TEK DSA 600 SERIES

Part No. 070-8182-00
Product Group 47

THE DSA 601A & DSA 602A DIGITIZING SIGNAL ANALYZERS

Programmer Reference

Please check for CHANGE INFORMATION at the rear of this manual.

First Printing FEB 1991

Tektronix
COMMITTED TO EXCELLENCE

Contents

Setting Up the Instrument

This section describes the implementation of each interface on the DSA 601A and DSA 602A, shows how to connect your DSA to other instruments that have either a GPIB or an RS-232-C interface, and explains how to set up the DSA's front panel for remote operation.

Connecting the DSA to a GPIB Network

Before connecting devices to the GPIB, you should be aware of some rules concerning GPIB networks, cables, and connectors.

GPIB Interface Requirements

GPIB networks can be connected in any configuration, subject to the following rules:

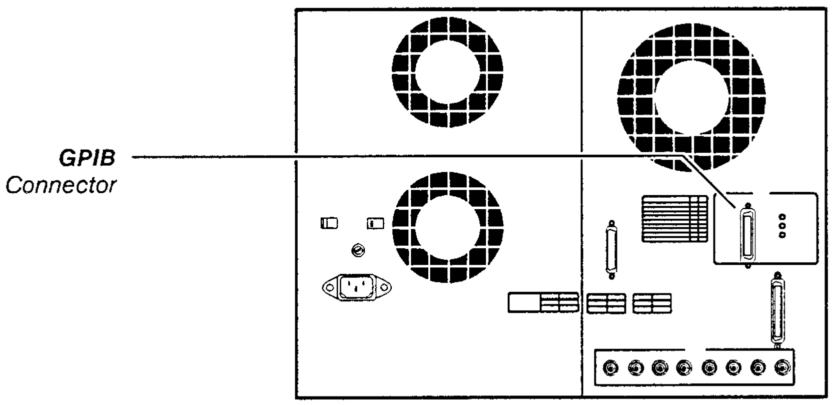
- No more than 15 devices (including the controller) can be included on a single bus.
- In order to maintain bus electrical characteristics, one device load must be connected every two meters (six feet) of cable length. Generally, each instrument represents one device load on the bus.
- The total cumulative cable length must not exceed 20 meters.
- At least two-thirds of the device loads must be powered on when the network is in operation.
- There must be only one cable path from each device to every other device on the network; loop configurations are not allowed.

Cables

An IEEE STD 488 GPIB cable (available from Tektronix) is required to connect two GPIB devices.

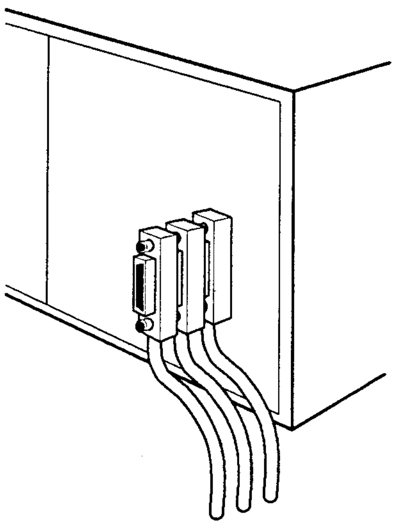
Connector

A 24-pin GPIB connector is located on the rear panel of the oscilloscope. The connector has a D-type shell and conforms to IEEE STD 488.



*Location of **GPIB** Connector on Rear Panel*

GPIB connectors can be stacked upon one another. See the illustration below.



*How **GPIB** Connectors can be Stacked Together*

Setting Up GPIB Parameters

The following steps tell how to set up the GPIB parameters at the front panel.

☐ Step 1: Press the **UTILITY** major menu button to the right of the display area. The Utility major menu appears in the major menu area at the bottom of the display.

☐ Step 2: Touch the **Page to Utility 2** selector on the menu, and then touch the **GPIB Parameters** selector. The **GPIB Parameters** pop-up menu appears in the display area (see the diagram on the next page), with the following selectors:

- **Mode** sets the DSA to be a talker/listener, talker only, or off the bus. Set the DSA to be a talker-only if it is connected to a listen-only device, such as a printer or plotter. Otherwise, set it to be a talker/listener.

The settings for the address and terminator parameters *must* match those of your controller. See the operating manual for your controller to select the appropriate parameters for its GPIB interface.

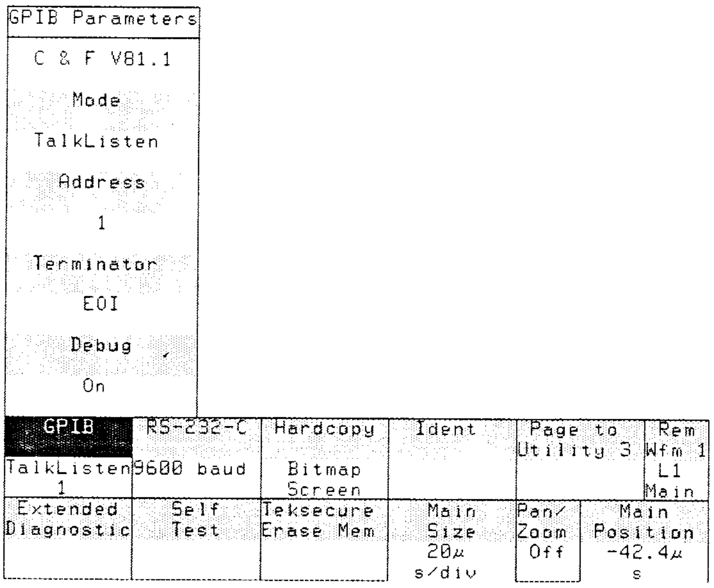
- **Address** sets the primary communication address of the DSA. The address range is 0 to 30.
- **Terminator** sets the method of indicating the end of device messages sent between the controller and the DSA. The choices are EOI (assert EOI line with transmission of last byte of message) or EOI/LF (send line feed character and assert EOI line with its transmission).

When debug is on, input/output processing is slowed.

- **Debug** specifies whether or not GPIB device-dependent messages (DSA commands) appear at the top of the DSA display.

☐ Step 3: Repeatedly touch the selector for each parameter (except **Address**) until the desired value appears.

☐ Step 4: Touch the **Address** selector to assign the knobs to address selection. Rotate either knob to change the address.



GPIB Parameters

C & F V81.1

Mode

TalkListen

Address

1

Terminator

EOI

Debug

On

GPIB TalkListen 1	RS-232-C 9600 baud	Hardcopy Bitmap Screen	Ident	Page to Utility 3	Rem Wfm 1 L1 Main
Extended Diagnostic	Self Test	Teksecure Erase Mem	Main Size 20µ s/div	Pan/ Zoom Off	Main Position -42.4µ s

Typical GPIB Settings on the ***GPIB Parameters*** *Pop-Up Menu*

After these parameters are set, the GPIB interface is ready to operate.

For more information, refer to the explanation of the **GPIB Parameters** pop-up menu in the *DSA 601A and DSA 602A User Reference*.

Connecting the DSA to an RS-232-C Device

The RS-232-C interface provides a point-to-point connection between two items of equipment, such as a computer or terminal and the DSA. The remainder of this section tells how to connect and set up the DSA for communication over the RS-232-C interface.

RS-232-C Interface Requirements

The RS-232-C standard defines two types of devices: Data Terminal Equipment (DTE) and Data Communications Equipment (DCE).

The DSA is configured as a DCE device. A 25-pin female D-type-shell RS-232-C connector is located on its rear panel. In industry-standard usage, a 25-pin male D-connector appears on DTE devices, and a 25-pin female D-connector appears on DCE devices. A straight-through male-to-female cable (at least 9-wire) of less than 50 feet is typically used for local DTE-to-DCE connection.

Note, however, that some DTE devices may have female connectors. Also, the RS-232-C ports of many personal computers are configured as DCE devices, with either a 25-pin or a 9-pin connector. Refer to the documentation that accompanies your computer or terminal to determine if it is a DTE or a DCE device.

The following table shows how the pins map when connecting the DSA to another device in any of three common configurations:

- The DSA to a 25-pin DTE device (most terminals).
- The DSA to a 25-pin DCE device (for example, an IBM PC or compatible with a 25-pin COM port).
- The DSA to a 9-pin DCE device (for example, an IBM PC or compatible with a 9-pin COM port).

In most cases, this pin-mapping information will allow you to connect the devices in these configurations.

RS-232-C Pin Mappings

DSA	25-Pin DTE	25-Pin DCE	9-Pin DCE
1	1	1	NC
2	2	3	3
3	3	2	2
4	4	8	7
5	5	20	8
6	6	6	6
7	7	7	5
8	8	4	1
20	20	5	4

For more complicated cases (such as when working with non-standard devices or cables), the pin-out information in the table below should allow you to wire an appropriate connector. The following suggestions may help:

- Pay special attention to the input signal requirements of the external device (many devices require a constant high signal at one or more input pins).
- For DCE-to-DCE connections, do not connect the output line of one DCE to the output line of the other. *Disregarding this restriction may damage one or both devices.*
- Ensure that the signal ground of the DSA is connected to the signal ground of the external device.
- Ensure that the chassis ground of the DSA is connected to the chassis ground of the external device.

DSA RS-232-C Pin-out

Pin Number	Function	Mnemonic	Direction†
1	Chassis Ground	-none-	
2	Transmit Data	TxD	Input
3	Received Data	RxD	Output
4	Request to Send	RTS	Input
5	Clear to Send	CTS	Output
6	Data Set Ready	DSR	Output
7	Signal Ground	-none-	
8	Data Carrier Detect	DCD	Output
20	Data Terminal Ready	DTR	Input

† *Direction is from the perspective of the controller or terminal.*

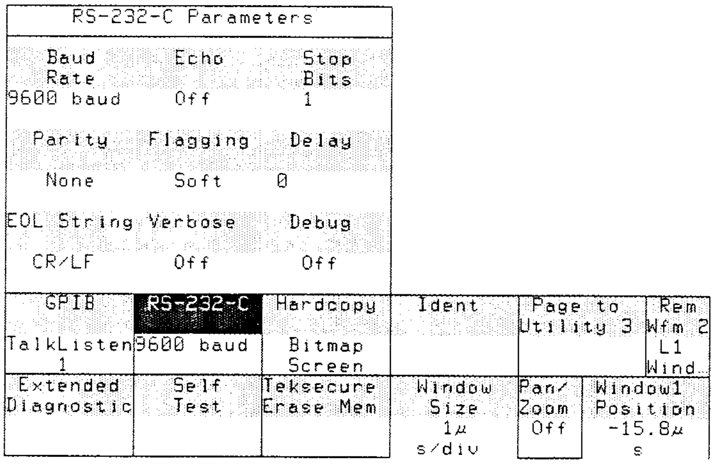
Setting Up RS-232-C Parameters

You can set the parameters of the RS-232-C interface from the front panel (using the Utility major menu and the steps described here), or from within a program (using the RS232 command). After these parameters are set, the RS-232-C interface is ready to operate.

Use the following steps to set up the RS-232-C parameters at the DSA front panel for remote operation.

☐ Step 1: Press the **UTILITY** major menu button. The Utility major menu appears at the bottom of the display.

☐ Step 2: Touch the **Page to Utility 2** selector on the menu, then touch the **RS232C** selector. The **RS232C Parameters** pop-up menu is now displayed.



```
RS-232-C Parameters

  Baud         Echo        Stop
  Rate                     Bits
9600 baud      Off         1

  Parity     Flagging      Delay

  None         Soft        0

EOL String   Verbose       Debug

  CR/LF        Off         Off

   GPIB     | RS-232-C  | Hardcopy  | Ident     | Page to   | Rem
            |           |           |           | Utility 3 | Wfm 2
TalkListen  | 9600 baud |  Bitmap   |           |           |  L1
    1       |           |  Screen   |           |           | Wind.
 Extended   |  Self     | Teksecure | Window    | Pan/  | Window1
Diagnostic  |  Test     | Erase Mem |  Size     | Zoom  | Position
            |           |           |   1µ      | Off   |  -15.8µ
            |           |           |  s/div    |       |   s
```

*Typical Settings on the **RS-232-C Parameters** Pop-Up Menu*

☐ Step 3: Repeatedly touch the selector for each parameter, except **Baud Rate** and **Delay**, until the value you want appears. Touching **Baud Rate** or **Delay** activates the knobs to control these parameters. The RS-232-C selectors are:

The baud rate, stop bits, and parity settings must match those of the controller or terminal, or RS-232-C data communication will be impossible. Also, the controller or terminal's RS-232-C port must be set to use 8-bit characters.

- **Baud Rate** sets the data transmission rate. The selections are 110,150, 300, 600, 1200, 2400, 4800, 9600, or 19200 baud.

- **Stop Bits** sets the number of stop bits sent after each character. The selections are 1, 1.5, or 2 bits.

- **Parity** sets the error check bit for each character. The selections are none, even, or odd. When the parity setting is odd or even, the DSA generates the selected parity on output and checks incoming data against the selected parity on input. When the parity setting is none, no input parity error checking is performed and no output parity is generated.

- **Echo** allows characters sent to the DSA to be echoed. When echo is turned on, all characters sent to the RS-232-C port are echoed; when echo is turned off, input characters are not echoed.

  Turn echo off when a computer program is transmitting data to the DSA (for example, when a BASIC program on a small computer is being used to control the DSA via the RS-232-C port). The computer program will not expect to see its commands echoed back and the program will fail. The first command your program sends to the DSA should be "ECHO OFF;VERBOSE OFF;INIT".

  Turn echo on when using a CRT, hardcopy terminal, or computer with a terminal emulation program. Turning echo on in this case allows you to see what you have just typed on your computer or terminal screen.

- **Flagging** sets the method of controlling the flow of data between devices. Flagging is a way for the device receiving data to tell the transmitting device when to stop or resume sending data. The selections are none, hard, or soft. When flagging is set to none, the DSA does not use or recognize any flagging.

  When flagging is set to hard, the DSA uses the DTR (Data Terminal Ready) and CTS (Clear To Send) lines to control data transmission. On output, the DSA transmits data only when the DTR line is asserted. When the DTR line is not asserted, the DSA stops transmitting data. On input, the DSA unasserts the CTS line to stop transmission when its input buffer is three-quarters full, and asserts the CTS line to restart transmission when its input buffer is three-quarters empty.

  Soft flagging is usually not used with binary data transfer, since the data may contain XON and XOFF character equivalents. Use hard flagging for binary data transfer.

  When flagging is set to soft, the DSA stops transmitting data on output when it receives an XOFF (DC3) character, and begins transmitting again when it receives an XON (DC1) character. On input, the DSA sends an XOFF character to halt transmission when its input buffer is three-quarters full, and sends an XON character to resume transmission when its input buffer is three-quarters empty.

- **Delay** sets the minimum delay time for the DSA to respond to a query. The delay range is 0 to 60 seconds, in multiples of 20 milliseconds.

- **Verbose** displays status and event messages as commands are executed. When verbose is turned on, each command sent to the DSA returns a response; for example, successfully executed commands return a response of "OK," successfully executed queries return their query data, and events return a response of "EVENT XXX", where XXX is an event code. When verbose is turned off, the controller must query the DSA to receive the message.

Turn verbose off when a computer program is transmitting data to the DSA (for example, when a BASIC program on a small computer is controlling the DSA with the RS-232-C interface). The first command your program sends the DSA should be "ECHO OFF;VERBOSE OFF;INIT".

Turn verbose on when using a CRT, hardcopy terminal, or computer with a terminal emulation program. Turning verbose on gives you feedback on the execution of commands you have typed.

