom button.
3. Press the **Talker Listener** side button.

   This sets the communication mode to Talker/Listener.

***NOTE**. The realtime spectrum analyzer accepts as a terminator either the software LF (Line Feed), sent as the last data byte, or the hardware EOI, with the EOI line asserted concurrently with the last data byte sent.*

4. Set the GPIB address using the rotary knob or the front-panel keypad. The value can be set from 1 to 30.

Command Syntax

Syntax

This section contains information on the Standard Commands for Programmable Instruments (SCPI) and IEEE 488.2 Common Commands you can use to program your 3026 Realtime Spectrum Analyzer. The information is organized in the following subsections:

- *SCPI Commands and Queries* – This subsection describes the SCPI command organization and syntax
- *IEEE 488.2 Common Commands* – This subsection lists the commands and argument structures that are common to all SCPI commands

SCPI Commands and Queries

SCPI is a standard created by a consortium that provides 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 format across all SCPI instruments, regardless of manufacturer. The realtime spectrum analyzer uses a command language based on the SCPI standard.

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.

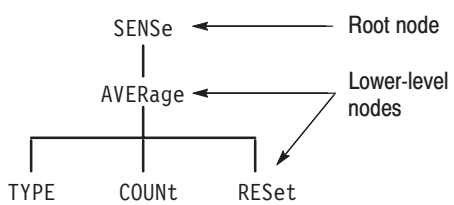


Figure 2-1: Example of SCPI subsystem hierarchy tree

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, SENSe is the root node and AVERage, TYPE, COUNt, and RESet are lower-level nodes. To create a SCPI command, start with the root node SENSe 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 2-15, list the valid values for all parameters.

For example, SENSe:AVERage:TYPE OFF 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. SENSe:AVERage:TYPE? is an example of a valid SCPI query using the hierarchy tree in Figure 2-1.

Command Arguments

Many commands accept either string or numeric arguments. For example: a boolean argument can either be “1” or “ON”.

Select signal parameter commands accept either a numeric value or one of the following strings:

MINimum. Use this argument to query the minimum value or set the parameter value to the minimum acceptable value.

MAXimum. Use this argument to query the maximum value or set the parameter value to the maximum acceptable value.

***NOTE**. If the realtime spectrum analyzer does not return a value in response to a MIN or MAX query, then the values are undefined and an error message is generated.*

Parameter Types

Every parameter in the command and query descriptions is of a specified type. The parameters are enclosed in brackets, such as <pattern>. The parameter type is listed after the parameter and is enclosed in parentheses, for example, (discrete). Some parameter types are defined specifically for the realtime spectrum analyzer command set and some are defined by ANSI/IEEE 488.2-1987 (see Table 2-1).

Table 2-1: Parameter types used in syntax descriptions

Parameter Type	Description	Example
binary	Binary numbers	#B0110
arbitrary block[1]	A specified length of arbitrary data	#512234xxxxx . . . where 5 indicates that the following 5 digits (12234) specify the length of the data in bytes; xxxxx ... indicates the data
boolean	Boolean numbers or values	ON or 1 OFF or 0
discrete	A list of specific values	MIN, MAX, UP, DOWN
hexadecimal[2]	Hexadecimal numbers (0–9, A, B, C, D, E, F)	#HAA, #H1
NR1[2,3] numeric	Integers	0, 1, 15, –1
NR2[2] numeric	Decimal numbers	1.2, 3.141516, –6.5
NR3[2] numeric	Floating point numbers	3.1415E–9, –16.1E5
NRf[2] numeric	Flexible decimal number that may be type NR1, NR2 or NR3	See NR1, NR2, NR3 examples
string[4]	Alphanumeric characters (must be within quotation marks)	“Testing 1, 2, 3”

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

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

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

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

Abbreviating Commands, Queries, and Parameters

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

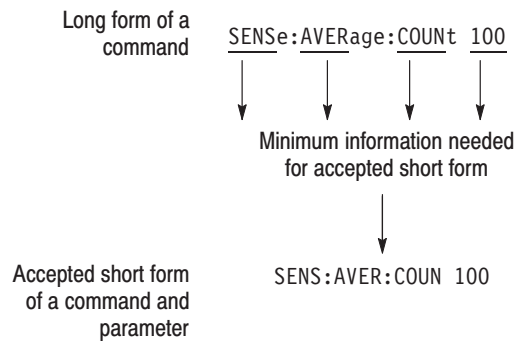


Figure 2-2: Example of abbreviating a command

Chaining Commands and Queries

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

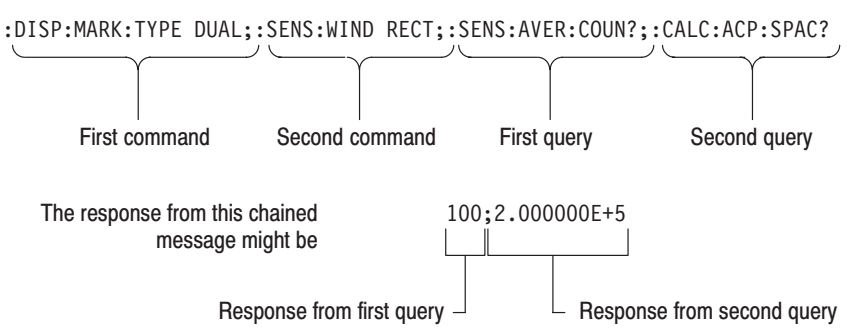


Figure 2-3: Example of chaining commands and queries

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

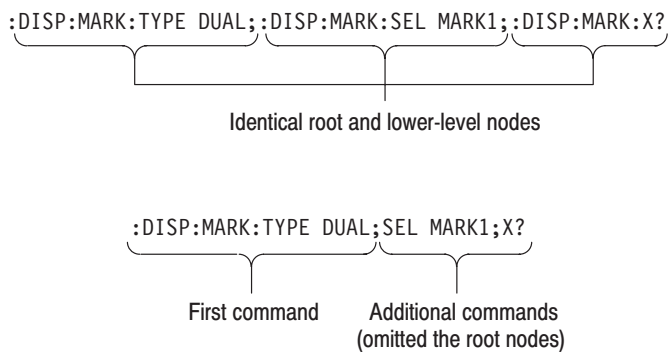


Figure 2-4: Example of omitting root and lower-level nodes in a chained message

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.

  ```
  HCOPY:DEVICE:DESTINATION PRINTER
  ```

  is the same as

  ```
  hcopy:device:destination printer
  ```

  and

  ```
  HCOPY:device:destination PRINTER
  ```

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

- No embedded spaces are allowed between or within nodes.

  correct: `HCOPY:DEVICE:DESTINATION PRINTER`

  incorrect: `HCOPY: DEVICE: DESTINATION PRIN TER`

IEEE 488.2 Common Commands

Description

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

Command and Query Structure

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

- `*ESE 16`
- `*CLS`

The following are examples of common queries:

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

Backus-Naur Form Definition

This manual may describe commands and queries using the Backus-Naur Form (BNF) notation. Table 2-2 defines the standard BNF symbols:

Table 2-2: BNF symbols and meanings

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

Message Terminators

This manual uses <EOM> (End of message) to represent a message terminator.

Symbol	Meaning
<EOM>	Message terminator

GPIB. The end-of-message terminator may be the END message (EOI asserted concurrently with the last data byte), the ASCII code for line feed (LF) sent as the last data byte, or both. The realtime spectrum analyzer always terminates messages with LF and EOI. It allows white space before the terminator.

Command Groups

This subsection describes the organization of the 3026 Realtime Spectrum Analyzer command as a number of functional groups. (See subsection *Command Descriptions* on page 2-15 for a complete description of each command in alphabetical order.)

Throughout this section, the parenthesized question symbol (?) follows the command header to indicate that both a command and query form of the command can be used.

Command Summaries

Tables 2-3 through 2-13 list the command that are part of the 10 functional groups.

CALCULATE Commands

Use these commands to control settings related to the measurement functions such as the kind of measurement and parameter settings, and to query measurement results and readiness state.

Table 2-3: CALCULATE commands

Header	Description
`CALCulate[1|2]?`	Query measurement item and measurement result
`CALCulate[1|2]:ACP: BANDwidth(?)`	Set or query ACP channel bandwidth
`CALCulate[1|2]:ACP:SPACing(?)`	Set or query ACP channel interval
`CALCulate[1|2]:FUNCtion(?)`	Set or query measurement item
`CALCulate[1|2]:FUNCtion: RESult?`	Query measurement results
`CALCulate[1|2]:FUNCtion: RESult:READy?`	Query measurement result is available or not.
`CALCulate[1|2]:OBW:RATE(?)`	Set or query OBW power rate

DISPLAY Commands

Use these commands to select display format, cursors, and perform other display related functions.

Table 2-4: DISPLAY Commands

Header	Description
DISPlay?	Query all the settings for display function
DISPlay:CURrent:TRACe(?)	Set or query the currently selected trace
DISPlay:CURrent:WINDow(?)	Set or query the current window
DISPlay:FORMat(?)	Set or query the display format
DISPlay:MARKer:FNUMber(?)	Move the active cursor to the specified flame or query the flame the current marker is on
DISPLay:MARKer:PEAK	Move the current marker to the peak point
DISPlay:MARKer:PEAK:SEPara-tion(?)	Set or query the peak search interval for the peak find mode.
DISPlay:MARKer:SELect(?)	Set or query the current marker
DISPlay:MARKer:TYPE(?)	Set or query the type(s) of the marker(s)
DISPlay:MARKer:X(?)	Set or query the horizontal position of the current marker
DISPlay:MARKer:X:UNIT?	Query the horizontal display unit for the current marker
DISPlay:MARKer:Y?	Query the vertical position of the current marker
DISPlay:MARKer:Y:UNIT?	Query the vertical display unit for the current marker
DISPlay:MENU[:NAME](?)	Set or query the menu selection state
DISPlay:MENU:STATe(?)	Set or query the menu display on/off state
DISPlay:TRACe:ACTIVE(?)	Set or query the input waveform display on/off state
DISPlay:TRACe:AVERage(?)	Set or query the average/peak hold mode on/off state
DISPlay:TRACe:REFerence(?)	Set or query the reference waveform display on/off state
DISPlay:WINDow[1|2]:TYPE(?)	Set or query the display mode

HARDCOPY Commands

Use these commands to execute hardcopy operations, and select the output port and output format.

Table 2-5: HARDCOPY commands

Header	Description
HCOPy?	Query all hardcopy related information
HCOPy:DEVice:DESTination(?)	Set or query output port of hardcopy
HCOPy:DEVice:LANGuage(?)	Set or query output format of hardcopy
HCOPy:IMMediate	Start a hardcopy operation

MEMORY Commands

Use these commands to control file and directory operations.

Table 2-6: MEMORY commands

Header	Description
MMEMory:CDIRectory(?)	Set or query the current working directory
MMEMory:MDIRectory	Make a new directory on the disk
MMEMory:RSETUP	Restore instrument setup data from a disk file
MMEMory:RTMASK	Restore trigger mask data from a disk file
MMEMory:RWAVE	Restore waveform data from a disk file
MMEMory:SAWAVe	Write averaged waveform data to a disk file
MMEMory:SORT(?)	Set or query disk file and directory information
MMEMory:SSETUP	Write instrument setup data to a disk file
MMEMory:STMASK	Write trigger mask data to a disk file
MMEMory:STWAVe	Write waveform data to a disk file as a text format
MMEMory:SWAVE	Write waveform data to a disk file

SENSE Commands

Use these commands to control settings related to data acquisition, and query acquisition data.

Table 2-7: SENSE commands

Header	Description
SENSe?	Query the all settings related to the data acquisition
SENSe:ACQuisition[:MODE](?)	Set or query the data acquisition mode
SENSe:ADEMod(?)	Set or query the analog demodulated signal

Table 2-7: SENSE commands (Cont.)