- **EOL String** sets the end-of-line message terminator for the response to a query. The selections are <CR> (carriage return), <LF> (line feed), <CR><LF> (carriage return followed by line feed), or <LF><CR> (line feed followed by carriage return).

When debug is turned on, input/output processing is slowed.

- **Debug** controls whether or not RS-232-C commands appear at the top of the DSA display as they are executed.

Command Syntax

This section explains the syntax and command processing conventions of the command set. The command set can be found in the section called Commands.

The command set can control the operations and functions of the DSA from an external interface (GPIB or RS-232-C). The same command syntax is used for both interfaces.

Contact your Tektronix field representative for information on the Tek Codes and Formats Standard.

The GPIB and RS-232-C command messages conform to the Tektronix Codes, Formats, Conventions, and Features Standard, or "Tek Codes and Formats" for short. It defines the format of program elements and statements for the command language.

You transmit commands to the DSA using an enhanced American Standard Code for Information Interchange (ASCII) character encoding. The DSA supports both the standard ASCII character set and an additional "escape" character set that includes graphic elements. The character sets are described in Appendix C of this manual.

Command Structure

Command language messages are composed of set and query commands. Query commands are simply called *queries*. Set commands tell the DSA to take a specific action. Queries ask the DSA to return information about its state.

Syntax Conventions

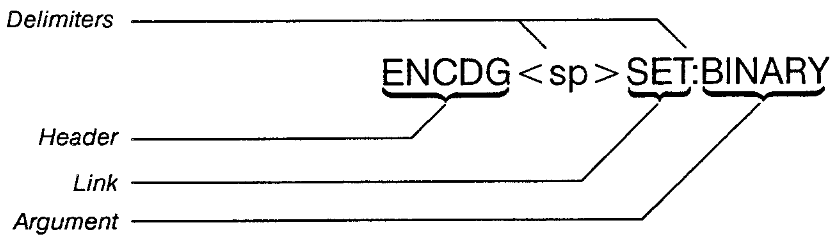
The following Backus-Naur Form (BNF) syntax conventions are used throughout this manual:

< >	Defined data type.
::=	Is defined as.
|	Exclusive OR.
{ }	One of a group is required.
[ ]	Optional item.
. . .	Previous elements may be repeated.

Commands are composed of four syntactic elements:

`<header>`	::=	The command name; if it ends with a question mark, the command is a query.
`<delimiter>`	::=	A space (<sp>), colon (:), comma (,), or semicolon (;) which breaks the message into segments for the DSA to process.
`<link>`	::=	A command sub-function. Not all commands have links.
`<argument>`	::=	A quantity, quality, restriction, or limit associated with the header or link.

The following illustration shows the four syntactic elements:



Example of Syntactic Elements

You can use most commands to set or query. However, some commands and links can only be used to set, while others can only be used to query. Attempting to query a set-only command or a set-only link always results in a syntax error.

The following is a list of symbols used in this manual:

<CR>	Carriage return (ASCII decimal 13).
<EOI>	End or Identify. IEEE std 488 specified message terminator signal.
<LF>	Line feed (ASCII decimal 10).
<ui>	Unsigned integer, range is 1 through 65,535.
<NR1>	Signed integer value.
<NR2>	Floating point value, without an exponent.
<NR3>	Floating point value, with an exponent.
<NRx>	{<NR1> | <NR2> | <NR3>}. Range is: −1E±300, 0, 1E±300, to 15 significant digits.
<asc bin>	ASCII-formatted pixel bin count data.
<asc curve>	ASCII-formatted waveform data.
<asc data>	ASCII-formatted histogram data.
<bblock>	Binary-block formatted waveform data.
<meas>	One or more of the DSA measurement commands. Measurements apply to the selected waveform unless a specific waveform is designated.
<id>	{A|...|Z}
<sp>	A space.
<qstring>	Quoted string data.
<slot>	L, C, or R, representing the Left, Center, or Right plug-in compartments.

Set Commands

Set commands cause the DSA to perform a function, or change a setting or mode. There are four basic types of set commands:

```
<header>
<header> <sp> <arg>[,<arg>]
<header> <sp> <link>:<arg>[{,<link>:<arg>}...]
<header> <sp>{<link>:<arg> | <arg>}
    [{,<link>:<arg> |,<arg>}...]
```

Queries

Queries cause the DSA to return a measurement, waveform (trace) data, or a status condition (for example, a current setting or mode). The DSA puts the response message in its output buffer.

Query commands have two basic structures:

```
<header>?
<header>?<sp> <link>[{,<link>}...]
```

Any response from a query that has a corresponding set command can be sent to the DSA as a valid command.

The response from a query-only command cannot be returned to the DSA as a set command.

However, a query response that includes a mixture of set and query-only commands can be returned to the DSA without generating an error.

Command Processing

The following rules apply when processing commands:

- You can use either upper or lower case.
- All command elements (headers, links, arguments, or punctuation) may be preceded or followed by white space (blank characters).
- Commands consisting solely of any combination of blank characters are called null commands. Null commands are ignored by the DSA and do not produce an error.

The following rules apply to quoted strings:

- You can use a maximum string length of 127 characters, unless otherwise noted.
- You cannot use strings that include an embedded ASCII NULL character (0). However, carriage returns and line feeds can be included as text in a string.
- The same type of quote that opens a quoted string must close that string. Examples:

  ```
  "this is a quoted string" and ´so is this´
  ```

  ```
  ´But this is not a quoted string"
  ```

- You can mix single and double quote marks within a string if the previous rule is followed. For example:

  ```
  "this is an ´acceptable´ string"
  and ´so "is" this´
  ```

- A quote may be included within a string by simply repeating the quote. Examples:

  ```
  "double "" quote"
  ´single ´´ quote´
  ```

  Results in:

  ```
  double " quote
  single ´ quote
  ```

Abbreviating Commands

Each command reserved word (header, link, or argument) that is transmitted to the DSA has an abbreviation. The abbreviated spelling is always shown in uppercase. If a command name contains punctuation, for example, a period (.) or a slash (/), it must be included in the abbreviated spelling. The complete list of reserved words and their abbreviations is in Appendix B.

Responses are returned with the full spelling unless the LONG-FORM command is set to OFF. Examples in this manual use abbreviated command spellings; responses are in long form.

Concatenating Commands

Any combination of commands may be joined with a semicolon. For example:

```
RQS ON;ENCDG?;UPTIME?
```

Responses to concatenated queries are separated by semicolons.

Disk Command Strings

The DSA's floppy disk commands are similar to MS-DOS commands. When using quoted strings to access the disk, it is important to remember what directory you are in. To avoid confusion, it is recommended that you always use the full pathname when referencing files.

Here are some rules to remember when using quoted strings to access the disk:

- "A:\" is the root directory. For example, RMDIR "A:\TEST" removes the directory TEST in the root directory.
- "A:" is the disk, but any directory reference will be relative to the current directory. For example, MKDIR "A:TEST_SUB " is the same as MKDIR "TEST_SUB ", but not necessarily the same as MKDIR "A:\TEST_SUB ".
- Check the status of the SETDEV command if unexpected results occur.

Defining New Command Strings

The DEF command enables you to create new command names. That is, you can rename an existing DSA command function, or you can concatenate several existing commands under a single, new command name.

For example, to create a command that gives you the date and time, you could give the following command:

```
DEF "DATIME?","DATE?;TIME?"
```

Getting Long-form or Short-form Responses from the DSA

Long form is easier to read; short form is more efficient during data transfers.

The LONGform command determines whether the DSA responds to queries in long form or short form. In long form, queries return fully spelled reserved words, and an event query returns both the numeric event code and its associated message string. In short form, queries return abbreviations of reserved words, and event queries return only the numeric event code.

Message Terminators

Message terminators are transmitted by a sending device to let receiving devices know that message transmission is complete. The DSA allows you to select a message terminator that is compatible with the controller or terminal you are using.

Terminators for the RS-232-C interface – are selected through the front panel using the **RS232C Parameters** pop-up menu from the Utility 2 major menu or through the interface using the RS232 command. RS-232-C terminators are: <CR>, <LF>, <CR><LF>, and <LF><CR>.

Line feeds and carriage returns embedded within binary block (<*bblock*>) data are treated as data bytes, not as message terminators. Once the DSA begins reading a binary block, line feeds and carriage returns are not processed as terminators until the byte count of the block is satisfied.

Terminators for the GPIB interface – are selected through the front panel only, using the **GPIB Parameters** pop-up menu from the Utility 2 major menu. GPIB terminators are: <EOI> and <EOI><LF>.

I/O Buffers

The following information pertains to both GPIB and RS-232-C input/output buffers, unless noted otherwise.

I/O buffer sizes – are 256 bytes for the GPIB input buffer, 1024 bytes for the RS-232-C input buffer, and 1024 bytes for the GPIB and RS-232-C output buffers.

Data that exceeds the sizes of the GPIB and RS-232-C input/output buffers (256/1024 bytes) can be accepted. The DSA parses input data as soon as it is received at either port, thereby continuously emptying the input buffers while processing commands.

If an external controller fills an input buffer before the DSA has an opportunity to process the contents, the DSA holds off the external controller (with GPIB interface signals or RS-232-C flagging) until the buffer has been processed.

Likewise, if a query response fills an output buffer, the DSA stops sending data to the buffer until some of the data are read by the external controller or terminal.

When a new message is received at the GPIB port—the DSA unconditionally clears its GPIB output buffer (no error is reported). This means that the GPIB output buffer of the DSA must be read by the controller after each message containing a query is sent, or the response will be lost (overwritten).

When GPIB input and output message buffers are full—the DSA unconditionally clears the GPIB output buffer. An execution error is also reported (event code 203, "I/O buffers full").

If the GPIB buffers are empty—and the DSA is talk-addressed and is not currently processing a GPIB command, it returns a Talked-With-Nothing-To-Say (TWNTS) message to the controller. This message is one byte with all eight bits set, ended by a message terminator (FF*(hex)*<*EOI*>). It is then up to the controller program to take appropriate action.

If a "hang" condition occurs, consult your controller or terminal operator manual for restart instructions.

If the RS-232-C output buffer is empty—and an external device attempts to read data from the RS-232-C port, the external device will "hang" the interface (no further input/output operations will be possible). This condition cannot occur when using a computer or terminal to send commands interactively to the DSA over the RS-232-C interface. This condition may occur when executing a program that expects input from the DSA's RS-232-C port. In such cases, it is up to the program to recognize a "timeout" condition for expected input and take appropriate action.

GPIB Specific Conventions

When the DSA receives a Device Clear (DCL) or Selected Device Clear (SDC) interface message from the GPIB, it does the following:

1. Clears any service requests and all pending events, except power on.
2. Clears the GPIB input and output buffers.
3. Restarts GPIB message processing in the DSA.

DCL and SDC interface messages do not change DSA settings or stored data, and do not interrupt front panel control or non-programmable functions.

RS-232-C Specific Conventions

You should be aware of the processing conventions that are specific to the RS-232-C interface. These conventions pertain to:

- Transferring binary block data.
- Echoing character input.
- Using Verbose mode.
- Processing "break" characters.
- RS-232-C I/O errors.

When transferring binary block data—to the DSA via the RS-232-C port, note the following points:

- Do not transmit binary block data to the DSA when ECHO is set to ON. Attempting to do so causes the input block to be discarded and generates event code 164.
- Do not use binary data transfers with soft flagging unless you can ensure that the data does not contain XON or XOFF characters. Using DTR/CTS (hard) flagging guarantees correct data transfer.
- All eight bits of a binary-block data byte contain meaningful information. To ensure that all eight bits are received or transmitted, an RS-232-C device must be configured to receive and transmit eight-bit characters (set the RS-232-C word length to eight bits).

Echoing character input—means that all characters received at the RS-232-C port are echoed back to the command source when ECHO is set to ON.

You can turn echo on or off from the front panel by selecting **RS232C** from the Utility 2 major menu and touching the **Echo** selector, or you can send one of the commands: RS232 ECHO:ON or RS232 ECHO:OFF.

When you are using a computer program to transmit commands to the DSA (for example, when a BASIC program is being used to control the DSA via the RS-232-C port), ECHO should be set to OFF.

When you are using either a CRT, a hardcopy terminal, or a computer running terminal-emulation software to send commands interactively to the DSA via the RS-232-C port, ECHO should be set to ON.

When ECHO is set to ON, it has the following effects on command input:

- The DSA solicits command input with a "> " prompt. When this prompt appears on an RS-232-C device, enter a valid command and terminator.

- All command input is buffered. Therefore, commands will not be analyzed or executed until a terminator is received at the RS-232-C port.

- Until the command is terminated, it may be edited with any of the following special characters:

  CONTROL-R retypes the current input command and places the cursor to the right of the last character of the command.

  CONTROL-U deletes the current command and returns the cursor to the start of the line.

  BACKSPACE, DEL or RUBOUT erase the character to the left of the cursor (the effect of the backspace character is compatible with CRT terminals, but not with hardcopy terminals). If a character has been erased with the backspace key, the newly edited command can be seen by using the CONTROL-R character (applies to both CRT and hardcopy terminals).

BACKSLASH (\) delimits special editing characters (CR, LF, BACKSPACE, DEL, CONTROL-R, or CONTROL-U) in a quoted string.

- Command input is discarded if it exceeds 256 bytes (the input buffer size) before a terminator is entered. If this happens, a command error (event code 163) is posted to the RS-232-C port and the input buffer is emptied.

- Non-printable ASCII characters are echoed with the visual representations shown in the following table:

Non-Printable ASCII Character Representations

ASCII Character	Echoed Character	ASCII Character	Echoed Character
NUL (0)	^@	DC2 (18)	^R †
SOH (1)	^A	DC3 (19)	^S ††
STX (2)	^B	. .	.
. .	.	NAK (21)	^U †
. .	.	. .	.
BS (8)	^H †	. .	.
HT (9)	^I	. .	.
LF (10)	^J †	SUB (26)	^Z
. .	.	ESC (27)	^[
. .	.	FS (28)	^\
CR (13)	^M †	GS (29)	^]
. .	.	RS (30)	^ ^
. .	.	US (31)	^_
DC1 (17)	^Q ††	DEL (127)	^? †

† *Only echoed when preceded with a backslash.*
†† *Only echoed when soft flagging is disabled.*

Using verbose mode – causes the DSA to return a response for each command sent. When VERBOSE is set to OFF, only valid queries return a response from the DSA.

You can turn verbose on or off from the front panel by selecting **RS232C** from the Utility 2 major menu and touching the **Verbose** selector, or you can send the commands RS232 VERBOSE:ON or RS232 VERBOSE:OFF.

When RS-232-C VERBOSE is set to ON, each semicolon or terminated input command causes the DSA to return one of these responses:

OK	Returned for a successfully executed set command.
<*query response*>	Returned for a successfully executed query.
EVENT <*NR1*>[,<*qstring*>]	Returned when the DSA detects an error while parsing or executing a query/set command, where the <*NR1*> value represents an event code and the optional <*qstring*> is an event code description string that describes the numerical event code. The event code description string <*qstring*> is only returned when LONGFORM is set to ON. If more than one error is detected while parsing a query or set command, only one EVENT response is returned to the RS-232-C port. All other errors are stacked and may be polled with the STBYTE? or EVENT? commands.

The following table demonstrates typical DSA response behavior with VERBOSE mode set to ON.

Examples of Responses with VERBOSE ON

Input Command(s)	DSA Response
LONGFORM OFF	OK
INPUT STO1;RS232? BAUD	OK;RS232 BAUD:9600
JUNK;INIT;INPUT?	EVENT 156;OK;INPUT STO1
JUNK;INIT	EVENT 156;OK

When RS-232-C VERBOSE is set to OFF, only valid queries cause the DSA to return responses to the RS-232-C. Errors associated with invalid commands are not discarded; they are stacked and may be polled at any time by using the STBYTE? or EVENT? commands.