Header	Description
SENSe:AVERage:COUNt(?)	Set or query the number of times of averaging
SENSe:AVERage:RESET	Reset the average/peak hold mode
SENSe:AVERage:TYPE(?)	Set or query the average /peak hold mode
SENSe:BLOCK[:SIZE](?)	Set or query the block size
SENSe:DATa?	Query the data for the logical flame that active marker is on
SENSe:FFT[:SIZE](?)	Set or query the number of FFT sampling points
SENSe:FRAMe:PERIOD(?)	Set or query the frame period
SENSe:FREQuency:CENTer(?)	Set or query the center frequency
SENSe:FREQuency:SPAN(?)	Set or query the frequency span setting
SENSe:GAIN:EXTernal:GAIN(?)	Set or query the gain correction value
SENSe:GAIN:EXTernal:STATe(?)	Set or query the gain correction on/off state
SENSe:LEVel(?)	Set or query the reference level
SENSe:LEVel:UNIT(?)	Set or query the display unit for vertical axis
SENSe:RF(?)	Set or query the input range
SENSe:WINDow[:TYPE](?)	Set or query the FFT window type

SOURCE Commands

Use these commands to select the reference clock signal source.

Table 2-8: SOURCE commands

Header	Description
SOURce:ROSCillator:SOURce(?)	Set or query the clock signal internal/external selection

STATUS Commands

Use these commands to address the instrument status and event queue.

Table 2-9: STATUS commands

Header	Description
STATus:OPERation[:EVENt]?	Destructive query of status register
STATus:OPERation:ENABle(?)	Set or query register to record event transitions
STATus:PRESet	Reset all status enable register
STATus:QUEue[:NEXT]?	Display event in error/event queue

Table 2-9: STATUS commands (Cont.)

Header	Description
STATus:QUEStionable[:EVENT]?	Destructive query of status register
SRATus:QUEStionable:CONDition?	Query condition register
STATus:QUEStionable:ENABle(?)	Set or query register to record event transitions

SYSTEM Commands

Use these commands to set system parameters such as system date and time.

Table 2-10: SYSTEM commands

Header	Description
SYSTem:DATE(?)	Set or query system data <year,month,day>
SYSTem:ERRor(?)	Display the event in the error or event queue
SYSTem:FTPD[:STATe](?)	Set or query the FTP setting
SYSTem:TIME(?)	Set or query system time <hour,minute,second>
SYSTem:VERSion?	List SCPI compliance version

TRIGGER Commands

Use these commands to set the trigger conditions for the internal and external trigger source.

Table 2-11: TRIGGER commands

Header	Description
TRIGger?	Query all the current trigger-related settings
TRIGger:COUNt(?)	Set or query the block count
TRIGger:FREQMASK:CONDition(?)	Set or query the trigger condition to trigger using a mask pattern
TRIGger:LEVel(?)	Set or query the level of the external trigger signal that generates the triggering event
TRIGger:MODE(?)	Set or query the trigger mode
TRIGger:POSition(?)	Set or query the trigger position
TRIGger:SOURce(?)	Set or query the trigger source

COMMON Commands

Common commands have a "*" prefix and address all of the installed modules.

Table 2-12: COMMON commands

Header	Description
*CAL?	Execute gain calibration and return its results
*CLS	Clear SESR, SBR and Event Queue
*ESE(?)	Set and query ESER
*ESR?	Query SESR
*IDN?	Query ID information about the data generator
*RCL	Recall instrument settings from the specified register
*SAV	Save instrument settings to the specified register
*SRE(?)	Set or query SRER
*STB?	Query SBR
*TST?	Perform self-test

Other Commands

This group is a collection of commands that cannot be classified in any other group.

Table 2-13: Other commands

Header	Description
ABORt	Stop data acquisition
INITiate[:IMMediate]	Restart data acquisition
RUNNing?	Query whether the instrument is either acquiring data or waiting for a trigger

Command Descriptions

This subsection lists each command and query in the 3026 Spectrum Analyzer command set alphabetically. Each command entry includes its command description and command group, its related commands (if any), its syntax, and its 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 DISPlay:FORMat command, just type DISP:FORM.

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. If neither symbol follows the command, it can only be used as a command.

ABORt

The ABORt command stops data acquisition. It performs the same function as the front panel STOP key except that acquisition always stops at the end of the current sweep. Refer to the *User manual* for data acquisition details.

Group Other

Related Commands None

Syntax ABORt

Examples ABORt
stops data acquisition.

*CAL?

The *CAL? query executes gain calibration (RFCal) for the instrument and returns its result. If execution of this command is attempted when the instrument has not warmed up, an error code (a non–zero value) returns without executing the calibration.

Group COMMON

Related Commands None

Syntax *CAL?

Responses <numeric value>
<numeric value>::=<NR1> indicates either of the following:

Zero value	the calibration is complete without error.
Non-zero value	an error has been detected.

Examples *CAL?
executes gain calibration and then returns the result.

CALCulate[1|2]?

The CALCulate[1|2]? query returns the measurement item and the result of measurement.

Group CALCULATE

Related Commands CALCulate[1|2]:FUNCtion, CALCulate[1|2]:FUNCtion:RESult

Syntax CALCulate[1|2]?

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments None

Responses {OFF|OBW|CN|ACP|POWER|NOISE},<NR3>[,<NR3>];<NR3>;<NR3>

Examples

```
CALCULATE[1|2]?
```

might return `CN;69.3579,106.626;;`

CALCulate[1|2]:ACP:BANDwidth(?)

The `CALCulate[1|2]:ACP:BANDwidth` command sets the bandwidth used for ACP measurement. The `CALCulate[1|2]:ACP:BANDwidth?` query returns the current bandwidth setting.

Group

CALCULATE

Related Commands

`CALCulate[1|2]:ACP:SPACing`

Syntax

```
CALCulate[1|2]:ACP:BANDwidth <numeric value>
CALCulate[1|2]:ACP:BANDwidth?
```

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments

`<numeric value>::=<NR3>`
where `<NR3>` is a bandwidth ranging from 1 Hz to 2E+6 Hz.

Responses

`<NR3>`

Examples

```
:CALCULATE:ACP:BANDWIDTH 3E+5
```

sets the bandwidth for ACP measurement to 300 kHz.

CALCulate[1|2]:ACP:SPACing(?)

The CALCulate[1|2]:ACP:SPACing command sets the channel interval used for ACP measurement. The CALCulate[1|2]:ACP:SPACing? query returns the current channel interval setting.

Group CALCULATE

Related Commands CALCulate[1|2]:ACP:BANDwidth

Syntax

```
CALCulate[1|2]:ACP:SPACing <numeric value>
CALCulate[1|2]:ACP:SPACing?
```

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments

<numeric value>::=<NR3>
where <NR3> is a bandwidth ranging from 1 Hz to 2E+6 Hz.

Responses <NR3>

Examples

```
:CALCULATE:ACP:SPACING 6E+5
```

sets the channel interval for ACP measurement to 600 kHz.

CALCulate[1|2]:FUNCtion(?)

The CALCulate[1|2]:FUNCtion command selects the measurement item. The CALCulate[1|2]:FUNCtion? query returns the current selected measurement item.

Group CALCULATE

Related Commands None

Syntax

```
CALCulate[1|2]:FUNCtion {OFF|OBW|CN|ACP|POWER|NOISE}
CALCulate[1|2]:FUNCtion?
```

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments

OFF	disables measurement.
OBW	enables OBW measurement.
CN	enables C/N and C/No measurement.
ACP	enables ACP measurement.
POWER	enables power measurement.
NOISE	enables noise level measurement.

Responses {OFF|OBW|CN|ACP|POWER|NOISE}

Examples

```
:CALCULATE:FUNCTION POWER
```

enables power measurement.

CALCulate[1|2]:FUNCtion:RESult?

The CALCulate[1|2]:FUNCtion:RESult? query returns the result of measurement.

Group CALCULATE

Related Commands None

Syntax

```
CALCulate[1|2]:FUNCtion:RESult?
```

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments None

Responses <NR3>[,<NR3>]

Examples

```
:CALCULATE:FUNCTION:RESULT?
```

during C/N measurement might return

```
−60.4917,−23.2237
```

In this case, the first value indicates the result of the C/N measurement, and the second value indicates the result of the C/No measurement.

CALCulate[1|2]:FUNCtion:RESult:READy?

The CALCulate[1|2]:FUNCtion:RESult:READy? query returns whether the query about the result of measurement is enabled.

Group CALCULATE

Related Commands None

Syntax CALCulate[1|2]:FUNCtion:RESult READy?

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments None

Responses

1	the query about the result of measurement is enabled.
0	the query about the result of measurement is disabled.

Examples

```
:CALCULATE:FUNCTION:RESULT:READY?
```

might return 1

This indicates that the query is enabled.

CALCulate[1|2]:OBW:RATE(?)

The CALCulate[1|2]:OBW:RATE command sets the power rate of the carrier signal relative to the total power for the OBW measurement. The CALCulate[1|2]:OBW:RATE? query returns the current power rate setting.

Group CALCULATE

Related Commands None

Syntax

```
CALCulate[1|2]:OBW:RATE <numeric value>
CALCulate[1|2]:OBW:RATE?
```

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments <numeric value>::=<NR2>
where <NR2> is a power rate ranging from 90.0 % to 99.8 %.

Responses <NR2>

Examples :CALCULATE:OBW:RATE 99
sets the power rate for the OBW measurement to 99 %.

*CLS

The *CLS common command clears SESR (Standard Event Status Register), the SBR (Status Byte Register) and the Event Queue, which are used in the instrument status and event reporting system. For more details, refer to Section 3 *Status and Events*.

Group COMMON

Related Commands None

Syntax *CLS

Examples *CLS
clears the SESR, the SBR, and the Event Queue.

DISPlay?

The DISPlay? query returns all the settings related to the display features.

Group DISPLAY

Related Commands None

Syntax

```
DISPlay?
```

Arguments None

Responses

```
[DISPLAY:][MENU:NAME]{NONE|SETU|UTIL|MEAS|DISP|SAV|REST|TRIG|
FREQ|SPAN|LEV|MARK};[STATE]{1|0}:[FORMAT]{SING|TWO};
[WINDOW:TYPE]{PROF|SPEC|WAT|ADEM};[CURRENT:WINDOW]{WIND1|WIND2};
[CURRENT:TRACE]{ACTIVE|REF};[TRACE]ACTIVE]{0|1};
[TRACE:REFERENCE]{0|1};[TRACE:AVERAGE]{0|1};
[MARKER:TYPE]{OFF|SING|DUAL};[MARKER:SELECT]{MARK1|MARK2};
[MARKER:Y]<NR3>;[MARKER:Y:UNIT]{DBM|DBV|DBMV|DBUV|DBUVM|W|V|
DEG|PERCENT|HZ|NONE};[MARKER:X]<NR3>;[MARKER:UNIT]{HZ|S|NONE};
[MARKER:PEAK:SEPARATION]<NR1>;[MARKER:FNUMBER]<NR1>;
```

Items in brackets [] are not displayed.

Examples

```
DISPLAY?
```

might return

```
MEAS;1;TWO;SPEC;WIND1;ACTIVE;1;0;0;SING;MARK1;
-1.160666E+02;BM;6.343750E+06;HZ;9;0
```

DISPlay:CURrent:TRAce (?)

When two waveforms (the reference and the one currently being acquired) are on display, the DISPlay:CURrent:TRAce command selects the desired waveform. The DISPlay:CURrent:TRAce? query returns the currently selected waveform.

Group DISPLAY

Related Commands None

Syntax

```
DISPlay:CURrent:TRAce{ACTIVE|REFerence}
DISPlay:CURrent:TRAce?
```

Arguments

ACTIVE	selects the waveform currently being acquired.
REFerence	selects the reference waveform.

Responses {ACTIVE|REF}

Examples

```
:DISPLAY:CURRENT:TRACE REFERENCE
```

selects the reference waveform.