The following table demonstrates typical DSA response behavior with VERBOSE set to OFF.

Examples of Responses with VERBOSE OFF

Input Command(s)	DSA Response
INPUT STO1;RS232? BAUD	RS232 BAUD:9600
JUNK;INIT;INPUT?	INPUT STO1
JUNK;INIT	*(none)*

The factory default state for verbose mode is off.

Verbose mode affects event communication at power-on. When the DSA is turned on and completes its power-on cycle, the DSA communicates events differently depending on the state of the verbose function.

- When VERBOSE is set to ON at power-on, an asynchronous message is written to the RS-232-C port. This message reports either that the instrument is operating satisfactorily (Event 401, "Power on"), or that diagnostics have discovered a fault (Event 394, "Test completed and failed").
- When VERBOSE is set to OFF at power-on, no asynchronous messages are written to the RS-232-C port. Instead, power-on events are stacked in the usual manner.

When the DSA senses a BREAK signal – at the RS-232-C port, it returns a special message that acknowledges this transmission. The form of the acknowledgement message depends on whether ECHO is set to ON or OFF.

- When ECHO is set to ON, the DSA signals that it has processed the BREAK signal by echoing a new prompt symbol "> " for command input.
- When ECHO is set to OFF, the DSA signals that it has processed the BREAK signal by sending the following character string to the RS-232-C device:

  DCL<*terminator*>

Reception of the BREAK signal clears the RS-232-C input and output buffers and restarts the DSA's RS-232-C message processing. BREAK signals do not change DSA settings or stored data, and do not interrupt front panel operation or non-programmable functions.

RS-232-C I/O errors—are reported when there is a problem with parity, framing, or input buffer overruns.

To report RS-232-C errors, the DSA prints an error message on the display and posts an event code to both the GPIB and the RS-232-C ports:

RS-232-C I/O Errors

I/O Error	Event Code	Information
Parity	653	Check to identify transmission errors (PARITY ON)
Framing	654	A stop bit was not detected when data was received at RS-232-C port (indicates baud-rate mismatch)
Input Buffer Overrun	655	Software or hardware input buffer overflowed with data (caused by improper or nonexistent flagging)

To recover from I/O errors, the DSA RS-232-C interface takes the following actions:

During these I/O error recovery steps (when ECHO is set to OFF), the DSA may process incomplete commands, causing spurious syntax or semantic errors to be reported.

- When ECHO is set to OFF, all unparsed input buffer data are discarded until a semicolon or <*terminator*> character is encountered. Command processing resumes or resynchronizes from the point at which the semicolon or <*terminator*> is found.

- When ECHO is set to ON, all buffered, but unparsed, input data are discarded and you are prompted again for input.

Commands

This section presents a complete description of the command set.

Functional Groups

The table below lists the groups and their functions. The following pages show all the command headers grouped by function.

Functional Groups in the Command Set

Group	Functions Controlled
Acquisition	Acquisition (digitizing) of waveforms.
Calibration/Enhanced Accuracy	Enhanced accuracy functions.
Channel/Vertical	Plug-in amplifier vertical parameters.
Cursor	Waveform cursor selection and positioning.
Data Transfer	Transfer of waveform data and front-panel settings to and from the DSA.
Diagnostics	Self-test diagnostics and extended diagnostics.
Display and Color	Front-panel display parameters and colors.
External I/O	Printer parameters, debug functions, and RS-232-C parameters.
Floppy Disk	Floppy disk functions.
Label and Text	Placement of user-defined labels and text.
Measurement	DSA measurement functions.
Miscellaneous/System	System and front-panel functions.
Status and Event	Instrument event reporting, hardware identification, and configuration information.
Time Base/ Horizontal	Main and window record length and position.
Triggering	Triggering parameters.
Waveform and Settings	Waveform creation and modification, and front-panel settings commands.

Acquisition

AUTOACQ	Selects traces and controls memory wrapping for repetitive single trigger acquisition.
AUTOSET	Adjusts the waveform settings for optimal display.
AVG	Turns waveform averaging on or off.
AVGTYPE	Selects backweighted or summation averaging.
CONDACQ	Controls the condition(s) on which the acquisition of waveforms stops.
DELTA	Compares an acquired waveform with an enveloped reference waveform.
DIGITIZER	Starts and stops waveform acquisition.
ENV	Turns waveform enveloping on or off.
FFT	Controls FFT (Fast Fourier Transform) parameters.
FILTER	Limits the digitizer bandwidth for anti-alias filtering.
INCACQ	Controls digitizer incremental acquire mode.
INTERLEAVE	Controls digitizer interleave mode, for maximum DSA sample rates.
NAVG	Sets the number of acquisitions to be used for waveform averaging.
NENV	Sets the number of acquisitions to be used for waveform enveloping.
NREPTRIG	Sets the number of conditional acquisitions, if repetitive trigger is set.

Calibration/Enhanced Accuracy

CALPROBE	Initiates the probe calibration routine.
CALSTATUS?	Queries the state of DSA Enhanced Accuracy.
CALTEMPDELTA?	Queries the change in degrees Celsius since the last calibration.
CCALCONSTANTS	Controls the calibration constants for the center plug-in unit.
CHSKEW?	Queries the skew (time delay) measured by the probe-calibration routine.
LCALCONSTANTS	Controls the calibration constants for the left plug-in unit.
MCALCONSTANTS	Controls the calibration constants for the DSA.
RCALCONSTANTS	Controls the calibration constants for the right plug-in unit.
SAVEFACTORY	Saves the factory calibration constants to NVRAM.
SELFCAL	Determines Enhanced Accuracy calibration mode and initiates manual self-calibration.

Channel/Vertical

CH <*slot*> <*ui*>	Sets parameters for the specified plug-in unit channel.
CH <*slot*> ?	Queries parameters for all channels in a plug-in unit.
CH?	Queries parameter information for all installed channels.

Cursor

CURMODE	Selects the default cursor operating characteristics.
CURSOR	Selects cursor operating characteristics for a selected trace.
DOT1ABS	Positions the first split or paired cursor to a specified absolute location.
DOT2ABS	Positions the second split or paired cursor to a specified absolute location.
DOT1REL	Positions the first split or paired cursor relative to the DOT1ABS location.
DOT2REL	Positions the second split or paired cursor relative to the DOT2ABS location.
H1BAR	Positions the first horizontal-bar cursor to specified absolute location.
H2BAR	Positions the second horizontal-bar cursor to specified absolute location.
V1BAR	Positions the first vertical-bar cursor to specified absolute location.
V2BAR	Positions the second vertical-bar cursor to specified absolute location.

Data Transfer

ABBWFMPRE	Controls whether a WFMPRE? query returns all links or an abbreviated set of links.
BYT.OR	Sets the byte order for binary data transfer.
BYT/NR	Sets the number of bytes per data point.
BIT/NR	Sets the number of bits per data point.
CURVE	Transfers unscaled waveform data. Scaling information is included in the waveform preamble.
ENCDG	Selects ASCII or binary format for data transfers.
INPUT	Selects the memory location in which to store a waveform transferred to the DSA.

OUTPUT	Selects the stored or displayed waveform to be transferred from the DSA.
REPCURVE	Controls the fast transfer of trace data from the DSA.
SET?	Queries the current front-panel settings.
WAVFRM?	Queries the waveform preamble and data points for the waveform specified by OUTPUT.
WFMPRE	Sets the links of the waveform preamble.

Diagnostics

DIAG?	Queries the result of the diagnostic tests.
TEST	Performs self-test or extended-test diagnostics.

Display and Color

COLOR	Determines the colors used in the display.
COLORMAP	Determines color model for the display and assigns waveform colors in the standard model.
DISPLAY	Controls display intensity, number of graticules, and waveform display (dots or vectors).

External I/O

ALTINKJET	Controls HP Thinkjet, PaintJet, and LaserJet printers.
BITMAP	Controls screen capture by an external computer.
COPY	Produces a printout of the display.
DEBUG	Displays the ASCII commands on the front-panel as they are executed.
HPGL	Controls Tektronix HC100 plotters and other devices conforming to the HPGL format.
PIN8	Controls standard Epson 8-pin bit image graphics printers, such as the Tektronix 4644.
PIN24	Controls extended Epson 24-pin dot graphics printers.
RS232	Sets the parameters of the RS-232-C interface.
TEK4692	Controls Tektronix 4692 Color Graphics Copiers and Printers with 4692 emulation mode.
TEK4696	Controls Tektronix 4696 and Tektronix 4695 Color Ink-Jet Printers.
TEK4697	Controls the Tektronix 4697 Color Ink-Jet Printer.

Floppy Disk

ATTRIBUTE	Changes read/write attribute of files.
BASENAME	Specifies default filenames.
CD	Changes directories.
CHDIR	Changes directories.
CHKDISK	Checks the disk for inconsistencies, and makes corrections if any are found.
DCOPY	Copies waveforms and settings from one location to another.
DIR?	Queries directories for a list of files.
FORMAT	Formats a floppy disk.
MKDIR	Makes a directory.
RENAME	Renames a file.
RENDIR	Renames a directory.
RMDIR	Removes a directory.
SETDEV	Sets the device that stores and recalls waveforms and settings. The device can be disk or RAM.
STOFMT	Sets the data format for stored waveforms.

Label and Text

LABABS	Positions the label associated with the selected waveform to an absolute location.
LABEL	Defines and deletes labels, and controls whether they are displayed.
LABREL	Positions the label associated with a waveform to a location relative to its absolute location.
TEXT	Defines and positions a temporary text string on the display.
TEXT<*ui*>	Defines and positions up to twelve text strings on the display. These strings remain until removed.

Measurement

BASELINE	Sets the absolute value of the baseline when measurement tracking is turned off.
COMPARE	Controls comparison mode.
DAINT	Sets the data interval (one period or the entire measurement zone) for taking measurements.
DISTAL	Sets the distal (most distant) reference level, typically 90% of the baseline-to-topline value.
DLYTRACE	Sets the reference (delayed) waveform used with the PDELAY measurement.

HISTOGRAM	Initiates a vertical or horizontal histogram display for a selected trace.
HNUMBER	Selects the harmonic number when SMODE is set to HARM.
LMZONE	Sets the left limit of the measurement zone.
MEAS?	Executes and returns the values of the measurements in the measurement list (MSLIST).
<*meas*>?	Queries the value of the specified measurement (<*meas*> is a measurement parameter).
MESIAL	Sets the mesial (middle) reference level, typically 50% of the baseline-to-topline value.
MLEVEL	Determines if parameters are absolute voltages or percentages of baseline-to-topline values.
MS<*meas*>?	Queries measurement statistics (min, max, mean, and std dev) for the specified <*meas*>.
MSCOUNT	Sets the number of samples used to compute measurement statistics.
MSLIST	Specifies the measurements in the measurement list.
MSLOPE	Sets the crossing slope for measurements.
MSNUM?	Queries the number of measurements in the measurement list (MSLIST).
MSREP<*meas*>	Generates statistics for the specified measurement on repeated single-shot acquisitions.
MSREPMEAS	Generates measurement statistics on repeated single-shot acquisitions.
MSTAT?	Queries measurement statistics for the measurements in MSLIST.
MSTO	Defines parameters for measurements on groups of stored waveforms.
MSTO<*meas*>?	Queries the specified measurement for statistical information on groups of stored waveforms.
MSTOMEAS?	Queries measurements (in MSLIST) for statistical information on groups of stored waveforms.
MSYS	Controls display of the Measure major menu.
MTIME	Determines if the measurement zone limits are absolute units or waveform percentages.
MTRACK	Controls measurement tracking (whether you or the DSA set the baseline and topline values).
NHIST.PT	Sets the number of points to be acquired for a histogram display.
NWAVFRM	Sets the number of waveforms to be acquired for a histogram display.

PINDEX	Selects the peak index when SMODE is set to PEAK.
PROXIMAL	Sets the proximal (nearest) reference level, typically 10% of the baseline-to-topline value.
REFLEVEL	Sets a user-defined signal reference level.
REFSET	Sets the reference value(s) used in comparison mode (COMPARE is set to ON).
REFTRACE	Selects the reference waveform used with GAIN, PHASE, and SKEW <*meas*>.
REP<*meas*>	Selects a particular measurement to be made on repeated single-shot acquisitions.
REPMEAS	Controls measurements that are made on repeated single-shot acquisitions.
RMZONE	Sets the right limit of the measurement zone.
SMODE	Selects whether SMAG and SFREQ measurements are made on a selected harmonic or peak.
SNRATIO	Sets the amplitude of a noise rejection band centered on the MESIAL level.
STATISTICS	Controls whether statistics are calculated.
TOPLINE	Sets the absolute value of the topline when measurement tracking (MTRACK) is turned off.
TTAVERAGE	Sets the number of samples used by the TTRIG measurement.

Measurement Parameters (<meas>)

CROSS	The time from the trigger point to a specified reference-level crossing.
DELAY	The time between the first and last mesial crossing within the measurement zone.
DUTY	The percentage of a period that a waveform spends above the mesial level.
FALLTIME	The transition time of a falling pulse edge, from the distal to proximal levels.
FREQ	Frequency (reciprocal of the period measurement).
GAIN	Ratio of the peak-to-peak amplitudes of the reference waveform versus the selected waveform.
MAX	Maximum amplitude (most positive peak voltage).
MEAN	Average amplitude (arithmetic mean voltage).
MID	Amplitude midpoint, halfway between the maximum amplitude and the minimum amplitude.
MIN	Minimum amplitude (most negative peak voltage).

OVERSHOOT	Difference between the maximum amplitude and the topline value, given as a percentage of the difference between the topline and baseline values.
PDELAY	Propagation delay between mesial crossings on two waveforms (see DLYTRACE command).
PERIOD	The time between the first and next mesial crossing of the same slope.
PHASE	The phase relationship of the reference waveform to the selected waveform (±360°)
PP	Peak-to-peak value; the voltage difference between the maximum and minimum amplitude.
RISETIME	The transition time of a rising pulse edge, from the proximal to distal levels.
RMS	True root-mean-square voltage.
SFREQ	The spectral frequency (harmonic or peak).
SKEW	The propagation delay or time delay between mesial crossings on two different waveforms.
SMAG	The spectral magnitude (harmonic or peak).
THD	Total harmonic distortion.
TTRIG	The time between the main trigger point and the window trigger point.
UNDERSHOOT	Difference between the baseline value and the minimum amplitude, given as a percentage of the difference between the topline and baseline values.
WIDTH	The time between the first and next mesial crossing of the opposite slope.
YTENERGY	The energy represented under the curve of a Yt waveform. (Can be divided by the resistance of the circuit to yield power measurements.)
YTMNS_AREA	The difference between the area under a Yt curve above a specified reference level, and the area under the curve below that level.
YTPLS_AREA	The absolute value of all areas between a Yt waveform and a user-specified reference level.

Miscellaneous/System

ABSTOUCH	Mimics a touch to the front-panel display area, the major menu buttons or a knob turn.
CALIBRATOR	Controls the front-panel calibrator output.
DATE	Sets the date on the system calendar.

DEF	Defines logical names for command strings.
DSYMENU?	Queries which major menu is currently displayed.
DSYSTOFMT	Specifies date or hundredths of seconds for stored waveform time and date strings.
FEOI	Forces a message terminator in a command string.
FPANEL	Controls front-panel lockout.
FPUPDATE	Controls when front-panel readouts are updated.
INIT	Initializes the system.
LONGFORM	Controls whether the DSA returns full or abbreviated query responses and event information.
OPTIONS?	Queries for a list of installed options.
PATH	Determines whether queries return link-argument information or only the arguments.
POWERON?	Queries the number of times the DSA has been powered on.
PROBE	Determines the result of a probe button press.
SCLOCKD	Controls whether the sampling clock is dithered.
SPEAKER	Controls whether the DSA beeps when the display is touched.
TEKSECURE	Completely erases NVRAM.
TIME	Sets the time on the system clock.
UNDEF	Deletes logical names previously defined with DEF.
UPTIME?	Queries the number of hours the DSA has been powered on.
USERID	Saves a quoted string in nonvolatile RAM.

Status and Event

CONFIG?	Queries which type of plug-in units are installed.
EVENT?	Queries event-code information.
ID?	Queries version numbers of system firmware.
IDPROBE?	Queries the channel number of the probe ID button last pressed.
PIVERSION?	Queries version numbers of plug-in unit firmware.
RQS	Sets whether the DSA asserts the SRQ line after an event occurs (GPIB only).
SRQMASK	Controls (masks) reporting of certain classes of events.
STBYTE?	Queries status byte information (RS-232-C only).
UID	Specifies the serial numbers of the DSA and its plug-in(s).

Time Base/Horizontal

ADJTRACE<*ui*>	Controls the horizontal magnification and position of the selected waveform.
MAINPOS	Sets the horizontal position of the main waveform record with respect to the main trigger.
TBMAIN	Sets the main horizontal (time base) parameters.
TBWIN	Sets the window horizontal parameters.
WIN1POS	Sets the horizontal position of the window 1 waveform with respect to the window trigger.
WIN2POS	Sets the horizontal position of the window 2 waveform with respect to the window trigger.

Triggering

TR?	Queries the same information as: TRMAIN?;TRWIN?
TRLEVEL	Controls trigger DC level mode.
TRMAIN	Sets the main-trigger parameters.
TRWIN	Sets the window-trigger parameters.
TSMAIN?	Queries the time from the trigger point to the 0 point, for real-time single-shot acquisitions only.
WTMODE	Sets the window-triggering mode.

Waveform and Settings

ADJTRACE<*ui*>	Controls pan/zoom mode, vertical size and position, and window trace separation.
CLEAR	Discards acquired data for displayed waveforms.
DELETE	Deletes stored waveforms or front-panel settings from disk or RAM.
FPSLIST?	Queries a list of stored front-panel settings.
FPSNUM?	Queries the number of stored front-panel settings.
NEXTFPS	Selects index for the next stored setting.
NEXTSTO	Selects index for the next stored waveform.
NVRAM?	Queries the amount of available nonvolatile RAM.
PZMODE	Controls multiple waveform pan/zoom mode.
RECALL	Recalls stored front-panel settings from memory or disk.
RECOVER	Recovers previously deleted, stored waveforms from memory.
REMOVE	Discards displayed waveforms and descriptions.
SCANSTOWFM	Controls scanning of stored waveforms.
SELECT	Designates the selected waveform.

SETSEQ	Controls sequencing of front-panel settings.
STOLIST?	Queries a list of all stored waveforms.
STONUM?	Queries the number of stored waveforms.
STORE	Copies displayed waveforms and front panel settings to memory or disk.
TRACE <*ui*>	Defines a waveform and its characteristics.
TRANUM?	Queries the number of displayed waveforms.
WFMSCALING	Controls scaling to create waveforms in fast (integer) or high-precision (floating-point) mode.

Using The Command Set

As you can see, the DSA's command set is large. Attempting to gain familiarity with the entire command set via small BASIC programs written on a controller, can be tedious, time-consuming work. A simple solution to begin using and learning the command set is to connect an ASCII terminal to the DSA's RS-232-C port. After it is connected, turn on the RS-232 ECHO and VERBOSE mode. This can be done by typing e <RETURN> v<RETURN> on the terminal. (Type DEF? for a definition of these commands.) Then enter commands or queries and see how the DSA responds.

ABBwfmpre

ABBwfmpre determines whether the response to a WFMpre? query is abbreviated or includes all links. The power-on default setting is ABBwfmpre OFF.

Syntax:
```
ABBwfmpre<sp>{OFF|ON}
ABBwfmpre?
```

Arguments

- **OFF**—turns WFMpre? response abbreviation off. When ABBwfmpre is set to OFF, the WFMpre? response includes all 21 links of the WFMpre command. Refer to the WFMpre command for response format.

- **ON**—turns WFMpre? response abbreviation on. When ABBwfmpre is set to ON, the WFMpre? response is:

```
WFMPRE ACSTATE:<arg>,NR.PT:<NR1>,
    PT.FMT:<arg>,XINCR:<NR3>,
    XMULT:<NR3>,XZERO:<NR3>,
    YMULT:<NR3>,YZERO:<NR3>
```

Examples

`ABB OFF`
turns WFMpre? response abbreviation off.

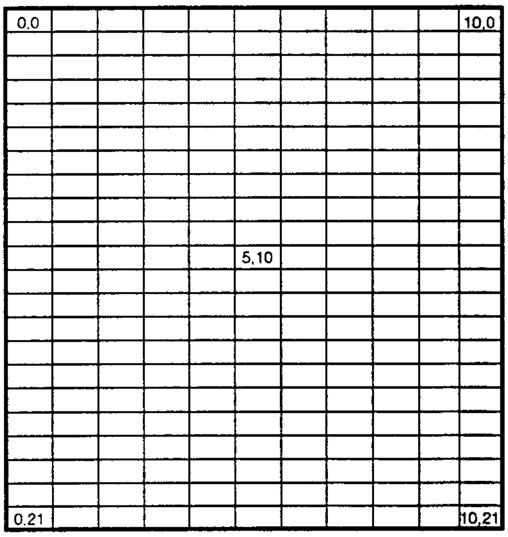
ABStouch

ABStouch activates a location on the front panel by giving its X,Y coordinates. ABStouch always works, regardless of the state of the front panel (FPAnel ON/OFF) or **TOUCH PANEL** button. Every front panel touch, whether from ABStouch or the front panel, is stored in a 20-deep first in, first out (FIFO) buffer.

Syntax: `ABStouch<sp>{CLEar|<x>,<y>}`
`ABStouch?`

Range: <x> = 0 to 11; <y> = 0 to 21

X,Y touch panel screen coordinates range from 0,0 (upper left) to 10,21 (lower right):



X, Y Touch Panel Screen Coordinates

Coordinates of the front panel buttons are listed in the following table:

Front Panel Button X,Y Coordinates

Button	X,Y	Button	X,Y
Waveform	11,0	Digitizer Run/Stop	11,8
Trigger	11,1	Autoset	11,9
Measure	11,2	Hardcopy	11,10
Store/Recall	11,3	Enhanced Accuracy	11,11
Utility	11,4	Left Knob Counterclockwise	12,-<ui>†
Touch Panel	11,5	Left Knob Clockwise	12,+<ui>
Right Fine Button	11,6	Right Knob Counterclockwise	13,-<ui>
Left Fine Button	11,7	Right Knob Clockwise	13,+<ui>

†*The value <ui> is the number of knob clicks.*

Arguments

- **CLEar** – empties the 20-deep FIFO buffer in which front panel touches are stored.

Query Responses

- **ABStouch?** – returns the oldest touch coordinates in the FIFO and removes them from the buffer. If no touches are in the buffer, ABStouch? returns:

```
ABStouch -1,-1
```

Note: You cannot use ABStouch to touch a channel button or a probe ID button.

Examples

`ABS CLE`
removes all front panel touches from the buffer.

`ABS 11,0`
activates the waveform button on the front panel.

ADJtrace

FRESolution
FSPan
HMAg
HPOsition
HVPosition
HVSize
PANzoom
TRSep
VPOsition
VSIze

ADJtrace adjusts the displayed position of the specified waveform without modifying the horizontal (time base) or vertical (channel) parameters.

Syntax:

```
ADJtrace<ui><sp><link>:<arg>
ADJtrace?
ADJtrace<ui>?[<sp><link>]
```

Range: `<ui>` = 1 to 8, and specifies the waveform.

Note: Certain links only apply to waveforms created in floating-point mode or integer mode. For information on waveform modes, refer to the WFMScaling command.

Query Responses

- **ADJtrace?** – returns the displayed position of all defined waveforms in low-to-high waveform order. The response format is:

```
ADJTRACE<ui> <link>:<arg>...[;ADJTRACE<ui>
   <link>:<arg>...]
```

- **ADJtrace<ui>?** — returns the displayed position for the specified waveform.

```
ADJTRACE<ui> PANZOOM:<arg>,HMAG:<NR3>,
   HPOSITION:<NRx>,HVPOSITION:<NR3>,
   HVSIZE:<NR3>,TRSEP:<NR3>,VPOSITION:<NR3>,
   VSIZE:<NR3>,FSPAN:<NR3>,FRESOLUTION:<NR3>
```

ADJtrace? Default Link Responses. Several links can only be set under restricted conditions (for example, you can only set VSIze on a floating-point waveform), but you can query any link at any time. The restricted links return the following predefined values if you query them under conditions when they cannot be set:

ADJtrace? Query-Only Set Responses

Link	Response
FREsolution	-1.0E+0
FSPan	-1.0E+0
HMAg	-1.0E+0
HPOsition	1.0E+16
HVPosition	1.0E+16
HVSize	-1.0E+0
TRSep	1.0E+16
VPOsition	1.0E+16
VSIze	-1.0E+0

Examples

```
ADJ2?
```

returns display position information, such as:

```
ADJTRACE2 PANZOOM:OFF, HMAG:-1.0E+0,
    HPOSITION:1.0E+16,HVPOSITION:1.0E+16,
    HVSIZE:-1.0E+0, TRSEP:1.0E+16,
    VPOSITION:1.0E+16,VSIZE:-1.0E+0,
    FSPAN:4.882E+6,FRESOLUTION:9.765E+3
```

FREsolution

Query Only. FREsolution returns the FFT bin width, or frequency per point.

Syntax: `ADJtrace<ui>?<sp>FREsolution`

Returns: `<NR3>`

Examples

`ADJ2? FRE`
returns the FFT bin width, such as:
`ADJTRACE2 FRES:9.765E+3`

Only valid for FFT traces.

FSPan

Query Only. FSPan returns the FFT frequency per division.

Syntax: `ADJtrace<ui>?<sp>FSPan`

Returns: `<NR3>`

Examples

`ADJ2? FSP`
returns the FFT frequency per division, such as:
`ADJTRACE2 FSPAN:4.882E+6`

Only valid for FFT traces.

HMAg

HMAg sets the waveform horizontal magnification factor when ADJtrace<ui> PANzoom is set to ON.

Syntax:
```
ADJtrace<ui><sp>HMAg:<NRx>
ADJtrace<ui>?<sp>HMAg
```

Range: `<NRx>` = 1, 2, 2.5, 4, 5, 8, 10, 16, 20, 25, 40, 50, 80, 100, 160, 200, 250, 400, 500, 800,1000, 2000, or 5000

The HMAg value depends on the record length of TBMain or TBWin. The maximum HMAg value for each length is shown in the following table.

Maximum HMAG Values

Record LENGTH	**Maximum HMAG**	**Record LENGTH**	**Maximum HMAG**
512	50	8192	1000
1024	100	10240	1000
2048	200	16384	2000
4096	500	20464	2000
5120	500	32768	5000

Examples

`ADJ4 HMA:80`
selects a horizontal magnification factor for record lengths greater than 512.

HPOsition

HPOsition sets the waveform horizontal position when ADJtrace<ui> PANzoom is set to ON.

Syntax:

```
ADJtrace<ui><sp>HPOsition:<NRx>
ADJtrace<ui>?<sp>HPOsition
```

Range: The HPOsition range `<NR1>` is in waveform points from 0 (zero) to an upper value determined by the horizontal magnification (HMAg) and the record length, using the following formula:

length - ceil (10.24 * max_HMAg / HMAg)

where the ceil( ) is the smallest integer value greater than or equal to the value in parentheses. Ceil rounds fractions to the next higher integer. For example, for a record length of 4096, the max_HMAg value is 500.

Assume the actual HMAg is 50. With these conditions, the HPOsition range is 0 to:

[4096 - ceil (10.24 * 500/50)] =
[4096 - ceil (102.4)] = [4096 - 103]= 3993

Examples

`ADJ2 HPO:300`
sets the horizontal position for waveform 2.

HVPosition

For XY waveforms created in floating-point mode, HVPosition sets the graphical position of the horizontal component of the waveform.

Syntax: `ADJtrace<ui><sp>HVPosition:<NRx>`
`ADJtrace<ui>?<sp>HVPosition`

Range: `<NRx>` = -1E+15 to 1E+15

Examples

`ADJ4 HVP:-8.9E-6`
sets the horizontal screen position for waveform 4.

HVSize

For XY waveforms created in floating-point mode, HVSize sets the graphical size of the horizontal component of the specified XY waveform.

Syntax: `ADJtrace<ui><sp>HVSize:<NRx>`
`ADJtrace<ui>?<sp>HVSize`

Range: `<NRx>` = 1E-15 to 1E+15

Examples

`ADJ2 HVS:4.5E-2`
sets the horizontal component screen size for waveform 2.

PANzoom

PANzoom sets horizontal magnification (Pan/Zoom mode) ON or OFF for the selected waveform. When PANZOOM is set to ON, you can horizontally magnify selected sections of a displayed waveform with ADJTRACE<*ui*> HMAG.

PANzoom is always set to ON for stored or scalar waveforms, but you cannot set it ON for XY waveforms. To control Pan/Zoom mode for all waveforms, refer to the PZMode command.

Syntax:
```
ADJtrace<ui><sp>PANzoom:{OFF|ON}
ADJtrace<ui>?<sp>PANzoom
```

Examples

```
ADJ2 PAN:ON
```
turns on horizontal magnification for waveform 2.

TRSep

For waveforms created in integer mode, TRSep (TRace SEParation) sets the window waveform separation in graticule divisions. The waveforms must have been created on the WIN1 or WIN2 time base, and cannot be XY waveforms.

Syntax:
```
ADJtrace<ui><sp>TRSep:<NRx>
ADJtrace<ui>?<sp>TRSep
```

Range: `<NRx>` = −5.0 to +5.0

Examples

```
ADJ3 TRS:-2.2
```
sets the window separation for waveform 3.

VPOsition

For waveforms created in floating-point mode, VPOsition sets the waveform vertical screen position.

Syntax:
```
ADJtrace<ui><sp>VPOsition:<NRx>
ADJtrace<ui>?<sp>VPOsition
```

Range: `<NRx>` = -1E+15 to 1E+15

Examples

`ADJ4 VPO:3.9E+2`
sets the vertical screen position for waveform 4.

VSIze

For waveforms created in floating-point mode, VSIZE sets the waveform vertical graphical size.

Syntax:
```
ADJtrace<ui><sp>VSIze:<NRx>
ADJtrace<ui>?<sp>VSIze
```

Range: `<NRx>` = 1E-15 to 1E+15

Examples

`ADJ4 VSI:4.5E-2`
sets the vertical screen size for waveform 4.

ALTinkjet

DIRection
FORMat
PORt
SECUre

ALTinkjet specifies printing parameters for HP Thinkjet, LaserJet, and PaintJet printers operating in HP graphics mode.

Syntax:
```
ALTinkjet<sp><link>:<arg>
ALTinkjet?[<sp><link>]
```

Note: ALTinkjet does not support Thinkjet and LaserJet printers operating in Epson emulation mode.

Examples

`ALT?`
returns printing parameters, such as:
```
ALTINKJET DATACOMPRESS:ON,
    DATAFORMAT:BINHEX,DIRECTION:VERT,
    FORMAT:SCREEN,PORT:CENTRONICS
```

DIRection

DIRection selects the printing orientation.