DISPlay:CURrent:WINDow (?)

The DISPlay:CURrent:WINDow command designates a waveform display window as the current window, to which changes to the settings apply. The DISPlay:CURrent:WINDow? query returns the current window.

Group DISPLAY

Related Commands DISPlay:FORMat, DISPlay:WINDow[1|2]:TYPE

Syntax

```
DISPlay:CURrent:WINDow {WINDow1|WINDow2}
DISPlay:CURrent:WINDow?
```

Arguments

WINDow1	selects the lower window in the data display area.
WINDow2	selects the upper window in the data display area.

Responses {WIND1|WIND2}

Examples

```
:DISPLAY:CURRENT:WINDOW WINDOW1
```

sets the lower window to the current window.

DISPlay:FORMat (?)

The DISPlay:FORMat command sets the display format in the data display area. The DISPlay:FORMat? query returns the current display format setting.

Group DISPLAY

Related Commands DISPlay:CURrent:WINDow, DISPlay:WINDow[1|2]:TYPE

Syntax

```
DISPlay:FORMat {SINGle|TWO}
DISPlay:FORMat?
```

Arguments

SINGle	displays the waveform in one display area.
TWO	displays the waveform in two display area.

Examples

```
:DISPLAY:FORMAT TWO
```

sets the instrument to display the waveform in two display area.

DISPlay:MARKer:FNUMber (?)

The DISPlay:MARKer:FNUMber command moves the active cursor to specified frame. The DISPlay:MARKer:FNUMber? query returns the frame number that the active cursor is on.

Group DISPLAY

Related Commands None

Syntax

```
DISPlay:MARKer:FNUMber {MAXimum|MINimum|NTRigger|<NR1>}
DISPlay:MARKer:FNUMber?
```

Arguments

MAXimum	moves the active cursor to the last frame.
MINimum	moves the active cursor to the fist frame.
NTRigger	moves the active cursor to the trigger frame.
<numeric value>::=<NR1>	frame number

Examples

```
:DISPLAY:MARKER:FNUMBER NTRIGGER
```

moves the active cursor to the trigger frame.

DISPlay:MARKer:PEAK

The DISPlay:MARKer:PEAK command moves the active marker to the specified peak in the waveform.

Group DISPLAY

Related Commands None

Syntax DISPlay:MARKer:PEAK {BIGGest|NRIGht|NLEFt}

Arguments

BIGGest	moves the marker to the highest peak.
NRIGht	moves the marker to the nearest peak to the right of the current marker position.
NLEFt	moves the marker to the nearest peak to the left of the current marker position.

Examples

```
:DISPLAY:MARKER:PEAK BIGGEST
```

moves the marker to the highest peak in the waveform.

DISPlay:MARKer:PEAK:SEParation (?)

The DISPlay:MARKer:PEAK:SEParation command sets the interval used by the peak detection feature. The DISPlay:MARKer:PEAK:SEParation? query returns the current setting of the interval.

Group DISPLAY

Related Commands None

Syntax
```
DISPlay:MARKer:PEAK:SEParation <numeric value>
DISPlay:MARKer:PEAK:SEParation?
```

Arguments
<numeric value>::=<NR1>
where <NR1> is the peak detection interval ranging from 1 % to 10 %.

Responses <NR1>

Examples
```
:DISPLAY:MARKER:PEAK:SEPARATION 4
```
sets the peak detection interval to 4 %.

DISPlay:MARKer:SELect (?)

The DISPlay:MARKer:SELect command selects the active marker when two markers (MARKER1 and MARKER2) are displayed on a waveform. The DISPlay:MARKer:SELect? query returns which marker is currently active.

Group DISPLAY

Related Commands DISPlay:MARKer:TYPE

Syntax
```
DISPlay:MARKer:SELect {MARKer1|MARKer2}
DISPlay:MARKer:SELect?
```

Arguments

MARKer1	selects MARKER1 (× marker).
MARKer2	selects MARKER2 (+ marker).

Examples

```
:DISPLAY:MARKER:SELECT MARKER2
```

sets MARKER2 to active.

DISPlay:MARKer:TYPE (?)

The DISPlay:MARKer:TYPE command selects the number of markers to be displayed on the current waveform. The DISPlay:MARKer:TYPE? query returns a string which corresponds to the number of markers displayed on the current waveform.

Group DISPLAY

Related Commands DISPlay:MARKer:SELect

Syntax

```
DISPlay:MARKer:TYPE {OFF|SINGle|DUAL}
DISPlay:MARKer:TYPE?
```

Arguments

OFF	displays no markers.
SINGle	displays the single marker (MARKER1).
DUAL	displays the dual markers (MARKER1 and MARKER2).

Examples

```
:DISPLAY:MARKER:TYPE DUAL
```

displays two markers.

DISPlay:MARKer:X (?)

The DISPlay:MARKer:X command set the horizontal position of the current marker. The DISPlay:MARKer:X? query returns the horizontal position of the current marker.

Group DISPLAY

Related Commands DISPlay:MARKer:Y?

Syntax
```
DISPlay:MARKer:X <numeric value>
DISPlay:MARKer:X?
```

Arguments
```
<numeric value> ::=<NRf>[<unit>]
<unit>={Hz|kHz|MHz|M|GHz|G|s|ms|us|u|ns|n}
```

Examples
```
:DISPLAY:MARKER:X?
```
might return 1.6000000E+07.

DISPlay:MARKer:X:UNIT?

The DISPlay:MARKer:X:UNIT? query the horizontal display unit on the graticule for the current marker.

Group DISPLAY

Related Commands DISPlay MARKer:Y:UNIT?

Syntax
```
DISPlay:MARKer:X:UNIT?
```

Arguments None

Responses
```
{HZ|S|NONE}
```

Examples
```
:DISPLAY:MARKER:X:UNIT?
```
might return HZ.

DISPlay:MARKer:Y?

The DISPlay:MARKer:Y? query returns the vertical position of the current marker.

Group DISPLAY

Related Commands DISPlay:MARKer:X?

Syntax DISPlay:MARKer:Y?

Arguments None

Responses <numeric value>::=<NR3>

Examples :DISPLAY:MARKER:Y?
might return −1.606673E+01

DISPlay:MARKer:Y:UNIT?

The DISPlay:MARKer:Y:UNIT? query the vertical display unit on the graticule for the current marker.

Group DISPLAY

Related Commands DISPlay:MARKer:X:UNIT?

Syntax DISPlay:MARKer:Y:UNIT?

Arguments None

Responses {DBM|DBV|DBMV|DBUV|DBUVM|W|V|DEG|PERCENT|HZ|NONE}

Examples :DISPLAY:MARKER:Y:UNIT?
might return DBM.

DISPlay:MENU[:NAME](?)

The DISPlay:MENU[:NAME] command selects the menu to be displayed on the screen The DISPlay:MENU[:NAME]? query returns the type and the state of the selected menu.

Group DISPLAY

Related Commands DISPlay:MENU:STATe

Syntax
```
DISPlay:MENU[:NAME] {DISPlay|MEASure|SETUp|UTILity|TRIG|
MARKer|SAVE|RESTore|FREQuency|SPAN|LEVel}
DISPlay:MENU[:NAME]?
```

Arguments

DISPlay	displays the display menu
MEASure	displays the measure menu
SETUp	displays the setup menu
UTILity	displays the utility menu
TRIG	displays the trigger menu
MARKer	displays the marker menu
SAVE	displays the save menu
RESTore	displays the restore menu
FREQuency	displays the frequency menu
SPAN	displays the span menu
LEVel	displays the level menu

Responses {DISP|MEAS|SETU|UTIL|TRIG|MARK|SAVE|REST|FREQ|SPAN|LEV},{1|0} where {1|0} indicate display status of the menu.

Examples
```
:DISPLAY:MENU:NAME UTILITY
```
selects the UTILITY menu.

DISPlay:MENU:STATe (?)

The DISPlay:MENU:STATe command sets whether or not menus are displayed on the screen. The DISPlay:MENU:STATe? query returns whether or not menus are displayed on the screen.

Group DISPLAY

Related Commands DISPlay:MENU:[:NAME]

Syntax

```
DISPlay:MENU:STATe <boolean>
DISPlay:MENU:STATe?
```

Arguments

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

`ON or 1`	Menus are displayed.
`OFF or 0`	Menus are not displayed.

Responses

```
<boolean>
```

Examples

```
DISPLAY:MENU:STATE ON
```

sets the instrument to display menus on the screen.

DISPlay:TRACe:ACTIVE(?)

The `DISPlay:TRACe:ACTIVE` command sets whether to display input waveform. The `DISPlay:TRACe:ACTIVE?` query returns the setting of whether to display input waveform.

Group DISPLAY

Related Commands None

Syntax

```
DISPlay:TRACe:ACTIVE <boolean>
DISPlay:TRACe:ACTIVE?
```

Arguments

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

`{ON|1}`	displays the input waveform.
`{OFF|0}`	hides the input waveform.

Responses

```
<boolean>
```

Examples

```
:DISPLAY:TRACE:ACTIVE OFF
```

hides the input waveform.

DISPlay:TRACe:AVERage(?)

The DISPlay:TRACe:AVERage command set whether to display the averaged or peak-hold waveform. The DISPlay:TRACe:AVERage? query returns whether the averaged or peak-held waveform is currently displayed.

Group DISPLAY

Related Commands SENSe:AVERage:TYPE

Syntax
```
DISPlay:TRACe:AVERage <boolean>
DISPlay:TRACe:AVERage?
```

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

{ON|1}	displays averaged or peak-held waveform.
{OFF|0}	displays acquired original waveform.

Responses <boolean>

Examples
```
:DISPLAY:TRACE:AVERAGE ON
```
displays averaged or peak-held waveform.

DISPlay:TRACe:REFerence (?)

The DISPlay:TRACe:REFerence command sets whether to display the reference waveform. The DISPlay:TRACe:REFerence? query returns the setting of whether to display the reference waveform.

Group DISPLAY

Related Commands None

Syntax
```
DISPlay:TRACe:REFerence <boolean>
DISPlay:TRACe:REFerence?
```

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

{ON|1}	displays the reference waveform.
{OFF|0}	hides the reference waveform.

Responses <boolean>

Examples :DISPLAY:TRACE:REFERENCE ON
displays the reference waveform.

DISPlay:WINDow[1|2]:TYPE(?)

The DISPlay:WINDow[1|2]:TYPE command sets what mode is used to display the acquired data. The DISPlay:WINDow[1|2]:TYPE? query returns the current display mode setting.

Group DISPLAY

Related Commands DISPlay:FORMat

Syntax DISPlay:WINDow[1|2]:TYPE {PROFile|SPECtrogram|WATerfall|ADEMod}
DISPlay:WINDow[1|2]:TYPE?

Switch “1” specifies the lower window of the data display area, and switch “2” specifies the upper one.

Arguments

PROFile	display the data in a spectrum view.
SPECtrogram	display the data in a spectrogram view.
WATerfall	display the data in a waterfall view.
ADEMod	displays the data, as an AM, FM, PM, or FSK demodulated signal.

Responses {PROF|SPEC|WAT|ADEM}

Examples :DISPLAY:WINDOW1:TYPE WATERFALL
displays the data in a waterfall view.

*ESE (?)

The *ESE common command sets the bits of the ESER (Event Status Enable Register) used in the status and events reporting system of the GPIB. The *ESE? query returns the contents of the ESER. Refer to Section 3 *Status and Events* for more information about the ESER.

Group COMMON

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

Syntax
```
*ESE <numeric value>
*ESE?
```

Arguments
<numeric value>::=<NR1>
where <NR1> is a decimal integer ranging from 0 to 255. The ESER bits will be set to the binary equivalent of the decimal integer sent.

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?

The *ESR? common query returns the contents of SESR (Standard Event Status Register) used in the status and events reporting system of the GPIB. Refer to Section 3 *Status and Events* for more information about *ESR?.