Syntax:
```
ALTinkjet<sp>DIRection:{HORiz|VERt}
ALTinkjet?<sp>DIRection
```

Arguments

- **HORiz**—prints rows left-to-right and top-to-bottom
- **VERt**—prints columns bottom-to-top and left-to-right.

Examples

`ALT DIR:HOR`
sets printer orientation to horizontal.

FORMat

FORMat selects the printing format.

Syntax:
```
ALTinkjet<sp>FORMat:{DIThered|DRAft|HIRes
                    |REDuced|SCReen}
ALTinkjet?<sp>FORMat
```

Arguments

- **DIThered** – prints with reduced saturation and increases contrast.(PaintJet only.)
- **DRAft** – prints selected fields in reverse video.
- **HIRes** – prints front panel intensified regions.
- **REDuced** – prints a fourth of the size of DRAft, but does not show intensified regions.
- **SCReen** – prints an exact replica of the screen without reformating for PaintJet.

Examples

`ALT FORM:DRA`
prints fields in reverse video.

PORt

PORt specifies the output port for the printer.

Syntax:
```
ALTinkjet<sp>PORt:{CENTRonics|DISK|GPIb|
                   RS232}
ALTinkjet?<sp>PORt
```

Arguments

- **CENTRonics** – specifies the centronics printer port.
- **DISk** – specifies the disk.
- **GPIb** – specifies the GPIB port.
- **RS232** – specifies the RS232-C port.

Examples

`ALT POR:RS232`
selects the RS-232-C port for hardcopy output.

SECUre

SECUre specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax:
```
ALTinkjet<sp>SECUre:{ON|OFF}
ALTinkjet?<sp>SECUre
```

Examples

`ALT SECU:OFF`
turns off the SECUre function.

ATTRibute

ATTRibute sets file read access.

Syntax:
```
ATTRibute<sp><qstring>,{"+r"|"-r"}
ATTRibute?<sp><qstring>
```

Arguments

- **<qstring>** – specifes a file on the disk.
- **"+r"** – sets file access to read only.
- **"-r"** – sets file access to read/write.

Examples

`ATTR "A:SECND.WFA","+r"`
makes SECND.WFA read only.

AUTOAcq

MEMWrap
REPS
SRQ
TRAce

AUTOACQ selects waveforms to be stored in repetitive single trigger mode (and Act on Delta mode when DELTa SAVe is set to ON) or to be transferred over the bus using the REPCURVE command. AUTOACQ also controls memory wrap in repetitive single trigger acquisition and generation of SRQ's on each acquistion.

Syntax:

```
AUTOAcq<sp><link>:<arg>
AUTOAcq?[<sp><link>]
```

MEMWrap

When MEMWRAP is ON, waveforms acquired in repetitive single trigger mode (and Act on Delta mode when DELTa is set to SAVe) are stored in a circular memory buffer. All available memory is allocated for repetitive single trigger acquisitions, and when memory is full, the oldest acquisitions are overwritten. Acquisition will continue until the digitizer is stopped, and the most recent acquisitions remain in memory. The number of acquistions that remain in memory is set by NREptrig (except when MEMWrap is ON, then NREptrig is ignored).

When MEMWRAP is OFF, repetitive single trigger acquisition stops when memory is full or when the number of waveform records specified by NREPTRIG have been acquired.

Syntax:

```
AUTOAcq<sp>MEMWrap:{OFF|ON}
AUTOAcq?<sp>MEMWrap
```

Examples

```
AUTOA MEMW:ON
```

REPS

Query only link that returns the number of acquisitions that occurred during the last or current repetitive single trigger.

Syntax: `AUTOAcq?<sp>REPS`

Examples

`AUTOA? REPS`
returns a count, such as:
`AUTOACQ REPS:116`

SRQ

SRQ enables or disables the generation of a service request (SRQ) for each acquisition. For Act on Delta, this command has the same effect as DELTa SRQ. When OFF, an SRQ is only generated after all acquisitions are completed or when the digitizer is stopped.

Syntax:
```
AUTOAcq<sp>SRQ:{OFF|ON}
AUTOAcq?<sp>SRQ
```

Examples

`AUTOA SRQ:ON`

TRAce

TRACE<*ui*> specifies which traces are stored after each repetition. Only waveforms that can be acquired concurrently may be set to ON. For Act on Delta, DELTa SAVe must be ON. At least one trace must be specified.

Syntax:
```
AUTOAcq<sp>TRAce<ui>:{OFF|ON}
AUTOAcq?<sp>TRAce<ui>
```

Query Note: AUTOACQ returns settings (ON or OFF) for defined traces only.

Examples

`AUTOA TRA2:ON`

AUTOSet

HORiz
VERt

AUTOSet controls vertical and horizontal automatic ranging and positioning of input signals on the selected waveform for both acquired and stored waveforms.

For acquired signals, the vertical size is set and the time base is adjusted. For stored waveforms, the display is scaled.

Syntax:
```
AUTOSet<sp>{STARt|UNDO|<link>:<arg>}
AUTOSet?[<sp><link>]
```

Arguments

- **STARt** – autosets the selected waveform. If no waveform is selected, the DSA samples all channels and autosets the first signal it encounters. AUTOSet completion is signaled with event code 464, "Autoset complete."
- **UNDO** – cancels a previous AUTOSet and returns to the settings in effect before the last AUTOSet STARt command.

Note: When the Main time base is not triggered, you cannot autoset a Window waveform.

Examples

`AUTOS UNDO`
cancels the previous AUTOSet.

`AUTOS?`
returns vertical and horizontal parameters, such as:

```
AUTOSET HORIZ:PERIOD,VERT:PP
```

HORiz

HORiz determines how autoset affects the horizontal display of the input signal.

Syntax:
```
AUTOSet<sp>HORiz:{EDGe|OFF|PERiod|PULse}
AUTOSet?<sp>HORiz
```

Arguments

- **EDGe** – displays one edge of the input signal expanded in the center of the display. A rising edge is displayed when TRMain SLOpe is PLUs. A falling edge is displayed when TRMain SLOpe is MINUs. EDGe is useful for preparing input signals for RISetime? and FALltime? measurements.
- **OFF** – turns off horizontal autoset.
- **PERiod** – displays at least three complete waveform cycles. PERiod is useful for preparing input signals for DUTy?, FREq?, MEAN?, PP?, PERiod?, and RMS? measurements.
- **PULse** – displays one pulse on the display; whether the pulse is positive-going or negative-going is set by TRMain SLOpe. PULse is useful for preparing input signals for WIDth? measurements.

Examples

```
AUTOS HOR:PER
```
displays at least three complete waveform cycles.

VERt

VERT controls how autoset affects the vertical sensitivity (gain) and offset of the input signal.

Syntax:

```
AUTOSet<sp>VERt:{ECL|OFF|PP|TTL}
AUTOSet?<sp>VERt
```

Arguments

- **ECL** – turns vertical autoset ON and the vertical and trigger settings are preset to ECL logic levels.
- **OFF** – turns off vertical autoset.
- **PP** – turns vertical autoset ON and the channel sensitivity and gain are set to display four to nine divisions of peak-to-peak amplitude; centered on the middle value.
- **TTL** – turns vertical autoset ON and the vertical and trigger settings are preset to TTL logic levels.

Examples

`AUTOS VER:ECL`
turns on vertical autoset and settings are preset to ECL logic levels.

AVG

AVG sets averaging ON or OFF for the vertical expression component (<y exp>) of the waveform description of the selected waveform. For YT waveforms, <y exp> defines the waveform. See the TRAce command for complete <y exp> syntax.

Syntax:

```
AVG<sp>{OFF|ON}
AVG?
```

Arguments

- **OFF**–removes the enclosing AVG() when <y exp> is enclosed with AVG(). You cannot set AVG OFF when <y exp> is not enclosed with AVG().
- **ON**–encloses <y exp> with AVG(), or AVG() replaces ENV() when <y exp> is enclosed with ENV. You cannot set AVG to ON if the selected waveform is XY or has only stored and/or scalar components.

Query Responses

- **AVG?**–returns the state of averaging for the entire <y exp> of the selected waveform. AVG ON means the entire <y exp> is enclosed by AVG. AVG OFF means the entire <y exp> is not enclosed, although an AVG function may be embedded within the description.

Examples Using AVG

<y exp> **Before**	**Command**	<y exp> **After**
L2	**AVG ON**	AVG(L2)
L1	**AVG OFF**	-error-
ENV(C1-C2)	**AVG ON**	AVG(C1-C2)
AVG(R1)	**AVG OFF**	R1
AVG(C4)	**AVG ON**	AVG(AVG(C4))

Examples

```
AVG ON
```

averages the vertical expression component of the waveform.

AVGType

BACKWeight
SUMMation

AVGType selects the type of averaging that is performed by the DSA.

Syntax:

```
AVGType<sp>{BACKWeight|SUMMation}
AVGType?
```

Arguments

- **BACKWeight** – specifies exponentially backweighted averaging. The current average value is weighted more heavily for recent acquisitions and less heavily for older acquisitions. This type of averaging allows a continuous display of the average data since the average value is computed after each acquisition. New data is constantly incorporated into the average value, reflecting slow changes in the input waveform. BACKWeight is the default AVGType.

- **SUMMation** – specifies standard summation averaging. Each acquisition is equally weighted in the average output. The average is not computed until the specified number of acquisitions has been completed.

Examples

```
AVGT SUMM
```

BASeline

The BASeline command sets the vertical baseline level for measurements.

Syntax: `BASeline<sp><NRx>`
`BASeline?`

Range: `<NRx>` = Any legal value.

BASeline sets the baseline level when MTRack (measurement tracking) is set to OFF or TOPline. BASeline is ignored when MTRack is set to BOTh or BASeline.

Examples

```
BAS -8.5E-1
```

BASEName

FPS
HCP
STO

BASENAME changes the default base name (or prefix) for disk-stored waveforms, front-panel settings, or hardcopy files. Base names may be up to eight characters long. Base names have an index that increments to ensure distinct automatic file naming. If a new base name is assigned, the index can increment only until all eight character spaces are used up. (For example, if you assign a seven letter base name, only one character is left for the index; it can only increment from 0 to 9.) STO is the default base name for stored waveforms, FPS is the default base name for front-panel settings, and HCP is the default base name for hard-copy files. In each case an index value is appended to the basename to form a filename. Filenames have extensions (or suffixes) that indicate the type of data contained in the file. For example:

STO12.WFA

STO is the basename of the file.
12 is the index counter.
WFA indicates that the data is a stored waveform in ASCII format.

Syntax:
```
BASEName<sp><link>:<qstring>
BASEName? [<sp><link>]
```

Range: `<qstring>` = $\leq$ 8 characters

FPS

FPS allows you to change the default base name for disk-stored front-panel settings files.

Syntax:
```
BASEName<sp>FPS:<qstring>
BASEName?<sp>FPS
```

Examples

`BASEN FPS:"PANEL"`
sets the front-panel settings basename to PANEL.

HCP

HCP allows you to change the default base name for disk-stored hardcopy files.

Syntax:
```
BASEName<sp>HCP:<qstring>
BASEName?<sp>HCP
```

Examples

`BASEN HCP:"PAPER"`
sets the hardcopy basename to PAPER.

STO

STO allows you to change the default base name for disk-stored waveform files.

Syntax:
```
BASEName<sp>STO:<qstring>
BASEName?<sp>STO
```

Examples

`BASEN STO:"WAVES"`
sets the stored waveform basename to WAVES.

BITMap

DATACompress
DATAFormat
DIRection
FORMat
PORt

BITMap specifies printing parameters for screen captures, in which data from the front panel display is processed by an external computer. Screen capture data include a title block and a pixel block.

Syntax:
```
BITMap<sp><link>:<arg>
BITMap?[<sp><link>]
```

BITMap Title Block. The title block contains three ASCII strings terminated by new-line characters. The first string includes the DSA's instrument name, time and date, and the serial number. The second line contains the number of pixels per raster line. The third line gives the number of raster lines.

When BITMap DATAFormat is set to BINary, the title block is terminated with an ASCII NULL character following the third new-line character.

When BITMap DATAFormat is set to BINHex, the title block is terminated with the third new-line character.

BITMap Pixel Block. The pixel block is a stream of data bytes. The DATACompress and DATAFormat links determine the format.

Examples

```
BITM?
```
returns screen capture printing information, such as:

```
BITMAP COLOR0:4095,COLOR1:0,COLOR2:3945,
    COLOR3:1776,COLOR4:2364,COLOR5:1020,
    COLOR6:2457,COLOR7:3840,DIRECTION:HORIZ,
    FORMAT:DITHERED,PORT:CENTRONICS,SECURE:OFF
```

DATACompress

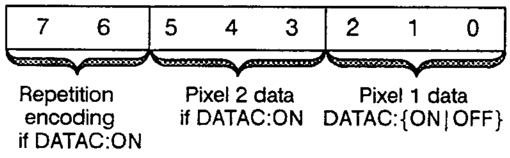
DATACompress specifies the pixel block data compression mode.

Syntax:
```
BITMap<sp>DATACompress:{OFF|ON}
BITMap?<sp>DATACompress
```

Arguments

- **OFF** – sets the data format such that each byte contains one 3-bit pixel value in the three least-significant bits.
- **ON** – selects a format where each byte contains two 3-bit pixel values, with the first pixel in the least-significant three bits (see the illustration below). The two most-significant bits in the byte encode the data repetition pattern, which is discussed below.

Pixel Block Data Byte. The following figure shows the bits in a pixel block data byte:



Bits in a Pixel Block Data Byte

Repetition Encoding. The table below lists the binary repetition encoding in bits 7 and 6 of the pixel data byte.

Data Repetition Encoding

Bit 7	Bit 6	Meaning
0	0	Following bytes contain repetition counts
0	1	Data pattern repeats once
1	0	Data pattern repeats twice
1	1	Data pattern repeats three times

When bits 7 and 6 encode the values 1 (01), 2 (10), or 3 (11), the pixel data is repeated one, two, or three times, respectively.

When bits 7 and 6 have the value 0 (00), the next one or two data bytes contain the repetition count. If the next byte has the decimal value 4 to 255, then that value is the pattern repetition count. If the next byte has the decimal value 1 to 3, then that value is the high-order bits of a 10-bit repetition count and the following byte contains the lower eight bits.

Example 1. The first byte produces one repetition of data 5,3; the second byte produces two repetitions of 2,1; the third byte produces three repetitions of 5,2.

Repetition Encoding in One Byte

Data Bytes	Resulting Pixel Values
01011101	5,3
10001010	2,1,2,1
11010101	5,2,5,2,5,2

Example 2. The first byte contains data 7,7 and repetition encoding of 0 to find the repetition count in the next byte. The second byte contains the repetition count of 10.

Repetition Encoding in Two Bytes

Data Bytes	Resulting Pixel Values
00111111	7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7
00001010	

Example 3. The first byte contains data 5,5 and repetition encoding of 0 to find the repetition count in the next byte. The second byte has the value 1, which means it contains the two high-order bits of a 10-bit repetition value. The third byte contains the lower eight bits, for a repetition count of 265.

Repetition Encoding in Three Bytes

Data Bytes	Resulting Pixel Values
00101101	5,5,5,5,5,5,5,5,5,5... (260 more 5,5 pairs)
00000001	
00001001	

Examples

`BITM DATAC:ON`
uses two 3-bit pixels per byte data format.

DATAFormat

DATAFormat specifies the pixel block data format.