Group COMMON

Related Commands *CLS, *ESE?, *SRE, *STB?

Syntax
```
*ESR?
```

Arguments None

Examples

```
*ESR?
```

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

HCOPy?

The HCOPy? query returns the set image data format and output port for hardcopy output.

Group HARDCOPY

Related Commands HCOPy:DEVice:LANGuage, HCOPy:DEVice:DESTination

Syntax

```
HCOPy?
```

Arguments None

Responses

```
{BMP|EPS},{FILE|PRIN|OFF}
```

Examples

```
HCOPY?
```

might return BMP,FILE

In this case, the instrument outputs hardcopy data to file on the floppy disk in the BMP format.

HCOPy:DEVice:DESTination (?)

The HCOPy:DEVice:DESTination command sets the hard copy output destination (printer or file). The HCOPy:DEVice:DESTination? query returns the currently specified hard copy destination.

Group HARDCOPY

Related Commands HCOPy?, HCOPY:DEVice:LANGuage

Syntax
```
HCOPy:DEVice:DESTination {OFF|PRINter|FILE}
HCOPy:DEVice:DESTination?
```

HCOPy:DEVice:LANGuage (?)

The HCOPy:DEVice:LANGuage command sets the hard copy output format. The HCOPy:DEVice:LANGuage? query returns the currently specified hard copy output format.

Group HARDCOPY

Related Commands HCOPy?, HCOPy:DEVice:DESTination

Syntax
```
HCOPy:DEVice:LANGuage {BMP|EPS}
HCOPy:DEVice:LANGuage?
```

Arguments

BMP	the standard bit-mapped graphics format.
EPS	the encapsulated Postscript image file format.

Responses {BMP|EPS}

Examples
```
:HCOPY:DEVICE:LANGUAGE BMP
```
sets the instrument to output hard copy in the BMP format.

Arguments

`OFF`	disables hard copy output.
`PRINter`	outputs the data to the printer.
`FILE`	outputs to the data a BMP or EPS file on the floppy disk.

Responses `{OFF|PRIN|FILE}`

Examples

```
:HCOPY:DEVICE:DESTINATION FILE
```

sets the hard copy output destination to a file on the floppy disk.

HCOPy:IMMediate

The `HCOPy:IMMediate` command immediately outputs the hardcopy with the current settings.

Group HARDCOPY

Related Commands None

Syntax `HCOPy:IMMediate`

Arguments None

Examples

```
:HCOPY:IMMEDIATE
```

starts hardcopy output.

*IDN?

The *IDN? common query returns the ID information of the instrument.

Group COMMON

Related Commands None

Syntax *IDN?

Arguments None

Responses <Manufacturer>, <Model>, <Serial Number>, <System Status>
where
<Manufacturer>::=SONY/TEK,
<Model>::=3026,
<Serial Number>::=Jxxxxxx,
<System Status>::=SCPI:<SCPI Version>, FW:<Firmware Version>,
HW:<Hardware Version>.

Examples *IDN?
might return SONY/TEK,3026,J300110,SCPI:94.0 FW:1.00 HW:1.00

INITiate[:IMMediate]

The INITiate[:IMMediate] command restarts data acquisition. It performs the same function as the front panel START key.

Group Other

Related Commands None

Syntax INITiate[:IMMediate]

Arguments None

MMEMory:CDIRectory (?)

The MMEMory:CDIRectory command changes the current working directory. The MMEMory:CDIRectory? query returns the current working directory path.

Group MEMORY

Related Commands MMEMory:MDIRectory

Syntax
```
MMEMory:CDIRectory <Directory Path>
MMEMory:CDIRectory?
```

Arguments
<Directory Path>::=<string>
where <string> is the name of the new current working directory.

Responses <Directory Path>

Examples
```
:MMEMORY:CDIRECTORY "\HARDCOPY\WORK3"
```
changes the current working directory to \HARDCOPY\WORK3.

MMEMory:MDIRectory

The MMEMory:MDIRectory command creates a new subdirectory. The command is invalid if a directory with the specified name already exists.

Group MEMORY

Related Commands MMEMory:CDIRectory, MMEMory:RDIRectory

Syntax MMEMory:MDIRectory <Directory Path>

Arguments <Directory Path>::=<string>
where <string> is the name or path of the new directory.

Examples
```
:MMEMORY:MDIRECTORY "WORK4"
```
creates the new directory WORK4 in the current working directory.

MMEMory:RSETUP

The MMEMory:RSETUP command restores the instrument setup from a disk file or the specified register.

Group MEMORY

Related Commands MMEMory:SSETUP

Syntax MMEMory:RSETUP {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file to be restored.

<D1–D4> the number of the register.

Examples

:MMEMORY:RSETUP D1
restores the instrument setup from the D1 register.

MMEMory:RTMASK

The MMEMory:RTMASK command restores the trigger mask from a disk file or the specified register.

Group MEMORY

Related Commands MMEMory:STMASK

Syntax MMEMory:RTMASK {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file to be restored.

<D1–D4> the number of the register.

Examples

:MMEMORY:RTMASK D1
restores the trigger mask from the D1 register.

MMEMory:RWAVe

The MMEMory:RWAVe command restores the waveform data from a disk file or the specified register.

Group MEMORY

Related Commands MMEMory:SWAVe

Syntax MMEMory:RWAVe {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file to be restored.

<D1–D4> the number of the register.

Examples

:MMEMORY:RWAVE D1

restores the waveform data from the D1 register.

MMEMory:SAWAVe

The MMEMory:SAWAVe command saves the averaged waveform data to a disk file or the specified register.

Group MEMORY

Related Commands None

Syntax MMEMory:SAWAVe {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file that the averaged waveform data is to be saved in.

<D1–D4> the number of the register.

Examples

:MMEMORY:SAWAVE D1

saves the averaged waveform data to the D1 register.

MMEMory:SORT (?)

The MMEMory:SORT command sets the display order for file information in disk directory listings. The MMEMory:SORT? query returns the display order for file information in disk directory listings.

Group MEMORY

Related Commands None

Syntax
```
MMEMory:SORT {NAME1|NAME2|TIME1|TIME2}
MMEMory:SORT?
```

Arguments

NAME1	orders the display according to the ASCII values in the file names (mostly alphabetic order).
NAME2	orders the display in the reverse order of the NAME1 order.
TIME1	orders the display with older (Date and Time) files first.
TIME2	orders the display with more recent (Date and Time) files first.

Responses {NAME1|NAME2|TIME1|TIME2}

Examples
```
:MMEMORY:SORT NAME1
```
sets the order of file information recorded in disk directory listings to alphabetical order by file name.

MMEMory:SSETUP

The MMEMory:SSETUP command saves the instrument setup to a disk file or the specified register.

Group MEMORY

Related Commands MMEMory:RSETUP

Syntax MMEMory:SSETUP {<string>|D1|D2|D3|D4}

Arguments

`<string>`	the name of the file that the instrument setup is to be saved in.
`<D1–D4>`	the number of the register.

Examples

```
:MMEMORY:SSETUP D1
```

saves the instrument setup to the D1 register.

MMEMory:STMASK

The `MMEMory:STMASK` command saves the trigger mask to a disk file or the specified register.

Group MEMORY

Related Commands `MMEMory:RTMASK`

Syntax

```
MMEMory:STMASK {<string>|D1|D2|D3|D4}
```

Arguments

`<string>`	the name of the file that the trigger mask is to be saved in.
`<D1–D4>`	the number of the register.

Examples

```
:MMEMORY:STMASK D1
```

saves the trigger mask to the D1 register.

MMEMory:STWAVe

The MMEMory:STWAVe command saves the waveform data to a disk file or the specified register as a text format.

Group MEMORY

Related Commands MMEMory:SWAVe

Syntax MMEMory:STWAVe {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file the waveform data is to be saved in.

<D1–D4> the number of the register.

Examples

:MMEMORY:STWAVE D1
saves the waveform data to the D1 register as a text format.

MMEMory:SWAVe

The MMEMory:SWAVe command saves the waveform data to a disk file or the specified register.

Group MEMORY

Related Commands MMEMory:RWAVe

Syntax MMEMory:SWAVe {<string>|D1|D2|D3|D4}

Arguments

<string> the name of the file the waveform data is saved.

<D1–D4> the number of the register.

Examples

:MMEMORY:SWAVE D1
saves the waveform data to the D1 register.

*RCL

The *RCL command recalls the instrument settings from the specified register.

Group COMMON

Related Commands None

Syntax *RCL{1|2|3|4}

Arguments 1 to 4 register number

Examples *RCL 1
recalls the instrument settings from the register 1.

RUNNing?

The RUNNing? query returns whether the instrument is either acquiring data or waiting for a trigger.

Group Other

Related Commands STARt, STOP

Syntax RUNNing?

Arguments None

Responses

1 The instrument is either acquiring waveform data or waiting for a trigger.

0 The instrument is not acquiring waveform data.

Examples :RUNNING?
might return 0.

This indicates that the instrument is not acquiring waveform data.

*SAV

The *SAV command saves the instrument settings to the specified register.

Group COMMON

Related Commands None

Syntax *SAV {1|2|3|4}

Arguments 1 to 4 register number

Examples *SAV 1
saves the instrument settings to the register 1.

SENSe?

The SENSe? query returns data acquisition settings.

Group SENSE

Related Commands None

Syntax SENSe?

Arguments None

Responses [:SENSE:RF]{1|0};[FREQUENCY:CENTER]<NRf>;[SPAN]<NRf>;
[LEVEL]<NR3>;[LEVEL:UNIT]{DBM|DBV|DBMV|DBUV|V|W};
[ADEMOD]{WIND1|WIND2},{AM|PM|FM|FSK};[WINDOW:TYPE]
{REC|BLACK|HAMM};[FFT:SIZE]{FFT1024|FFT256};
[FRAME:PERIOD]<NR3>;[BLOCK SIZE]<NR1>;[AVERAGE:TYPE]
{OFF|RMS|EXPONENTIAL|PEAK};[AVERAGE:COUNT]<NR1>;
[GAIN:EXTERNAL:GAIN]<NR2>;[GAIN:EXTERNAL:STATE]{1|0};
[ACQUISITION:MODE]{ROLL|BLOCK}

Items in brackets [ ] are not displayed.

Examples

```
:SENSE?
```

might return

```
1;900000000;2.000000E+6;-1.000000E+01;DBM;WIND1,AM;
BLACK;FFT1024;1.6000000E-4;100;PEAK;16;0.000000;0;ROLL
```

SENSe:ACQuisition[:MODE](?)

The SENSe:ACQuisition[:MODE] command sets the data acquisition mode to Roll or Block. The SENSe:ACQuisition[:MODE]? query returns the current setting of the data acquisition mode.

Group SENSE

Related Commands None

Syntax

```
SENSe:ACQuisition[:MODE]{ROLL|BLOCK}
SENSe:ACQuisition[:MODE]?
```

Arguments

ROLL	sets the data acquisition mode to Roll.
BLOCK	sets the data acquisition mode to Block.

Responses {ROLL|BLOCK}

Examples

```
:SENSE:ACQUISITION:MODE BLOCK
```

sets the data acquisition mode to Block.

SENSe:ADEMod(?)

The SENSe:ADEMod command sets the type of demodulation that is applied to the signal when the analog demodulation display is used. TheSENSe:ADEMod? query returns the type of this signal.

Group SENSE

Related Commands None

Syntax
```
SENSe:ADEMod {WINDow1|WINDow2},{AM|PM|FM|FSK}
SENSe:ADEMod?
```

Arguments

WINDow1	selects the lower window.
WINDow2	selects the upper window.
AM	demodulates an AM signal.
PM	demodulates a PM signal.
FM	demodulate an FM signal.
FSK	demodulates an FSK signal.

Responses {WIND1|WIND2},{AM|PM|FM|FSK}

Examples
```
:SENSE:ADEMOD WINDOW1,FM
```
sets the type of demodulation in the lower window to FM.