Syntax:
```
BITMap<sp>DATAFormat:{BINary|BINHex}
BITMap?<sp>DATAFormat
```

Arguments

- **BINary** – format specifies that data are output in a stream without delimiters.
- **BINHex** – format specifies that data are output as ASCII hexidecimal bytes and each raster line is terminated with a new-line character.

Examples

`BITM DATAF:BIN`
outputs data in a continuous stream.

DIRection

DIRection selects the printing orientation.

Syntax:
```
BITMap<sp>DIRection:{HORiz|VERt}
BITMap?<sp>DIRection
```

Arguments

- **HORiz** – prints rows left to right and from top to bottom.
- **VERt** – prints columns bottom to top and from left to right.

Examples

`BITM DIR:HOR`
selects a horizontal printer orientation.

FORMat

FORMat selects print formatting.

Syntax:
```
BITMap<sp>FORMat:{DIThered|DRAft|HIRes|
                  REDuced|SCReen}
BITMap?<sp>FORMat
```

Arguments

- **DIThered** – reduces saturation for icon and text backgrounds to improve print contrast for the Tektronix 4692, Tektronix 4696, and Tektronix 4697 printers.
- **DRAft** – prints black-on-white background except for selected icons or text which are printed white-on-black background.
- **HIRes** – dithers icon and text backgrounds and increases foreground saturation to improve contrast for monochrome printers with limited gray-scale capability.
- **REDuced** – prints black-on-white background only.
- **SCReen** – specifies one-to-one mapping of 3-bit pixel information.

Examples

`BITM FORM:DIT`
reduces saturation for better print contrast.

PORt

PORt specifies the output port for the printer.

Syntax:
```
BITMap<sp>PORt:{CENTRonics|DISk|GPIb
                              |RS232}
BITMap?<sp>PORt
```

Arguments

- **CENTRonics** – specifies the centronics printer port.
- **DISk** – specifies the disk.
- **GPIb** – specifies the GPIB port.
- **RS232** – specifies the RS232-C port.

Examples

`BITM POR:GPI`
selects the GPIB port for hardcopy output.

SECUre

SECUre, when set to ON, specifies that only the waveform(s) and the graticule are sent to the printer.

Syntax:
```
BITMap<sp>SECUre:{ON|OFF}
BITMap?<sp>SECUre
```

Examples

`BITM SECU:OFF`
Turns off the SECUre function.

BIT/nr

BIT/nr sets the number of bits per binary waveform point. When BIT/nr is set to 8, BYT/nr is automatically set to 1. When BIT/nr is set to 16, BYT/nr is automatically set to 2.

Syntax: `BIT/nr<sp>{8|16}`
`BIT/nr?`

Arguments

- **8** – selects eight bits per waveform point.
- **16** – selects 16 bits per waveform point.

Note: This command does not affect the binary format of stored waveforms.

Examples

`BIT/ 16`
sets the the number of bits per binary waveform point to sixteen.

BYT/nr

BYT/nr sets the number of bytes per binary waveform point. When BYT/nr is set to 1 BIT/nr is automatically set to 8. When BYT/nr is set to 2 BIT/nr is automatically set to 16.

Syntax: `BYT/nr<sp>{1|2}`
`BYT/nr?`

Arguments

- **1** – selects one byte per waveform point.
- **2** – selects two bytes per waveform point.

Note: This command does not affect the binary format of stored waveforms.

Examples

`BYT/ 1`
sets the the number of bytes per binary waveform point to one.

BYT.or

BYT.or selects which byte of the binary waveform (<bblock>) data is transmitted first during a binary CURVe, binary measurement, or binary histogram data transfer. Correct byte order depends on the controller.

Syntax:
```
BYT.or<sp>{LSB|MSB}
BYT.or?
```

Arguments

- **LSB** – selects the least significant byte to be transmitted first. LSB has a faster data transfer rate.
- **MSB** – selects the most significant byte to be transmitted first. This is the power-on default.

Note: This command does not affect the binary format of stored waveforms.

Examples

```
BYT. LSB
```
selects the least significant bit as the first byte transmitted.

CALIbrator

AMPLitude
FREq
IMPedance

CALIbrator controls the front panel calibrator output signal.

Syntax:
```
CALIbrator<sp><link>:<arg>
CALIbrator?[<sp><link>]
```

AMPLitude

AMPLitude selects the amplitude of the calibrator square wave signal, depending on the value of the CALIbrator FREq link. When FREq is 0 Hz, you can set the AMPLitude to a DC level. When FREq is 1 kHz or 1 MHz, AMPLitude is forced to a +5 V or +0.5 V square wave, respectively.

Syntax:
```
CALIbrator<sp>AMPLitude:<NRx>
CALIbrator?<sp>AMPLitude
```

Range: `<NRx>`

AMPLitude Range

FREq	AMPLitude Values
0 Hz	-10.000 V to +9.9951 V (DC level)
1 kHz	+5 V (0 to +5 V square wave)
1.024 MHz	+0.5 V (0 to +0.5 V square wave)

Examples

```
CALI AMPL:5
```

FREq

FREq selects the frequency of the square wave calibrator output: 0 Hz, 1 kHz, or 1.024 MHz.

Syntax:
```
CALIbrator<sp>FREq:<NRx>
CALIbrator?<sp>FREq
```

Examples

```
CALI FRE:1000
```

IMPedance

Query Only. IMPedance returns the output impedance in ohms. The IMPedance response depends on the CALIbrator FREquency. IMPedance returns 50 when FREquency is 1 MHz; it returns 450 when FREquency is 0 Hz or 1 kHz.

Syntax: `CALIbrator?<sp>IMPedance`

Returns: 50 or 450

Examples

`CALI? IMP`

returns an output impedance, such as:

```
CALIBRATION IMPEDANCE:450
```

CALProbe

FULl
SHOrt

Set Only. CALProbe initiates the probe calibration routine. The routine includes probe calibration, deskew, and an optional probe compensation adjustment.

Successful completion of probe calibration is signaled with event code 475, "Probe calibration completed and passed."

Syntax: `CALProbe<sp><link>:<arg>`

FULl

FULl provides a pause in the calibration routine for manual probe compensation adjustment. When you have completed the probe compensation adjustment, touch the front panel display to terminate the CALProbe routine.

Syntax: `CALProbe<sp>FULl:<slot><ui>`

Range: `<ui>` = 1 to the number of plug-in unit channels.

Examples

`CALP FUL:R2`
runs the calibration routine and pauses for manual adjustments.

SHOrt

SHOrt does not pause in the calibration routine for manual probe compensation adjustment. The routine terminates after providing probe calibration and deskewing.

Syntax: `CALProbe<sp>SHOrt:<slot><ui>`

Range: `<ui>` = 1 to the number of plug-in unit channels.

Examples

`CALP SHO:C4`
runs the calibration routine without pausing.

CALStatus

Query Only. CALStatus returns the calibration (accuracy) status of the DSA.

Syntax: `CALStatus?`

Query Responses

- **NENHANCED**—The DSA is in normal accuracy state while warming up.
- **ENHANCED**—The DSA is in Enhanced Accuracy state after warming up.
- **NEWCONFIG**—A new plug-in unit has been installed and is warming up.

Examples

`CALS?`
returns calibration (accuracy) status, such as:
`CALSTATUS ENHANCED`

CALTempdelta

Query Only. CALTempdelta returns the change of temperature in degrees Celsius from the last calibration.

Syntax: `CALTempdelta?`

Returns: `<NR3>`

Examples

`CALT?`
returns the change in temperature, such as:
`CALTEMPDELTA 3.0E+0`

CCAIconstants

CCAIconstants sets or queries the calibration constants of the center plug-in unit.

Syntax: `CCAlconstants<sp><ui>:<NRx>`
`CCAlconstants?[<ui>]`

Range: `<ui>` is the constant (range is plug-in unit specific).
`<NRx>` is the value of the constant.

Note: You can only set CCAIconstants when an internal jumper has been installed by a qualified service person.

Examples

`CCA? 33`
returns the center plug-in unit calibration constant, such as:
```
CCALCONSTANTS 33:5.003517E-2
```

CD

CD changes the current working directory on the floppy disk. This command is identical to the CHDIR command.

Syntax: `CD<sp><qstring>`
`CD?`

Range: `<qstring>` specifies the name of the directory.

Query Responses

- **CD?**—returns the current working directory.

Examples

`CD?`
returns the current working directory, such as:
```
CD "A:\WAVEFRMS"
```

CH

AMPoffset
BW
BWHi
BWLo
COUpling
IMPedance
MNSCoupling
MNSOffset
MNSProbe
OFFSet
PLSCoupling
PLSOffset
PLSProbe
PROBe
PROTect
SENsitivity
UNIts
VCOffset

CH sets vertical channel parameters of the plug-in units.

Syntax:
```
CH<slot><ui><sp><link>:<arg>
CH?
CH<slot>?
CH<slot><ui>?[<sp><link>]
```

Range: <ui> = 1 to the number of plug-in unit channels.

Links That Affect Only Non-Differential Amplifiers

COUpling	PROBe

Links That Affect Only Differential Amplifiers

AMPoffset	PLSCoupling
MSCoupling	PLSOffset
MNSOffset	PLSProbe
MNSProbe	PROTect
	VCOffset

Links That Affect All Amplifiers

BW	IMPedance
BWHi	OFFSet
BWLo	SENsitivity
	UNIts

11A33 Amplifier Considerations. The IMPedance, SENsitivity, MNSCoupling, PLSCoupling, and PROTect links of the 11A33 Differential Amplifier affect one another. Modifying one of these links may change the value of another. If a link is changed, no warning message is issued. Refer to the link entries for examples.

Level 2 TekProbe. In some cases, attaching a Level 2 TekProbe to an input channel may cause a plug-in unit to reject coupling or impedance values that are normally valid. See the appropriate plug-in unit *User Reference Supplement* for information.

Note: Plug-in units that support the BW link do not support BWHi or BWLo, and vice versa.

Query Responses

- **CH?**—returns the vertical parameter links and arguments for all channels in low-to-high numeric order and in L, C, R <slot> sequence.

- **CH<slot>?**—returns the same information as CH?, for all channels in the specified <slot> in low-to-high numeric order.

- **CH<slot><ui>?**—returns links and arguments for the specified channel, depending on the plug-in unit.

Plug-in units that support BWHi/BWLo return these in place of the BW link.

A non-differential amplifier returns these links:

```
CH<slot><ui> COUPLING:<arg>,
    OFFSET:<NR3>,BW:<NR3>,
    IMPEDANCE:<NR3>,PROBE:<qstring>,
    SENSITIVITY:<NR3>,UNITS:<qstring>
```

A differential amplifier returns these links:

```
CH<slot><ui> MNSCOUPLING:<arg>,
    PLSCOUPLING:<arg>,PROTECT:<arg>,
    OFFSET:<NR3>,AMPOFFSET:<NR3>,
    BW:<NR3>,IMPEDANCE:<NR3>,
    MNSOFFSET:<NR3>,MNSPROBE:<qstring>,
    PLSOFFSET:<NR3>,PLSPROBE:<qstring>,
    SENSITIVITY:<NR3>,UNITS:<qstring>,
    VCOFFSET:<NR3>
```

Examples

CHL?

returns the current vertical parameters for the left plug-in, such as:

```
CHL1 COUPLING:DC,OFFSET:3.0E+0, BW:4.0E+8,IM-
    PEDANCE:1.0E+6,PROBE:"NONE",
    SENSITIVITY:2.0E+0,UNITS:"VOLTS";
    CHL2 COUPLING:DC,OFFSET:0.0E+0,BW:4.0E+8,
    IMPEDANCE:1.0E+6,PROBE:"NONE",
    SENSITIVITY:2.0E+0,UNITS:"VOLTS"
```

AMPoffset

AMPoffset sets the voltage to be subtracted from the input signal, after the plus and minus differential input signals have been subtracted from each other. AMPoffset vertically positions the signal on the display. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>AMPoffset:<NRx>
CH<slot><ui>?<sp>AMPoffset
```

Range: `<NRx>` value is plug-in unit specific.

Examples

`CHR1 AMP:1.0`
sets the voltage to be subtracted from the input signal.

BW

BW sets the channel bandwidth. Out-of-range values are forced to acceptable maximum or minimum values; no warning message is returned.

Syntax:
```
CH<slot><ui><sp>BW:<NRx>
CH<slot><ui>?<sp>BW
```

Range: `<NRx>` value is plug-in unit specific.

Examples

`CHC1 BW:20000000`
sets the bandwidth for channel 1 of the center plug-in unit.

BWHi

BWHi sets the high bandwidth of a channel. This link is only valid for plug-in units with BWHi function. Out-of-range values are forced to acceptable values; no warning message is returned.

Syntax:
```
CH<slot><ui><sp>BWHi:<NRx>
CH<slot><ui>?<sp>BWHi
```

Range: `<NRx>` value is plug-in unit specific.

Examples

`CHC2 BWH:1.0E+9`
sets the high bandwidth for channel 2 of the center plug-in unit.

BWLo

BWLo sets the low bandwidth of a channel. This link is only valid for plug-in units with BWLo function. Out-of-range values are forced to acceptable values; no warning message is returned.

Syntax:

```
CH<slot><ui><sp>BWLo:<NRx>
CH<slot><ui>?<sp>BWLo
```

Range: `<NRx>` value is plug-in unit specific.

Examples

`CHR1 BWL:20E+6`
sets the low bandwidth for channel 1 of the right plug-in unit.

COUpling

COUpling selects the channel input coupling. This applies to nondifferential amplifiers only.

Syntax:

```
CH<slot><ui><sp>COUpling:{AC|DC|OFF|VC}
CH<slot><ui>?<sp>COUpling
```

Examples

`CHL1 COU:DC`
selects DC input coupling for channel 1 of the left plug-in unit.

IMPedance

IMPedance sets the channel input impedance in ohms. Out-of-range values are forced to acceptable values; no warning message is given.

Syntax:
```
CH<slot><ui><sp>IMPedance:<NRx>
CH<slot><ui>?<sp>IMPedance
```

Range: <NRx> = 50, 1E+6, or 1E+9

11A33 Amplifier Note: When PROTect is set to ON or when either MNSCoupling or PLSCoupling is set to AC, 1 GΩ is not allowed. (IMPedance is forced to 1 MΩ.)

Examples

`CHL2 IMP:1E6`
sets impedance for channel 2 of the left plug-in unit.

MNSCoupling

MNSCoupling sets the minus input coupling of the specified channel. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>MNSCoupling:{AC|DC|OFF|
                            VC}
CH<slot><ui>?<sp>MNSCoupling
```

When this link is set to OFF or VC (voltage comparator), the specified minus input is internally disconnected from its external signal source. (Refer to the CH<slot><ui> VCOffset link.)

11A33 Amplifier Note: When MNSCoupling is set to AC, IMPedance is restricted to 50 Ω or 1 MΩ.

Examples

`CHR1 MNSC:AC`
sets the minus input coupling for channel 1 of the right plug-in unit.

MNSOffset

MNSOffset sets the probe offset voltage that will be subtracted from the minus input of the specified channel. This applies to differential amplifiers only.

Syntax: `CH<slot><ui><sp>MNSOffset:<NRx>`
`CH<slot><ui>?<sp>MNSOffset`

Range: `<NRx>` value is plug-in specific.

MNSOffset requires an offset-type Level 2 probe (such as a Tek P6231). If a non-offset-type probe is attached, the MNSOffset value is saved, and applied later when an appropriate probe is connected.

Examples

`CHR1 MNSO -3.4`
sets the probe offset voltage for channel 1 of the right plug-in unit.

MNSProbe

Query Only. MNSProbe returns the type of probe currently connected to the minus input. This applies to differential amplifiers only.

Syntax: `CH<slot><ui>?<sp>MNSProbe`

Query Responses

- **"Level 1"**
- **"Level 2/<probe type>/<serial number>"**
- **"NONE"**

Examples

`CHR1? MNSP`
returns the probe type connected to the minus input, such as:

```
CHR1 MNSPROBE:"LEVEL 1"
```

OFFSet

OFFSet sets input vertical offset.

The differential OFFSet link modifies the AMPoffset, MNSOffset, PLSOffset, or VCOffset links, depending on coupling and probes. Refer to the appropriate plug-in unit *User Reference Supplement* for more information

For nondifferential amplifiers, OFFSet sets the voltage to be subtracted from the input signal, to vertically position the signal on the display.

Syntax:
```
CH<slot><ui><sp>OFFSet:<NRx>
CH<slot><ui>?<sp>OFFSet
```

Range: `<NRx>` is the offset range.

The OFFSet range for 11A32, 11A34, and 11A52 non-differential amplifiers depends on the SENsitivity setting:

OFFSET Range for 11A32, 11A34, & 11A52

SENsitivity Range	OFFSet Range
1E-3 V to 99.5E-3 V	-1 V to +1 V
100E-3 V to 995E-3 V	-10 V to +10 V
1 V to 10 V	-100 V to +100 V

OFFSet range for the 11A71 Non-differential Amplifier uses the SENsitivity value of the appropriate channel (CH<slot> <ui>? SENsitivity):

range = -10 * (SEN) to +10 * (SEN)

OFFSet range for the 11A72 Non-differential Amplifier uses the SENsitivity value of the appropriate channel (CH<slot> <ui>? SEN):

range = -25 * (SEN) to +25 * (SEN)

For the OFFSet range of differential and non-differential amplifiers not mentioned, see the applicable *User Reference Supplement*.

Examples

`CHC1 OFFS:-0.9`
sets the vertical offset for channel 1 of the center plug-in unit.

PLSCoupling

PLSCoupling sets the plus input coupling of the specified channel. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>PLSCoupling:{AC|DC|OFF|
                             VC}
CH<slot><ui>?<sp>PLSCoupling
```

When this link is set to OFF or VC (voltage comparator), the specified plus input is internally disconnected from its external signal source. (Refer to the CH<slot> <ui> VCOffset link.)

11A33 Amplifier Note: When PLSCoupling is set to AC, IMPedance is restricted to 50 Ω or 1 MΩ.

Examples

`CHR1 PLSC:AC`
sets input coupling to AC for channel 1 of the right plug-in.

PLSOffset

PLSOffset sets the probe offset voltage that is subtracted from the plus input of the specified channel. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>PLSOffset:<NRx>
CH<slot><ui>?<sp>PLSOffset
```

PLSOffset requires an offset-type Level 2 probe (such as a Tek P6231). If a non-offset-type probe is attached, the PLSOffset value is saved and applied later when an appropriate probe is connected.

Examples

`CHR1 PLSO:2.1`
sets the probe offset voltage for channel 1 of the right plug-in unit.

PLSProbe

Query Only. PLSProbe returns the type of probe currently connected to the plus input of the channel. This applies to differential amplifiers only.

Syntax: `CH<slot><ui>?<sp>PLSProbe`

Query Responses

- **"Level 1"**
- **"Level 2/<probe type>/<serial number>"**
- **"NONE"**

Examples

`CHR1? PLSP`

returns the current probe type connected to the plus input for channel 1 of the right plug-in unit, such as:

`PLSPROBE:"NONE"`

PROBe

Query Only. PROBe returns the type of probe currently connected to the specified channel. This applies to nondifferential amplifiers only.

Syntax: `CH<slot><ui>?<sp>PROBe`

Query Responses

- **"Level 1"**
- **"Level 2/<probe type>/<serial number>"**
- **"NONE"**

Examples

`CHL1? PROB`

returns the current probe type connected to channel 1 of the left plug-in unit, such as:

`CHL1 PROBE:"LEVEL 2/P6231/B011623"`

PROTect

PROTect restricts the SENsitivity and IMPedance settings of an 11A33 Amplifier. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>PROTect:{OFF|ON}
CH<slot><ui>?<sp>PROTect
```

Arguments

- **OFF** – allows the normal ranges to apply without restrictions.
- **ON** – restricts the SENsitivity range at 100 mV to 10 V and IMPedance is restricted to 50 Ω (active probe) or 1 MΩ (passive probe).

Examples

```
CHR1 PROT:ON
```
restricts sensitivity and impedance.

SENsitivity

SENsitivity sets the channel vertical size.

For the SENsitivity range and step size of other plug-in units, refer to the appropriate *User Reference Supplement*.

Syntax:
```
CH<slot><ui><sp>SENsitivity:<NRx>
CH<slot><ui>?<sp>SENsitivity
```

Range: <NRx> = 1E–3 V to 10 V †
<NRx> = 10E–3 V to 1 V ††

† *Range for 11A32, 11A33, 11A34, 11A52 amplifiers. Refer to the appropriate plug-in unit* User Reference Supplement *for the resolution (step size).*

†† *Range for 11A71 and 11A72 amplifiers in 1-2-5 steps.*

11A33 Amplifier Notes: When PROTect is set to ON, the SENsitivity range is restricted to 100E-3 V to 10 V. (Values below 100 mV are forced to 100 mV.)

When PROTect is set to OFF, neither MNSCoupling or PLSCoupling is set to AC, and SENsitivity is between 100 mV and 10 V; changing IMPedance to 1 GΩ then changes SENsitivity to 99.5 mV.

Examples

```
CHL2 SEN:2
```
sets the vertical size for channel 2 of the left plug-in unit.

UNIts

Query Only. UNIts returns the units of the channel.

Syntax: `CH<slot><ui>?<sp>UNIts`

Returns: `<qstring>`

Examples

```
CHL2? UNI
```
returns the units of the specified channel, such as:
```
CHL2 UNITS:"VOLTS"
```

VCOffset

When either PLSCoupling or MNSCoupling is set to VC, VCOffset sets an internal comparison voltage; VCOffset has no other effect. This applies to differential amplifiers only.

Syntax:
```
CH<slot><ui><sp>VCOffset:<NRx>
CH<slot><ui>?<sp>VCOffset
```

Examples

```
CHR1 VCO:-1.5
```
sets the internal comparison voltage.

CHDIR

CHDIR changes the current working directory on the floppy disk. This command is identical to the CD command.

Syntax: `CHDIR<sp><qstring>`
`CHDIR?`

Range: `<qstring>` specifies the name of the directory.

Query Responses

- **CHDIR?** – returns the current working directory.

Examples

```
CHDIR?
```
returns the current working directory, such as:
```
CHDIR "A:\WAVEFORM"
```

CHKDsk

CHKDsk checks the floppy disk for inconsistencies. If problems can be corrected, CHKDsk will make the corrections.

Syntax: `CHKDsk<sp>"A:"`

Examples

```
CHKD "A:"
```
Checks the floppy disk.

CHSkew

Query Only. CHSkew returns the measured skew (time delay) values in seconds for each channel that is included on a waveform description.

Syntax: `CHSkew?`

Examples

```
CHS?
```
returns a time delay, such as:
```
CHSKEW C1:1.2E-9,L1:-2.3E-10
```

CLEar

Set Only. CLEar discards previously acquired data for ALL displayed waveforms, the specified labeled waveform, or for the specified waveform. (Refer also to the REMove command.)

Syntax: `CLEar<sp>{ALL|<qstring>|TRAce<ui>}`

Range: `<ui>` = 1 to 8, and specifies the waveform.

Arguments

- **ALL** – selects all displayed waveforms. An error is not reported for sending CLEar ALL when no waveforms are defined.
- **<qstring>** – selects a specified labeled waveform. Wildcard characters are valid with <qstring>. (Refer to Label Wildcard Characters on page 182 for wildcard definitions.)
- **TRAce<ui>** – selects a specified waveform.

Examples

`CLE TRA5`
discards all previously acquired data for waveform 5.

COLor

HUE
LIGhtness
SATuration

COLor controls the front panel colors

Syntax:

```
COLor<sp>DEFAult
COLor<ui><sp>{DEFAult|<link>:<arg>}
COLor?
COLor<ui>?[<sp><link>]
```

Range: `<ui>` = 0 to 7, and specifies the color index.

Color Indexes — Original System

<ui>	Color Specified
0	Background
1	Graticule
2	Unselected Main Waveform
3	Selectable Field
4	Selected Main Waveform
5	Unselected Window Waveform
6	Selected Window Waveform
7	Cursors and Measurement Bars

Color Indexes — Standard System

<ui>	Color Specified
0	Background
1	Waveform 1
2	Waveform 2
3	Waveform 3
4	Waveform 4
5	Window Waveforms
6	Graticule and Selectors
7	Cursors and Measurement Annotation

Arguments

- **DEFAult**—sets the factory default hue, lightness, and saturation for one or all colors.

Note: Refer to the *Tektronix Color Standard HLS* coordinate system for the definitions of hue, saturation, and lightness.

Examples

`COL1 DEFA`
sets Waveform Color1 to the factory default settings.

`COL DEFA`
sets all front panel colors to the factory default settings.

HUE

HUE sets the hue of the specified color.

Syntax:
```
COLor<ui><sp>HUE:<NRx>
COLor<ui>?<sp>HUE
```

Range: `<NRx>` = 0 to 360 degrees

Examples

`COL1 HUE:120`
sets the hue for the first waveform color.

LIGhtness

LIGhtness sets the lightness of the specified color

Syntax:
```
COLor<ui><sp><LIGhtness>:<NRx>
COLor<ui>?<sp><LIGhtness>
```

Range: `<NRx>` = 0 to 100 percent

Examples

`COL1 LIG:30`
sets the lightness for the first waveform color.

SATuration

SATuration sets the saturation of the specified color

Syntax:
```
COLor<ui><sp><SATuration>:<NRx>
COLor<ui>?<sp><SATuration>
```

Range: `<NRx>` = 0 to 100 percent

Examples

```
COL1 SAT:80
```
sets the saturation for the first waveform color.

COLORMap

SYStem
TRAce

The COLORMap command selects the display color system (the color model).

Syntax:
```
COLORMap<sp><link>:<arg>
COLORMap? [<sp><link>]
```

SYStem

SYStem defines the color system.

Syntax:
```
COLORMap<sp>SYStem:{ORIginal|STANdard}
COLORMap?<sp>SYStem
```

Arguments

- **ORIginal**—assigns colors on a functional basis. That is, the selected main waveform has a different color from unselected waveforms, and the selected window waveform has a different color from unselected window waveforms.
- **STANdard**—assigns colors on a waveform basis and the selected waveform is brightened on the screen. The TRAce<*ui*> link, described below, assigns colors to waveforms in this system.

Examples

```
COLORM SYS:STAN
```

TRAce

TRAce assigns a color to the specified trace. The four available colors have numbers 1, 2, 3, and 4. Any of these colors may be assigned to any of the eight possible traces.

Note: *Window traces have the color number 5. You cannot change the color number for a window trace.*

Syntax:
```
COLORMap<sp>TRAce<ui>:COLor<ui>}
COLORMap?<sp>TRAce<ui>
```

Range: `COLor<ui>` = 1 to 4

Examples

```
COLORM TRA1:COL4
```

COMpare

COMpare controls the measurement comparison mode.

Syntax:
```
COMpare<sp>{OFF|ON}
COMpare?
```

Arguments

- **OFF** – turns comparison mode off. A measurement query returns the value of the measurement followed by an accuracy qualifier. COMpare OFF is the normal measurement mode.
- **ON** – turns comparison mode on. A measurement query compares the measurement value with a reference value set with the REFset command, and then returns the difference with an accuracy qualifier. If the reference measurement is undefined or the measurement qualifier is UN (uncertain), the returned comparison qualifier is also UN.

Note: For the list of measurement accuracy qualifiers and their definitions, refer to page 193.

Examples

```
COM ON
```
turns measurement comparison mode on.

CONDacq

FILl
REMAining
TRIgger
TYPe

CONDacq sets the following conditions for waveform acquisition: completion of a specified condition, continuous acquisition, or acquisition on a specified number of triggers.

Completion of any conditional acquisition TYPE (i.e., all types except CONTInuous) is signaled by event code 450, "Conditional acquire complete."

Syntax:
```
CONDacq<sp><link>:<arg>
CONDacq? [<sp><link>]
```

Examples

`COND?`
returns acquisition parameters, such as:
```
CONDACQ FILL:99,REMAINING:0,TRIGGER:MAIN,
   TYPE:CONTINUOUS.
```

FILl

FILl sets the percentage of waveform record completion for CONDacq TYPe:FILl.

Syntax:
```
CONDacq<sp>FILl:<NRx>
CONDacq?<sp>FILl
```

Range: `<NRx>` = 1 to 100 percent

Examples

`COND FIL:80`
sets the percentage of waveform record needed to stop acquisition.

REMAining

Query Only. REMAining returns a value indicating how much of the selected acquisition TYPe must still be acquired to complete acquisition.

Syntax: `CONDacq?<sp>REMAining`

Returns: `<NR1>`

For each acquisition type, the response is:

- AVG – number of averages remaining.
- BOTh – number of averages and envelopes remaining.
- CONTInuous – is not meaningful; returns 0.
- DELTa – is not meaningful; returns 0.
- ENV – number of envelopes remaining.
- FILl – percentage of fill remaining.
- HIST.pt – number of points remaining.
- REPtrig – number of repetitive triggers remaining in count. Not meaningful when AUTOAcq MEMWrap is set to ON.
- SEQuence – is not meaningful; returns 0.
- SINgle – is not meaningful; returns 0.
- WAVfrm – number of complete trace records remaining.

Note: When conditional acquisition is complete and the digitizer has stopped, the REMAining query always returns 0 (zero).

Examples

`COND? REMA`
returns how much of the specified acquisition type has not been acquired, such as:

```
CONDACQ REMAINING:22
```

TRIgger

TRIgger selects the trigger used when TYPe is set to SINgle, SEQuence, or REPtrig.

Syntax:
```
CONDacq<sp>TRIgger:{MAIn|WINdow}
CONDacq?<sp>TRIgger
```

Examples

```
COND TRI:WIN
```

TYPe

TYPe selects the acquisition type.

Syntax:
```
CONDacq<sp>TYPe:{AVG|BOTh|CONTInuous|
                 DELTa|ENV|FIL1|HIST.pt|
                 REPtrig|SEQuence|SINgle|
                 WAVfrm}
CONDacq?<sp>TYPe
```

Arguments

- **AVG** – acquires NAVg number of averages for all waveforms that include AVG in their description.
- **BOTh** – acquires NAVg number of averages or NENv number of envelopes for all waveforms that include either AVG and ENV in their description.
- **CONTInuous** – acquires continuously until halted with DIGitizer STOP.

- **DELTa** – acquires until the delta condition is met. Needs DIGitizer RUN to start acquisition.
- **ENV** – acquires NENv number of envelopes for all waveforms that include ENV in their description.
- **FILl** – acquires a waveform record to the percentage set by CONDacq FILl.
- **HIST.pt** – acquires until NHIST.pt waveform points are in the histogram for the selected waveform.
- **REPtrig** – acquires and stores NREP number of waveforms. Each acquisition requires a valid trigger. (Use DIGitizer RUN to start acquisition.)
- **SEQuence** – acquires on a single trigger for all defined waveforms. Use DIGitizer RUN to start acquisition.
- **SINgle** – acquires on a single trigger from the selected time base. Use DIGitizer RUN to start acquisition.
- **WAVfrm** – acquires NWAVfrm number of complete waveform records for the selected waveform.

Note: Setting TYPe to AVG, BOTh, CONTInuous, ENV, or FILl starts acquisition.

Examples