SENSe:AVERage:COUNt (?)

The SENSe:AVERage:COUNt command sets the number of frames used for averaging in average/peak hold mode. The SENSe:AVERage:COUNt? query returns the number of frames.

Group SENSE

Related Commands SENSe:AVERage:TYPE

Syntax
```
SENSe:AVERage:COUNt <numeric value>
SENSe:AVERage:COUNt?
```

Arguments <numeric value>::=<NR1>
where <NR1> is number of times of averaging ranging from 2 to 100.

Responses <NR1>

Examples :SENSE:AVERAGE:COUNT 16
sets the number of averaging to 16.

SENSe:AVERage:RESET

The SENSe:AVERage:RESET command resets the average/peak hold data.

Group SENSE

Related Commands SENSe:AVERage:COUNt, SENSe:AVERage:TYPE

Syntax SENSe:AVERage:RESET

Arguments None

SENSe:AVERage:TYPE (?)

The SENSe:AVERage:TYPE command sets the average/peak hold mode. The SENSe:AVERage:TYPE? query returns the current setting of the average/peak hold mode.

Group SENSE

Related Commands SENSe:AVERage:COUNt

Syntax
```
SENSe:AVERage:TYPE {OFF|RMS|EXPONENTIAL|PEAK}
SENSe:AVERage:TYPE?
```

Arguments

OFF	disables all the average/peak hold functions.
RMS	selects RMS averaging.
EXPONENTIAL	selects averaging with an exponential delay function.
PEAK	selects peak–hold.

Responses {OFF|RMA|EXPONENTIAL|PEAK}

Examples
```
:SENSE:AVERAGE:TYPE RMS
```
sets the average/peak hold mode to RMS.

SENSe:BLOCK[:SIZE] (?)

The SENSe:BLOCK[:SIZE] command sets the block size of the data that is acquired in block mode. The SENSe:BLOCK[:SIZE]? query returns the current block size setting.

Group SENSE

Related Commands SENSe:ACQuisition[:MODE]

Syntax
```
SENSe:BLOCK[:SIZE] <numeric value>
SENSe:BLOCK[:SIZE]?
```

Arguments <numeric value>::=<NR1>
where <NR1> is a block size.

The set range is as follows:
If FFT points = 1,024 : 20 to 1,000 (in increments of 20)
If FFT points = 256 : 20 to 4,000 (in increments of 20)

Responses <NR1>

Examples :SENSE:BLOCK:SIZE 500
sets the block size to 500.

SENSe:DATa?

The SENSe:DATa? query returns the data for the logical frame that contains the marker on the current waveform in the current window.

The number of data elements returned varies, as follows, with the hardware settings:

- If the logical frame consists of one physical frame.

  The data for the physical frame is sent as it is. The number of data elements is as follows:

  If FFT points = 1,024 : 800 data elements (for baseband and 5 MHz span), 640 data elements (for other settings)
  If FFT points = 256 : 160 data points

- If the logical frame consists of two or more physical frames:

  The data corresponding to the horizontal range currently being displayed is sent. The number of data elements is 450, which equals the number of pixels contained in the area displaying the waveform. The range of the data to send can be varied by using Horizontal Axis to adjust the horizontal range displayed.

"DEFINE LENGTH ARBITRARY BLOCK RESPONSE DATA", defined in IEEE 488.2, is used to return the information. Single precision floating point data, which consists of 4-byte data elements, returns in Little Endian form.

Group SENSE

Related Commands DISPlay:CURrent:WINDow, DISPlay:CURrent:TRACe, SENSe:LEVel:UNIT

Syntax SENSe:DATa?

Arguments None

Examples :SENSE:DATA?
might return #3644.

Because the GPIB buffer size is restricted, the SENSe:DATa? query must not repeat as follows:

```
DISP:MARK:FNUM 0;:SENS:DAT?;:DISP:MARK:
FNUM 1;:SENS:DAT?<EOI>
```

In this case, insert EOI codes between commands as follows:

```
DISP:MARK:FNUM 0;:SENS:DAT?<EOI>
DISP:MARK:FNUM 1;:SENS:DAT?<EOI>
```

SENSe:FFT[:SIZE] (?)

The SENSe:FFT[:SIZE] command sets the number of FFT sampling points. The SENSe:FFT[:SIZE]? query returns the current setting of the number of FFT sampling points.

Group SENSE

Related Commands None

Syntax

```
SENSe:FFT[:SIZE]{FFT1024|FFT256}
SENSe:FFT[:SIZE]?
```

Arguments

FFT1024	sets 1,024 for the number of sampling points.
FFT256	sets 256 for the number of sampling points.

Responses {FFT1024|FFT256}

Examples

```
:SENSE:FFT:SIZE FFT256
```

sets the FFT sampling points to 256.

SENSe:FRAMe:PERIOD (?)

The SENSe:FRAMe:PERIOD command sets the frame period that is used to acquire data in block mode. The SENSe:FRAMe:PERIOD? query returns the current frame period setting.

The period you may specify varies with the span setting.

Group SENSE

Related Commands SENSe:ACQuisition[:MODE]

Syntax
```
SENSe:FRAMe:PERIOD {MAXimum|MINimum|<numeric value>}
SENSe:FRAMe:PERIOD?
```

Arguments

MAXimum sets the frame period to the maximum.
MINimum sets the frame period to the minimum.

<numeric value>::=<NR3>
where <NR3> is a frame period.

Responses {MAXIMUM|MINIMUM|<NR3>}

Examples
```
:SENSE:FRAME:PERIOD 3.2E-4
```
sets the frame period to 320 μs.

SENSe:FREQuency:CENTer (?)

The SENSe:FREQuency:CENTer sets the center frequency. The SENSe:FREQuency? query returns the current center frequency setting.

Group SENSE

Related Commands None

Syntax
```
SENSe:FREQuency:CENTer {<numeric value>|MAXimum|MINimum}
SENSe:FREQuency:CENTer?
```

Arguments

```
<numeric value>::=<NRf>[<unit>]
<unit>::={Hz|MHz|GHz}
```

MAXimum	sets the center frequency to the maximum.
MINimum	sets the center frequency to the minimum.

Responses

```
<numeric value>
```

Examples

```
:SENSE:FREQUENCY:CENTER 900MHz
```

sets the center frequency to 900 MHz.

SENSe:FREQuency:SPAN (?)

The SENSe:FREQuency:SPAN command sets the frequency span. The SENSe:FREQuency:SPAN? query returns the current frequency span setting.

Group

SENSE

Related Commands

```
SENSe:RF
```

Syntax

```
SENSe:FREQuency:SPAN {<numeric value>|MAXimum|MINimum}
SENSe:FREQuency:SPAN?
```

Arguments

```
<numeric value>::=<NRf>[<unit>]
```

where <unit>::={Hz|MHz|GHz}

MAXimum	sets the frequency span to the maximum.
MINimum	sets the frequency span to the minimum.

Responses

```
<numeric value>
```

Examples

```
:SENSE:FREQUENCY:SPAN 1MHz
```

sets the frequency span to 1 MHz.

SENSe:GAIN:EXTernal:GAIN (?)

The SENSe:GAIN:EXTernal:GAIN command sets the gain correction value that is used when an external device is connected to the instrument. The SENSe:GAIN :EXTernal:GAIN? query returns the current setting of the gain correction value.

Group SENSE

Related Commands SENSe:GAIN:EXTernal:STATe

Syntax
```
SENSe:GAIN:EXTernal:GAIN <numeric value>
SENSe:GAIN:EXTernal:GAIN?
```

Arguments
<numeric value>::=<NR2>
where <NR2> is a gain correction value ranging from –20.0 dB to 90.0 dB in 0.1 dB steps.

Responses <NR2>

Examples
```
:SENSE:GAIN:EXTERNAL*GAIN 10
```
sets the gain correction value to 10 dB.

SENSe:GAIN:EXTernal:STATe (?)

The SENSe:GAIN:EXTernal:STATe command sets whether gain correction is enabled. The SENSe:GAIN:EXTernal:STATe? query returns whether gain correction is currently enabled.

Group SENSE

Related Commands SENSe:GAIN:EXTernal:GAIN

Syntax
```
SENSe:GAIN:EXTernal:STATe <boolean>
SENSe:GAIN:EXTernal:STATe?
```

Arguments
<boolean>::={ON|OFF|1|0}
{ON|1} enables gain correction.
{ON|0} disables gain correction.

Responses {1|0}

Examples :SENSE:GAIN:EXTERNAL:STATE ON
enables gain correction.

SENSe:LEVel (?)

The SENSe:LEVel command sets the reference level. The SENSe:LEVel? query returns the current reference level setting.

Group SENSE

Related Commands SENSe:LEVel:UNIT

Syntax SENSe:LEVel <numeric value>
SENSe:LEVel?

Arguments <numeric value>::=<NR3>
where <NR3> is a reference revel.

The reference level set range varies with the selected input range and the graticule unit of the vertical axis. The value in parentheses () represent the set range that applies when the 50 Hz to 10 MHz input range is used.

dBm	–50.0 dBm to 30.0 dBm (–30.0 dBm to 30.0 dBm)
dBV	–63.0 dBV to 17.0 dBV (–43.0 dBV to 17.0 dBV)
dBmV	–3.0 dBmV to 77.0 dBmV (17.0 dBmV to 77.0 dBmV)
dBμV	57.0 dBμV to 137.0 dBμV (77.0 dBμV to 137.0 dBμV)
V	707.1 μuV to 7.071 V (7.071 mV to 7.071 V)
W	10.00 nW to 1.000 W (1.000 μW to 1.000W)

Responses <NR3>

Examples :SENSE:LEVEL -10
sets the reference level to –10 dBm when the graticule unit of the vertical axis is set to dBm.

SENSe:LEVel:UNIT (?)

The SENSe:LEVel:UNIT command sets the graticule unit of the vertical axis. The SENSe:LEVel:UNIT? query returns the current graticule unit of the vertical axis.

Group SENSE

Related Commands SENSe:LEVel

Syntax
```
SENSe:LEVel:UNIT {DBM|DBV|DBMV|DBUV|V|W}
SENSe:LEVel:UNIT?
```

Arguments

DBM	sets the graticule unit of the vertical axis to dBm.
DBV	sets the graticule unit of the vertical axis to dBV.
DBMV	sets the graticule unit of the vertical axis to dBmV.
DBUV	sets the graticule unit of the vertical axis to dBμV.
V	sets the graticule unit of the vertical axis to V.
W	sets the graticule unit of the vertical axis to W.

Responses {DBM|DBV|DBMV|DBUV|V|W}

Examples
```
:SENSe:LEVel:UNIT DBV
```
sets the graticule unit of the vertical axis to dBV

SENSe:RF (?)

The SENSe:RF command sets the input range to RF or baseband mode. The SENSe:RF? query returns the current input range setting.

Group SENSE

Related Commands None

Syntax
```
SENSe:RF <boolean>
SENSe:RF?
```

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

`{ON|1}`	sets the input range to10 MHz–3 GHz (RF mode).
`{OFF|0}`	sets the input range to 50 Hz–10 MHz (BASEBAND mode).

Responses `{1|0}`

Examples `:SENSE:RF ON`
sets the input range to 10 MHz–3 GHz (RF mode).

SENSe:WINDow[:TYPE] (?)

The `SENSe:WINDow[:TYPE]` command sets the FFT window type. The `SENSe:WINDow[:TYPE]?` query returns the current FFT window type setting.

Group SENSE

Related Commands None

Syntax
```
SENSe:WINDow[:TYPE] {RECTangle|BLACKman|HAMMing}
SENSe:WINDow[:TYPE]?
```

Arguments

`RECTangle`	selects rectangle.
`BLACKman`	selects Blackman-Harris.
`HAMMing`	selects hamming.

Responses `{RECT|BLACK|HAMM}`

Examples `:SENSE:WINDOW:TYPE BLACKMAN`
sets the FFT window type to Blackman-Harris.

SOURce:ROSCillator:SOURce (?)

The SOURce:ROSCillator:SOURce command sets whether the internal clock oscillator or an external clock input signal is used as the clock signal source. The SOURce:ROSCillator:SOURce? query returns whether the internal clock oscillator or an external clock input signal is used as the clock signal source. .

Group SOURCE

Related Commands None

Syntax
```
SOURce:ROSCillator:SOURce {INTernal|EXTernal}
SOURce:ROSCillator:SOURce?
```

Arguments

`INTernal` uses the internal clock source.

`EXTernal` uses the external clock source.

Responses `{INT|EXT}`

Examples
```
:SOURCE:ROSCILLATOR:SOURCE EXTERNAL
```
sets the external clock signal for the reference clock.

*SRE (?)

The *SRE common command sets the bits of the SRER (Service Request Enable Register). The *SRE? common query returns the contents of SRER.

Group COMMON

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

Syntax
```
*SRE <numeric Value>
*SRE?
```

Arguments
```
<numeric Value>::=<NR1>
```
where the argument must be decimal number from 0 to 255. The SRER bits are set in binary bit according to the decimal number.

Examples

```
*SRE 48
```
sets the SRER to 48 (binary 00110000). This sets the ESB and MAV bits.

```
*SRE?
```
might return 32 which indicates that the SRER contains the binary number 00100000.

STATus:OPERation[:EVENt]?

The `STATus:OPERation[:EVENt]?` query returns the contents of the OSR (Operational Status Register).

Executing this command causes the contents of the register to be cleared.

Group STATUS

Related Commands `STATus:QUEStionable[:EVENt]?`

Syntax

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

Arguments None

Examples

```
:STATUS:OPERATION:EVENT?
```
might return 0.

STATus:OPERation:CONDition?

The STATus:OPERation:CONDition query returns the contents of the CR (Condition Register).

Group STATUS

Related Commands STATus:QUEStionable:CONDition?

Syntax STATus:OPERation:CONDition?

Arguments None

Examples :STATUS:OPERATION:CONDITION?
might return 81.

STATus:OPERation:ENABle (?)

The STATus:OPERation:ENABle command set the register that enables the individual bits within the Event Register, which records event transition. The STATus:OPERation:ENABle? query returns the current Event Register settings.

Group STATUS

Related Commands None

Syntax STATus:OPERation:ENABle <numeric value>
STATus:OPERation:ENABle?

Arguments <numeric value>::=<NR1>

Responses <NR1>

STATus:PRESet

The STATus:PRESet command resets the all status enable register.

Group STATUS

Related Commands *CLS

Syntax STATus:PRESet

Arguments None

STATus:QUEue[:NEXT]?

The STATus:QUEue[:NEXT] query returns an error from the error/event queue.

Group STATUS

Related Commands SYSTem:ERRor?

Syntax STATus:QUEue[:NEXT]?

Arguments None

Responses <NR1>, <string>

Examples :STATUS:QUEUE:NEXT?
might return –221,"Settings conflict;–disp?"

STATus:QUEStionable[:EVENt]?

The STATus:QUEStionable[:EVENt] query returns the QSR (Questionable Status Register).

Executing this command causes the contents of the register to be cleared.

Group STATUS

Related Commands STATus:OPERation[:EVENt]?

Syntax STATus:QUEStionable[:EVENt]?

Arguments None

Examples :STATUS:QUESTIONABLE:EVENT?
might return 0.

STATus:QUEStionable:CONDition?

The STATus:QUEStionable:CONDition? query returns the contents of the CR(Condition Register).

Group STATUS

Related Commands STATus:QUEStionable[:EVENt]?

Syntax STATus:QUEStionable:CONDition?

Arguments None

Examples :STATUS:QUESTIONABLE:CONDITION?
might return 81.

STATus:QUEStionable:ENABle (?)

The STATus:QUEStionable:ENABle command sets the register that enables the individual bits within the Event Register. The STATus:QUEStionable:ENABle? query returns the settings.

Group STATUS

Related Commands None

Syntax
```
STATus:QUEStionable:ENABle <numeric value>
STATus:QUEStionable:ENABle?
```

Arguments <numeric value>::=<NR1>

Responses <NR1>

*STB?

The *STB? common query returns the value of the SBR (Status Byte Register). Bit 6 of the SBR is read as a MSS (Master Status Summary) bit. Refer to Section 3 *Status and Events*, for more details on the SBR.

Group COMMON

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

Syntax *STB?

Arguments None

Responses
<numeric value>::=<NR1>
where <NR1> is a decimal integer, which must range from 0 to 255.

Examples
*STB?
might return 96, which indicates that the SBR contains the binary number 01100000.

SYSTem:DATE (?)

The SYSTem:DATE command sets the internal clock date. The SYSTem:DATE? query returns the internal clock date.

Group SYSTEM

Related Commands SYSTem:TIME

Syntax
```
SYSTem:DATE <Year>,<Month>,<Day>
SYSTem:DATE?
```

Arguments

<Year>::=<NR1>	the year
<Month>::=<NR1>	the month
<Day>::=<NR1>	the day

Responses <Year>,<Month>,<Day>

Examples
```
:SYSTEM:DATE 1998,6,10
```
sets the date.

SYSTem:ERRor?

The SYSTem:ERRor? query returns an error message from the error/event queue.

Group SYSTEM

Related Commands STATus:QUEue[:NEXT]?

Syntax SYSTem:ERRor?

Arguments None

Responses <NR1>, <string>

Examples
```
:SYSTEM:ERROR?
```
might return 0,"No error".

SYSTem:FTPD[:STATe] (?)

When this instrument is networked, the SYSTem:FTPD[:STATe] command is used to enable or disable file transfer using FTP (File Transfer Protocol). The SYSTem:FTPD[:STATe]? query returns the current FTP setting.

Group SYSTEM

Related Commands None

Syntax

```
SYSTem:FTPD[:STATe] <boolean>
SYSTem:FTPD[:STATe]?
```

Arguments

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

{ON|1} enables FTP.
{OFF|0} disables FTP.

Responses <boolean>

Examples

```
:SYSTEM:FTPD:STATE ON
```

enables FTP.

SYSTem:TIME (?)

The SYSTem:TIME command sets the internal clock time. The SYSTem:TIME? query returns the internal clock time.

Group SYSTEM

Related Commands SYSTem:DATE

Syntax
```
SYSTem:TIME <Hour>,<Minute>,<Second>
SYSTem:TIME?
```

Arguments

<Hour>	the hours
<Minute>	the minutes
<Second>	the seconds

Responses <Hour>,<Minute>,<Second>

Examples
```
:SYSTEM:TIME 10, 10, 35
```
sets the time.

SYSTem:VERSion?

The SYSTem:VERSion? query returns the SCPI version number complied with the instrument.

Group SYSTEM

Related Commands None

Syntax SYSTem:VERSion?

Arguments None

Responses <NR2>

Examples
```
:SYSTEM:VERSION
```
might return 1994.0.

TRIGger?

The TRIGger? query returns all of the currently specified settings related to the trigger function.

Group TRIGGER

Related Commands TRIGger:COUNt, TRIGger:FREQMASK:CONDition, TRIGger:LEVel,TRIGger:MODE, TRIGger:POSition, TRIGger:SOURce

Syntax TRIGger?

Arguments None

Responses [:TRIGGER:MODE]{NORM|AUTO};[SOURCE]{FREQMASK|TIMELEVEL|EXTIN};[COUNT]<NR1>;[FREQMASK:CONDITION]{BREAK|INSIDE};[LEVEL]<NR2>;[POSITION]<NR1>

Items in brackets [ ] are not displayed.

Examples :TRIGGER?
might return NORM;FREQMASK;1;BREAK;1.000000;50.

TRIGger:COUNt (?)

The TRIGger:COUNt command sets the number of blocks (block count) of data acquired in block mode. The TRIGger:COUNt? query returns the current block count setting.

Group TRIGGER

Related Commands SENSe:BLOCk[:SIZE]

Syntax TRIGger:COUNt {MAXimum|MINimum|<numeric value>}
TRIGger:COUNt?

Arguments

`MAXimum` sets the block count to the maximum.

`MINimum` sets the block count to the minimum.

`<numeric value>::=<NR1>` block count

The maximum block count is determined as follows, depending on the FFT point setting:

For 256 points : 4,000/current block size setting
For 1,024 points : 1,000/current block size setting

Responses

`<numeric value>`

Examples

```
:TRIGGER:COUNT 20
```

sets the block count to 20.

TRIGger:FREQMASK:CONDition (?)

The `TRIGger:FREQMASK:CONDition?` command sets the condition that causes a trigger to occur when the frequency mask is being used. The `TRIGger:FREQMASK:CONDition` query returns the current setting.

Group

TRIGGER

Related Commands

None

Syntax

```
TRIGger:FREQMASK:CONDition {BREAK|INSIDE}
TRIGger:FREQMASK:CONDition?
```

Arguments

`BREAK` causes a trigger when a waveform value outside the trigger mask pattern is encountered.

`INSIDE` causes a trigger when a waveform value inside the trigger mask pattern area is encountered.

Responses

`{BREAK|INSIDE}`

Examples

```
:TRIGGER:FREQMASK:CONDITION BREAK
```

causes a trigger when a waveform value outside the trigger mask pattern area is encountered.

TRIGger:LEVel (?)

If the trigger is to be generated by a signal above a particular amplitude level in the time domain, the TRIGger:LEVel command is used to set the level. The TRIGger:LEVel? query returns the current trigger level setting.

Group TRIGGER

Related Commands TRIGger:SOURce

Syntax
```
TRIGger:LEVel <numeric value>
TRIGger:LEVel?
```

Arguments
<numeric value>::=<NR2>
where <NR2> is a trigger level value ranging from 0.000 V to 8.000 V.

Responses <NR2>

Examples
```
:TRIGGER:LEVEL 0.01
```
sets the level to 0.01 V.

TRIGger:MODE (?)

The TRIGger:MODE command sets the trigger mode. The TRIGger:MODE? query returns the current trigger mode setting.

Group TRIGGER

Related Commands None

Syntax
```
TRIGger:MODE {NORMal|AUTO}
TRIGger:SLOPe?
```

Arguments

NORMal	selects the Normal mode.
AUTO	selects the Auto mode.

Examples
```
:TRIGGER:MODE AUTO
```
sets the trigger mode to Auto.

TRIGger:POSition (?)

The TRIGger:POSition command sets the trigger position. The TRIGger:POSition? query returns the current trigger position setting.

The trigger position is the ratio of acquisition frames before the trigger in a data block.

Group TRIGGER

Related Commands None

Syntax
```
TRIGger:POSition <numeric value>
TRIGger:Position?
```

Arguments
<numeric value>::=<NR1>
where <NR1> is a trigger position ranging from 10 % to 90 % in steps of 1 %.

Responses <NR1>

Examples
```
:TRIGGER:POSITION 50
```
sets the trigger position to 50 %.

TRIGger:SOURce(?)

The TRIGger:SOURce command selects the trigger source for generating the trigger. The TRIGger:SOURce? query returns the current trigger source setting.

Group TRIGGER

Related Commands TRIGger:MODE

Syntax
```
TRIGger:SOURce {FREQMASK|TIMELEVEL|EXT IN}
TRIGger:SOURce?
```

Arguments

FREQMASK	uses the trigger mask pattern to generate the trigger.
TIMELEVEL	uses the signal amplitude level within the time domain to generate the trigger.
EXT IN	uses the signal input through the front panel EXT TRIG connector, to generate the trigger.

Responses {FREQMASK|TIMELEVEL|EXT IN}

Examples :TRIGGER:SOURCE FREQMASK
uses a trigger mask pattern to generate the trigger.

*TST?

The *TST? common query performs the self test sequence. No response is returned.

Group COMMON

Related Commands None

Syntax *TST?

Arguments None

Responses None

Status and Event Reporting

Status and Events

The interface in the 3026 Realtime Spectrum Analyzer includes a status and event reporting system that enables the user to monitor crucial events that occur in the instrument. The realtime spectrum analyzer is equipped with four registers and one queue that conform to IEEE Std 488.2-1987. This section will discuss these registers and queues along with status and event processing.

Registers

There are two main types of registers:

- Status Registers: stores data relating to instrument status. This register is set by the realtime spectrum analyzer.
- Enable Registers: determines whether to set events that occur in the instrument to the appropriate bit in the status registers and event queues. This type of register can be set by the user.

Status Registers

There are five types of status registers:

- Status Byte Register (SBR)
- Standard Event Status Register (SESR)
- Operational Register
- Questionable Register
- Instrument Summary Registers (ISR)

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 3-1 and Table 3-1). These bits are used to monitor the output queue, SESR and service requests, respectively. The contents of this register are returned when the *STB? query is used.

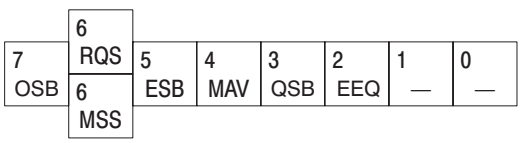


Figure 3-1: The Status Byte Register (SBR)

Table 3-1: SRB bit functions

Bit	Function
7	Summary of the operation status register.
6	RQS (Request Service)/MSS (Master Status Summary). 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 Status Summary (MSS) bit and indicates that the instrument has issued a service request for one or more reasons. The MSS bit is never cleared to 0 by the *STB? query.
5	Event Status Bit (ESB). This bit indicates whether or not a new event has occurred after the previous Standard Event Status Register (SESR) has been cleared or after an event readout has been performed.
4	Message Available Bit (MAV). This bit indicates that a message has been placed in the output queue and can be retrieved.
3	Summary of the Questionable Status Byte register.
2	Summary of the Error Event Queue
1–0	Not used

Standard Event Status Register (SESR)

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

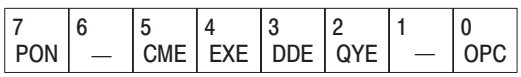


Figure 3-2: The Standard Event Status Register (SESR)

Table 3-2: SESR bit functions

Bit	Function
7	Power On (PON). Indicates that the power to the instrument is on.
6	Not used.
5	Command Error (CME). Indicates that a command error has occurred while parsing by the command parser was in progress.
4	Execution Error (EXE). Indicates that an error occurred during the execution of a command. Execution errors occur for one of the following reasons: ■ When a value designated in the argument is outside the allowable range of the instrument, or is in conflict with the capabilities of the instrument ■ When 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.

Operational Status Register (OSB)

Only bit 13 (OSB) of the Operational Status Register is used by the realtime spectrum analyzer. This bit indicates the status of the Operational Instrument Summary Register (OISR). Use `:STAT:OPER:COND?` to perform a non-destructive query of this register.

Questionable Status Register (QSB)

Only bit 13 (QSB) of the Questionable Status Register is used by the realtime spectrum analyzer. This bit indicates the status of the Questionable Instrument Summary Register (QISR). Use `:STAT:QUEST:COND?` to perform a non-destructive query of this register.

Instrument Summary Registers (OISR and QISR)

Both the Operational Register and Questionable Register receive status from their Instrument Summary Registers (OISR and QISR). The OISR indicates which slots are occupied in the realtime spectrum analyzer. The QISR indicates which modules have reported malfunctions during their power on diagnostics.

Use `:STAT:QUEST:INST:COND?` or `:STAT:OPER:INST:COND?` to perform a non-destructive query of these registers. The query will return an integer value whose bits represent the status of each slot.

***NOTE**. The most significant bit (MSB) of each ISR is always set to "0".*

For example: given the following response to the `:CAT:FULL?` query:

```
"CPU:0","CLOCK:1","BG1:2","AGL1:3","AVG1:7","DVG1:8"
```

The response from the `STAT:OPER:INST:COND?` query will be 399. Converting the decimal integer 399 to binary gives 110001111. Starting at the least significant digit, this value indicates that slots 0, 1, 2, 3, 7, and 8 are occupied.

Enable Registers

There are two types of enable registers: the Event Status Enable Register (ESER) and the Service Request Enable Register (SRER).

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, the user 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 3-3). This register is used by the user to designate whether the SBR ESB bit should be set when an event has occurred and 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.

7	6	5	4	3	2	1	0
PON	—	CME	EXE	DDE	QYE	—	OPC

Figure 3-3: The Event Status Enable Register (ESER)

Service Request Enable Register (SRER)

The SRER is made up of bits defined exactly the same as bits 0 through 7 in the SBR (see Figure 3-4). This register is used by the user to determine what 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.

7	6	5	4	3	2	1	0
OSB	—	ESB	MAV	QSB	—	—	—

Figure 3-4: The Service Request Enable Register (SRER)

Queues

There are two types of queues in the status reporting system used in the realtime spectrum analyzer: output queues and event queues.

Output Queue

The output queue is a FIFO 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 will be 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 will occur and the output queue will be emptied; however, the operation will proceed even if an error occurs.

Event Queue

The event queue is a FIFO queue and stores events as they occur in the instrument. If more than 32 events occur, event 32 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]?

Status and Event Processing Sequence

Figure 3-5 shows an outline of the sequence for status and event processing.

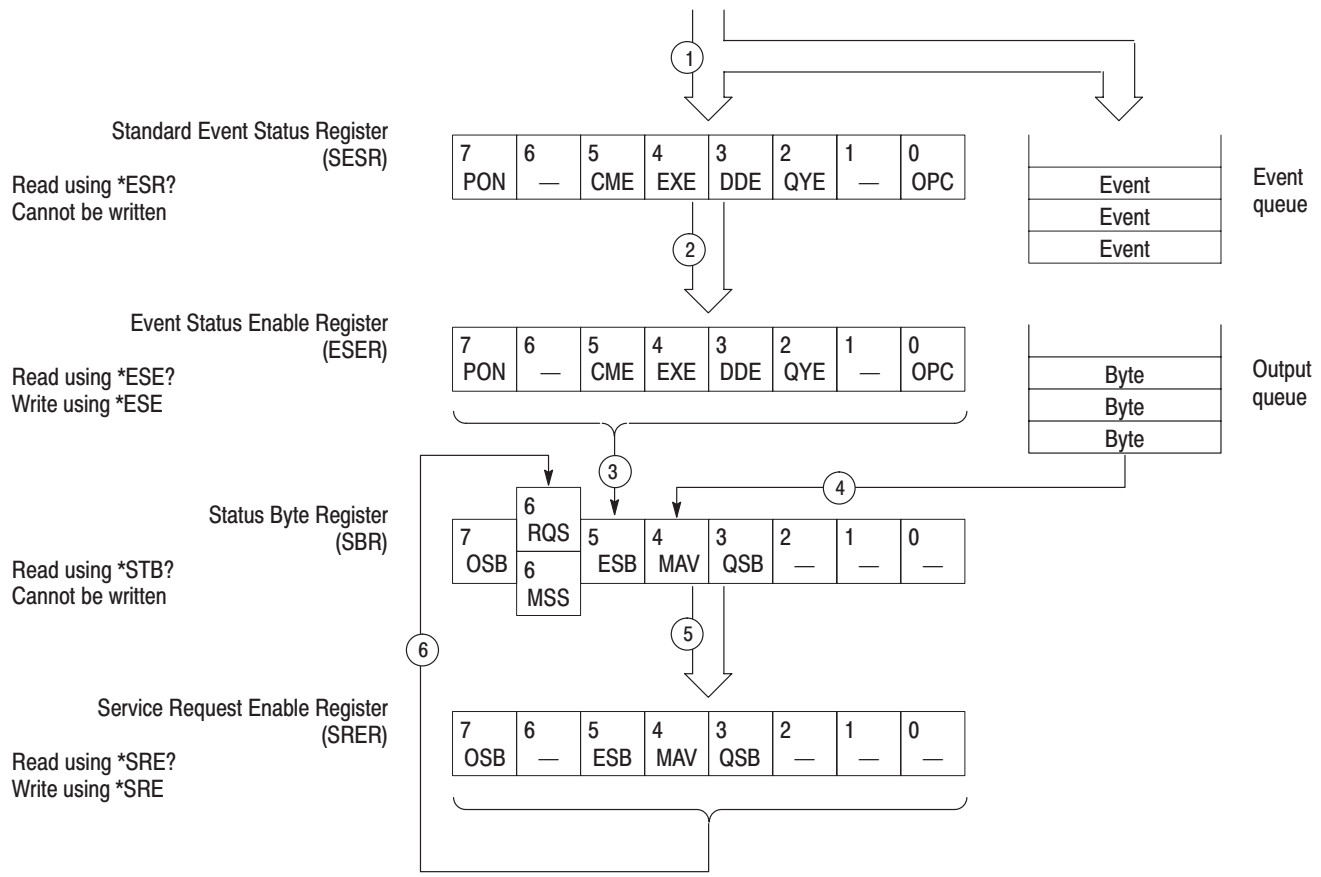


Figure 3-5: Status and event processing sequence

1. If an event has occurred, the SESR bit corresponding to that event is set and the event is placed in the event queue.
2. A bit corresponding to that event in the ESER has is set.
3. The SBR ESB bit is set to reflect the status of the ESER.
4. When a message is sent to the output queue, the SBR MAV bit is set.
5. Setting either the ESB or MAV bits in the SBR sets the respective bit in the SRER.
6. When the SRER bit is set, the SBR MSS bit is set and a service request is generated when using the GPIB interface.

Messages

Tables 3-3 through 3-6 show the codes and messages used in the status and event reporting system in the realtime spectrum analyzer.

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>"
```

Table 3-3 shows the messages generated when there is a syntax error in the command.

Table 3-4 shows the messages generated when an error is detected while a command is being executed.

Table 3-5 shows the messages generated when an internal instrument error is detected. When this type of error occurs, it may be due to a hardware problem.

Error Messages and Codes

Error codes with a negative value are SCPI standard error codes; errors with a positive value are unique to the realtime spectrum analyzer.

Command Errors

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

Table 3-3: Command errors

Error code	Error message
–100	command error
–101	invalid character
–102	syntax error
–103	invalid separator
–104	data type error
–105	GET not allowed
–108	parameter not allowed
–109	missing parameter
–110	command header error
–111	header separator error
–112	program mnemonic too long
–113	undefined header
–114	header suffix out of range
–120	numeric data error
–121	character
–123	exponent too large
–124	too many digits
–128	numeric data not allowed
–130	suffix error
–131	invalid suffix
–134	suffix too long
–138	suffix not allowed
–140	character data error

Table 3-3: Command errors (Cont.)

Error code	Error message
–141	invalid character data
–144	character data too long
–148	character data not allowed
–150	string data error
–151	invalid string data
–158	string data not allowed
–160	block data error
–161	invalid block data
–168	block data not allowed
–170	command expression error
–171	invalid expression
–178	expression data not allowed
–180	macro error
–181	invalid outside macro definition
–183	invalid inside macro definition
–184	macro parameter error

Execution Errors

These error codes are returned when an error is detected while a command is being executed.

Table 3-4: Execution errors

Error code	Error message
–200	execution error
–201	invalid while in local
–202	settings lost due to RTL
–210	trigger error
–211	trigger ignored
–212	arm ignored
–213	init ignored
–214	trigger deadlock

Table 3-4: Execution errors (Cont.)

Error code	Error message
–215	arm deadlock
–220	parameter error
–221	settings conflict
–222	data out of range
–223	too much data
–224	illegal parameter value
–225	out of memory
–226	lists not same length
–230	data corrupt or stale
–231	data questionable
–240	hardware error
–241	hardware missing
–250	mass storage error
–251	missing mass storage
–252	missing media
–253	corrupt media
–254	media full
–255	directory full
–256	FileName not found
–257	FileName error
–258	media protected
–260	execution expression error
–261	math error in expression
–270	execution macro error
–271	macro syntax error
–272	macro execution error
–273	illegal macro label
–274	execution macro parameter error
–275	macro definition too long
–276	macro recursion error
–277	macro redefinition not allowed
–278	macro header not found
–280	program error

Table 3-4: Execution errors (Cont.)

Error code	Error message
–281	cannot create program
–282	illegal program name
–283	illegal variable name
–284	program currently running
–285	program syntax error
–286	program runtime error

Device Specific Errors

These error codes are returned when an internal instrument error is detected. This type of error may indicate a hardware problem.

Table 3-5: Device specific errors

Error code	Error message
–300	device specific error
–310	system error
–311	memory error
–312	PUD memory lost
–313	calibration memory lost
–314	save/recal memory lost
–315	configuration memory lost
–330	self test failed
–350	queue overflow

Query Errors

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

Table 3-6: Query errors

Error code	Error message
–400	query error
–410	query interrupted
–420	query unterminated
–430	query deadlocked
–440	query unterminated after indefinite period

Appendices

Appendix A: Character Chart

Table A–1: ASCII & GPIB Code Chart

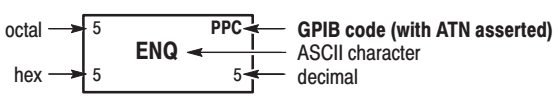
B7 B6 B5 BITS B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	20 DLE 10 16	40 LA0 SP 20 32	60 LA16 0 30 48	100 TA0 @ 40 64	120 TA16 P 50 80	140 SA0 ` 60 96	160 SA16 p 70 112
0 0 0 1	1 GTL SOH 1 1	21 LL0 DC1 11 17	41 LA1 ! 21 33	61 LA17 1 31 49	101 TA1 A 41 65	121 TA17 Q 51 81	141 SA1 a 61 97	161 SA17 q 71 113
0 0 1 0	2 STX 2 2	22 DC2 12 18	42 LA2 " 22 34	62 LA18 2 32 50	102 TA2 B 42 66	122 TA18 R 52 82	142 SA2 b 62 98	162 SA18 r 72 114
0 0 1 1	3 ETX 3 3	23 DC3 13 19	43 LA3 # 23 35	63 LA19 3 33 51	103 TA3 C 43 67	123 TA19 S 53 83	143 SA3 c 63 99	163 SA19 s 73 115
0 1 0 0	4 SDC EOT 4 4	24 DCL DC4 14 20	44 LA4 $ 24 36	64 LA20 4 34 52	104 TA4 D 44 68	124 TA20 T 54 84	144 SA4 d 64 100	164 SA20 t 74 116
0 1 0 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 LA5 % 25 37	65 LA21 5 35 53	105 TA5 E 45 69	125 TA21 U 55 85	145 SA5 e 65 101	165 SA21 u 75 117
0 1 1 0	6 ACK 6 6	26 SYN 16 22	46 LA6 & 26 38	66 LA22 6 36 54	106 TA6 F 46 70	126 TA22 V 56 86	146 SA6 f 66 102	166 SA22 v 76 118
0 1 1 1	7 BEL 7 7	27 ETB 17 23	47 LA7 ' 27 39	67 LA23 7 37 55	107 TA7 G 47 71	127 TA23 W 57 87	147 SA7 g 67 103	167 SA23 w 77 119
1 0 0 0	10 GET BS 8 8	30 SPE CAN 18 24	50 LA8 (28 40	70 LA24 8 38 56	110 TA8 H 48 72	130 TA24 X 58 88	150 SA8 h 68 104	170 SA24 x 78 120
1 0 0 1	11 TCT HT 9 9	31 SPD EM 19 25	51 LA9) 29 41	71 LA25 9 39 57	111 TA9 I 49 73	131 TA25 Y 59 89	151 SA9 i 69 105	171 SA25 y 79 121
1 0 1 0	12 LF A 10	32 SUB 1A 26	52 LA10 * 2A 42	72 LA26 : 3A 58	112 TA10 J 4A 74	132 TA26 Z 5A 90	152 SA10 j 6A 106	172 SA26 z 7A 122
1 0 1 1	13 VT B 11	33 ESC 1B 27	53 LA11 + 2B 43	73 LA27 ; 3B 59	113 TA11 K 4B 75	133 TA27 [5B 91	153 SA11 k 6B 107	173 SA27 { 7B 123

Table continued on next page.

Table A–1: ASCII & GPIB Code Chart (Cont.)

B7 B6 B5 BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
1 1 0 0	14 FF 12 C	34 FS 28 1C	54 LA12 , 44 2C	74 LA28 < 60 3C	114 TA12 L 76 4C	134 TA28 \ 92 5C	154 SA12 l 108 6C	174 SA28 ¦ 124 7C
1 1 0 1	15 CR 13 D	35 GS 29 1D	55 LA13 - 45 2D	75 LA29 = 61 3D	115 TA13 M 77 4D	135 TA29] 93 5D	155 SA13 m 109 6D	175 SA29 } 125 7D
1 1 1 0	16 SO 14 E	36 RS 30 1E	56 LA14 . 46 2E	76 LA30 > 62 3E	116 TA14 N 78 4E	136 TA30 ^ 94 5E	156 SA14 n 110 6E	176 SA30 ~ 126 7E
1 1 1 1	17 SI 15 F	37 US 31 1F	57 LA15 / 47 2F	77 UNL ? 63 3F	117 TA15 O 79 4F	137 UNT _ 95 5F	157 SA15 o 111 6F	177 RUBOUT (DEL) 127 7F
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY



Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: Reserved Words

The words in the following list are reserved words for use with the 3026 Realtime Spectrum Analyzer.

*CAL
*CLS
*ESE
*ESR
*IDN
*RCL
*SAV
*SRE
*STB
*TST
ABORt
ACP
ACQuisition
ACTIVE
AVERage
ADEMod
BANDwidth
BLOCK
CALCulate
CENTer
CDIRectory
CONDition
COUNt
CURrent
DATa
DATE
DEVice
DESTination
DISPlay
ENABle
EVENt
ERRor
EXTernal
FFT
FORMat
FNUMber
FRAMe
FREQMASK
FREQuency
FTPD
FUNCtion
GAIN
HCOPy
IMMediate
INITiate
LEVel
LANGuage
MARKer
MDIRectory
MENU
MMEMory
MODE
NAME
OBW
OPERation
PEAK
PERIOD
POSition
PRESet
QUEStionable
QUEue
RATE
REFerence
RESET
RESult
RF
ROSCillator
RSETUP
RTMASK
RUNNing
RWAVe
SAWAVe
SELect
SENSe
SEParation
SIZE
SORT
SOURce
SPACing
SPAN
SSETUP
STATe
STATus
STMASK
STWAVe
SWAVe
SYSTem
TIME
TRACe
TYPE
TRIGger
UNIT
VERSion
WINDow
X
Y

Appendix C: Interface Specification

This appendix lists and describes the GPIB functions and messages that the 3026 Realtime Spectrum Analyzer 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 C–1: GPIB interface function implementation

Interface function	Implemented subset	Capability
Acceptor Handshake (AH)	AH1	Complete
Source Handshake (SH)	SH1	Complete
Listener (L)	L4	Basic Listener Unaddress if my talk address (MTA) No talk only mode
Talker (T)	T5	Basic Talker, Serial Poll Unaddress if my-listen-address (MLA)
Device Clear (DC)	DC1	Complete
Remote/Local (RL)	RL1	Complete
Service Request (SR)	SR1	Complete
Parallel Poll (PP)	PP0	None
Device Trigger (DT)	DT1	Complete
Controller (C)	C0	None
Electrical Interface	E2	Three-state driver

- Acceptor Handshake (AH). Allows a listening device to help coordinate the 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 transfer of data bytes.

- 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.
- 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.
- Device Clear (DC). Allows a device to be cleared or initialized, either individually or as part of a group of devices.
- 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 data generator.
- Service Request (SR). Allows a device to request service from the controller.
- Controller (C). Allows a device with the capability to send the device address, universal commands, and addressed commands to other device over the interface to do so.
- Electrical Interface (E) Identifies the type of the 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 lists the GPIB Universal and Addressed commands that the 3026 Realtime Spectrum Analyzer implements. A brief description of each function follows the table.

Table C–2: GPIB interface messages

Interface message	Implemented
Device Clear (DC)	Yes
Local Lockout (LLO)	Yes
Serial Poll Disable (SPD)	Yes
Serial Poll Enable (SPE)	Yes
Parallel Poll Unconfigure (PPU)	No
Go To Local (GTL)	Yes
Selected Device Clear (SDC)	Yes
Group Execute Trigger (GET)	Yes

Table C–2: GPIB interface messages (Cont.)

Interface message	Implemented
Take Control (TCT)	No
Parallel Poll Configure (PPC)	No

- Device Clear (DCL). Clears (initializes) all devices on the bus that have a device clear function, whether the controller has addressed them or not.
- Local Lockout (LLO). Disables the return to local function.
- 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 the 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.
- Serial Poll Disable (SPD). Changes all devices on the bus from the serial poll state to the normal operating state.
- Go To Local (GTL). Causes the listen-addressed device to switch from remote to local (front-panel) control.
- Select Device Clear (SDC). Clears or initializes all listen-addressed devices.
- Group Execute Trigger (GET). Triggers all applicable devices and causes them to initiate their programmed actions.
- Take Control (TCT). Allows controller in charge to pass control of the bus to another controller on the bus.
- 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.

Appendix D: Factory Initialization Settings

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

Table D–1: Factory initialized settings

Header	Default settings
CALCULATE commands	
CALCulate:ACP:BANDwidth	5.000000E+05
CALCulate:ACP:SPACing	2.000000E+05
CALCulate:FUNCTION	OFF
CALCulate:OBW:RATE	99.000000
DISPLAY commands	
DISPlay:CURrent:WINDow	WINDOW1
DISPlay:FORMat	SINGLE
DISPlay:MARKer:PEAK	BIGGEST
DISPlay:MARKer:SELect	MARKER1
DISPlay:MARKer:TYPE	OFF
DISPlay:TRACe:ACTIVE	ON
DISPlay:TRACe:AVERage	OFF
DISPlay:TRACe:REFerence	OFF
DISPlay:WINDOW[1|2}:TYPE	PROFILE
HARDCOPY commands	
HCOPy:DEVice:DESTination	OFF
HCOPy:DEVice:LANGuage	EPS
MEMORY commands	
MMEMory:SORT	NAME1
SENSE commands	
SENSe:ACQuisition	ROLL
SENSe:ADEMod	AM
SENSe:AVERage:COUNt	10
SENSe:AVERage:TYPE	EXPONENTIAL
SENSe:BLOCK:SIZE	20
SENSe:FFT	FFT1024

Table D–1: Factory initialized settings (Cont.)

Header	Default settings
SENSe:FRAME:PERIOD	MAXIMUM
SENSe:FREQuency:CENTer	100000000
SENSe:EXTernal:GAIN	0.0
SENSe:LEVel	0.0
SENSe:LEVel:UNIT	DBM
SENSe:RF	ON
SENSe:SPAN	2.000000E+06
SENSe:WINDow	BLACKman
SOURCE commands	
SOURce:ROSCillator:SOURce	INTERNAL
SYSTEM commands	
SYSTem:FTPD	OFF
TRIGGER commands	
TRIGger:COUNt	1
TRIGger:FREQMASK:CONDition	BREAK
TRIGger:LEVel	1.000000
TRIGger:MODE	NORM
TRIGger:POSition	50
TRIGger:SOURce	FREQMASK
COMMON commands	
*ESE	0
*SRE	0

Index

Index

A

B

C

D

S

T

W