```
COND TYP:ENV
```
acquires a specified number of envelopes.

CONFig

Query Only. CONFig returns information on which types of plug-in units are installed. If a compartment is empty, CONFig? returns "N/7K".

Syntax: `CONFig?`

Examples

`CONF?`

returns the type of plug-in units installed, such as:

```
CONFIG LEFT:"11A34V",CENTER:"11A71",
   RIGHT:"N/7K"
```

COPy

FORMat
PRInter

COPy sends a copy of the front panel display to the port specified in the appropriate printer command.

Syntax:
```
COPy [<sp>{ABORT|STARt|<link>:<arg>}]
COPy?<sp>{STAtus|<link>}
```

Arguments

- **ABOrt** – terminates the hardcopy output in process and clears the queue of copy requests.
- **STARt** – initiates a front panel copy, spooling the data into memory even if another copy request is printing or spooling.

Note: If you enter COPy with no argument when no other copy request is printing or spooling, a copy is started. However, if a copy request is spooling, entering COPy aborts the spooling copy and does not initiate a copy.

Query Responses

- **COPy? STAtus** – returns the printing status of front-panel copies. IDLe means no copies are printing or spooling; ABORTIng, PRINting, and SPOoling are self-explanatory.

Examples

`COP ABO`
stops the hardcopy output.

`COP? STA`
returns the printing status, such as:

```
COPY STATUS:IDLE
```

FORMat

FORMat selects the output format for the currently selected printer.

Syntax:

```
COPy<sp>FORMat:{DIThered|DRAft|HIRes|
                REDuced|SCReen}
COPy?<sp>FORMat
```

Arguments

- **DIThered** – improves print contrast for Tektronix 4692, Tektronix 4696, and Tektronix 4697 printers by reducing saturation for icon and text backgrounds.
- **HIRes** – improves contrast for monochrome printers with limited gray-scale capability by dithering icon and text backgrounds and increasing saturation of the foregrounds.
- **DRAft** – prints black-on-white background except for selected icons or text, which are printed white-on-black background.
- **REDuced** – is a quarter-size version of DRAft and prints black-on-white background only.
- **SCReen** – is a one-to-one mapping of 3-bit pixel information. (Refer to the BITMap command.)

Note: The COPy FORMat link is included for compatibility with the 11401 and 11402 Mainframes. For new applications, use the FORMat link of the appropriate printer command.

Examples

```
COP FORM:HIR
```

uses dithering and increased saturation for improved print contrast.

PRInter

PRInter selects the target printer. Refer to the individual printer commands to select the printer parameters.

Syntax:

```
COPy<sp>PRInter:{ALTinkjet|
                 BITMap|HPGl|PIN8|
                 PIN24|TEK4692|
                 TEK4696|TEK4697}
COPy?<sp>PRInter
```

Examples

`COP PRI:TEK4696`
selects a Tektronix 4696 printer as the output device.

CROss

Query Only. CROss returns the time from the trigger point to a specified reference level crossing, followed by an accuracy qualifier. (Refer to page 193 for qualifier definitions.) The reference level is set with the REFLevel command. The crossing slope is set with the MSLOpe command. Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `CROss?`

Returns: `<NR3>` or <bblock>

Examples

`CRO?`
returns the time from the trigger point to a reference level crossing, such as:

```
CROSS 6.9284065E-8,EQ
```

CURMode

CMOde
DEFAult
HOLd

CURMode sets the cursor type for all traces.

Syntax:
```
CURMode<sp><link>:<arg>
CURMode?[<sp><link>]
```

CMOde

CMOde determines the coarse step interval for paired or split dot cursors.

Syntax:
```
CURMode<sp>CMO:{FINTerval|MPEak|PMPeak|
                         PPEak}
CURMode?<sp>CMO
```

Arguments

- **FINTerval**—the cursor moves at fixed intervals for each knob click. This is the standard mode of operation.
- **MPEak**—the cursor moves to the next minus peak with each knob click.
- **PMPeak**—the cursor moves to the next plus or minus peak with each knob click.
- **PPEak**—the cursor moves to the next plus peak with each knob click.

Examples

`CURM CMO:PMP`
Sets the cursor to move to the next plus or minus peak with each knob click.

DEFAult

DEFAult sets the default cursor type.

Syntax:
```
CURMode<sp>DEFAult:{HBArs|PAIred|SPLit|
                         VBArs}
CURMode?<sp>DEFAult
```

Arguments

- **HBArs** – are horizontal bar cursors.
- **PAIred** – cursors are two dots that appear on one trace.
- **SPLit** – cursors are two dots that appear on one or more traces.
- **VBArs** – are vertical bar cursors.

Examples

`CURM DEFA:VBA`
sets the default cursor type to vertical bars.

HOLd

HOLd determines whether the cursors will remain displayed when the cursor menu is exited. If HOLD is set to ON, the cursors remain on the screen until they are removed.

Syntax:

```
CURMode<sp>HOLd:{OFF|ON}
CURMode?<sp>HOLd
```

Examples

`CURM HOL:OFF`
cursors will not be displayed when the cursor menu is not displayed.

CURSor

READout
REFErence
TYPe
XUNit
YUNit

CURSor sets cursor operating characteristics for the selected trace, such as the cursor type (dot or bar), the reference cursor, and whether front panel readouts are displayed.

Syntax:
```
CURSor<sp><link>:<arg>
CURSor?[<sp><link>]
```

READout

READout controls whether front panel cursors and their corresponding knob readouts are displayed and active from the front panel.

Syntax:
```
CURSor<sp>READout:{OFF|ON}
CURSor?<sp>READout
```

Arguments

- **OFF**—turns off the display of the cursors and their values in the cursors menu. However, cursors can be set or queried with remote commands regardless of READout setting.
- **ON**—turns on the display of the cursors and their values in the cursors menu.

Note: When FPUpdate is set to NEVer, setting CURSor READout to ON displays the cursors, but not their readouts.

Examples

`CURS REA:ON`
activates front panel cursors and knob readouts.

REFErence

REFErence selects the reference waveform for split cursors.

Syntax:
```
CURSor<sp>REFErence:TRAce<ui>
CURSor?<sp>REFErence
```

Range: `<ui>` = 1 to 8, and specifies the waveform.

When the specified REFErence waveform is not the selected waveform, the CURSor TYPe is automatically set to SPLit. When the CURSor TYPe is set to PAIred, the REFErence waveform is set to the selected waveform. A newly-created waveform uses itself for the default REFErence for .

XY Note: You cannot change the REFErence waveform for an XY waveform.

Note: It is not an error if you specify a REFErence waveform that is not yet defined. The REFErence waveform is only checked when CURSor REAdout is set ON or at a DOT2Abs? query. If the REFErence waveform is then undefined, it is changed to the selected waveform.

Examples

`CURS REFE:TRA5`
selects waveform 5 as the reference waveform.

TYPe

TYPe selects the cursor type. Setting the TYPe to PAIRed automatically sets the REFErence waveform to the selected waveform.

Syntax:
```
CURSor<sp>TYPe:{HBArs|PAIred|SPLit|VBArs}
CURSor?<sp>TYPe
```

Arguments

- **HBArs**—horizontal bar cursors.
- **PAIred**—cursors are two dots that appear on one trace.
- **SPLit**—cursors are two dots that appear on one or more traces.
- **VBArs**—vertical bar cursors.

XY Note: SPLit cursors are not permitted on XY waveforms.

Examples

```
CURS TYP:VBA
```
selects vertical bar cursors.

XUNit

Query Only. XUNit returns the horizontal units of the selected waveform.

Syntax: `CURSor?<sp>XUNit`

Query Responses

Possible units are AMPS, DB, DEGrees, DIVs, HERtz, OHMs, SEConds, VOLts, and WATts.

Examples

```
CURS? XUN
```
returns the selected waveform's horizontal units, such as:

```
CURSOR XUNIT:SECONDS
```

YUNit

Query Only. YUNit returns the vertical units of the selected waveform.

Syntax: `CURSor?<sp>YUNit`

Query Responses

Possible units are AMPS, DB, DEGrees, DIVs, HERtz, OHMs, SEConds, VOLts, and WATts.

Examples

```
CURS? YUN
```
returns the selected waveform's vertical units, such as:

```
CURSOR YUNIT:VOLTS
```

CURVe

CURVe transfers unscaled waveform data to and from the controller in binary or ASCII format. Each waveform that is transferred has an associated waveform preamble that contains information such as scaling factors and the number of data points transferred. Refer to the WFMpre command for the waveform preamble.

Syntax:
```
CURVe<sp><Curve data>
CURVe?
```

The query form retrieves data from the DSA. The data source is specified by the OUTput command. The entire CURVe? response can be sent back to the DSA as a set command.

The set form sends data to the DSA from the controller. An incoming waveform is always stored; it is never active or acquired. The STO (store) location for the data is specified by the INPut command. The power-on default INPut location is STO1.

<Curve data> can be in ASCII (<asc curve>) or binary (<bblock>) format. The format is set by the ENCdg WAVfrm command.

ASCII Transfer. Data transferred as an <asc curve> use the following format:

```
<asc curve> ::= <NR1> [ ,<NR1> ] ...
```

where <NR1> values are data points within the range -32768 to +32767.

For most YT waveforms, each <NR1> value represents one data point in the waveform record. For enveloped YT waveforms, every two <NR1> values represent one max/min pair in the waveform record. For XY waveforms, every two consecutive <NR1> values represent one X,Y coordinate pair in the waveform record. (The X-coordinate is the first point in the pair.)

Binary Transfer. Data is transferred as comma-separated binary blocks in the format:

```
<bblock>[,<bblock>]
```

where:

```
<bblock> ::= %<byte cnt><bin pt>...<checksum>
```

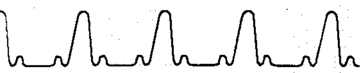
where <byte cnt> is a two-byte binary integer (MSB first) giving the length in bytes of the remainder of the binary block, including checksum; <bin pt> is a two-byte binary data point in the range -32768 to +32767; <checksum> is an 8-bit, twos complement of the modulo 256 sum of <byte cnt> and all <bin pt> data.

The transmission order for data points is set by the BYT.or command. There are no separators (such as commas) between data points. There is only one binary block returned for record lengths 512-20464. For 323768 length records, two comma-separated binary blocks are returned.

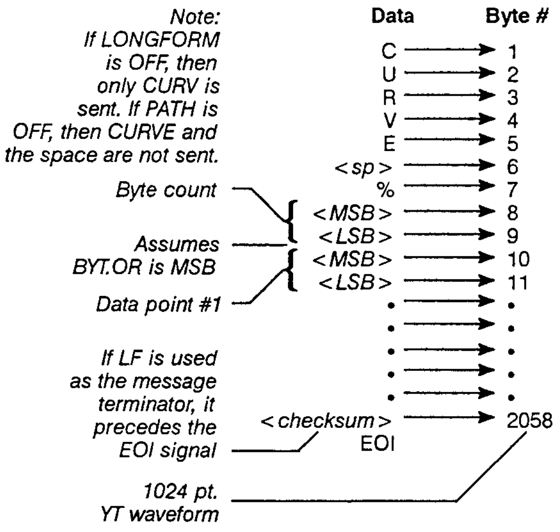
Predefined CURVe? Data Values. The following data point values are predefined for CURVe?:

Predefined CURVE? Data Values

Data Value 16-bit	Data Value 8-bit	Meaning
+32767	+255	Vertical Overrange. Data point is high off-screen and cannot be displayed with current scaling parameters.
-32767	-255	Vertical Underrange. Data point is low off-screen and cannot be displayed with current scaling parameters.
-32768	-256	Null Data. Data point that has not been acquired.



The figure below illustrates binary data transfer:


Note: If LONGFORM is OFF, then only CURV is sent. If PATH is OFF, then CURVE and the space are not sent.
Byte count
Assumes BYT.OR is MSB
Data point #1
If LF is used as the message terminator, it precedes the EOI signal
1024 pt. YT waveform
Data
Byte #
C 1
U 2
R 3
V 4
E 5
<sp> 6
% 7
<MSB> 8
<LSB> 9
<MSB> 10
<LSB> 11
<checksum> 2058
EOI

Binary Data Transfer

Waveform Scaling. CURVe transfers unscaled waveform data which must be scaled in order to be analyzed. The following formulas use values from the waveform preamble (see the WFMpre command) to scale the coordinate values of each point transferred.

There are two scaling formulas for **YT** waveforms:

$$X_n = \text{XZEro} + \text{XINcr} * n$$
$$Y_n = \text{YZEro} + \text{YMUlt} * \text{data\_pt\_n}$$

where X*n* is the scaled horizontal coordinate of the *n*th data point in XUNits; Y*n* is the scaled vertical coordinate of the *n*th data point in YUNits; XZEro, XINcr, YZEro, and YMUlt are values from the WFMpre command; *n* is the sequence number of the *n*th retrieved data point (range is 0 to WFMpre NR.pt - 1); data_pt_n is the value of the *n*th unscaled point (as retrieved by CURVe?).

There are two scaling formulas for **XY** waveforms:

$$X\mathit{n} = \text{XZEro} + \text{XMUlt} * \text{data\_pt\_nx}$$
$$Y\mathit{n} = \text{YZEro} + \text{YMUlt} * \text{data\_pt\_ny}$$

where X*n* is the scaled X-coordinate of the *n*th unscaled X,Y pair in XUNits; Y*n* is the scaled Y-coordinate of the *n*th unscaled X,Y pair in YUNITs; XZEro, XMUlt, YZEro, and YMUlt are values from the WFMpre command; data_pt_nx is the value of the *n*th unscaled X-coordinate (as retrieved by CURVe?); data_pt_ny is the value of the *n*th unscaled Y-coordinate.

Sending a Waveform Without a Preamble. It is possible to send a waveform to the DSA without supplying a preamble. If a stored waveform exists at the INPut STO location, it is overwritten and its preamble is used with the new waveform. If no stored waveform exists at the INPut STO location, the following default preamble is used with the new waveform:

Default Preamble Parameters

<link>:	**<arg>**	**<link>:**	**<arg>**
ACState:	ENHanced	YUNit:	VOLts
NR.pt:	1024	YZEro:	0.0
PT.fmt:	Y	LABel:	" " (null)
XINcr:	5.0E-7	TIMe:	-current-
XUNit:	seconds	DATE:	-current-
XZEro:	0.0	TSTIME:	0.0
YMUlt:	1.5625E-4		

These are the power-on default values. When any of these links are modified (set) with the WFMpre command, the new values are used.

The following example is an excerpt from an ASCII-formatted data transfer. (The shortest data transfer contains 11 points when ADJ<ui> HMAG is set to 50 and the record length is set 512.)

Examples

```
CURV?
```

returns curve data; a partial listing would look like:

```
CURVE 4022,3130,2756,1297,709,1073,822,685,111
2,777,1666,2249,3615,4180,4231,4113,988,-2241,
-5609,-128,-3076,-9924,-8434,-8112,...
```

DAInt

DAInt sets the data measurement interval.

Syntax:
```
DAInt<sp>{SINgle|WHOle}
DAInt?
```

Arguments

- **SINgle** – sets the interval to a single period of the waveform.
- **WHOle** – selects the measurement interval set by the LMZone and RMZone commands.

DAInt affects the MEAN?, RMS?, YTEnergy?, YTMns_area?, and YTPls_area? measurements. These measurements return an ER qualifier if DAInt is set to SINgle and no period can be found.

Note: The measurement qualifiers are defined on page 192.

Examples

`DAI SIN`
selects a single period for the data measurement interval.

DATE

DATE sets the date on the internal calendar.

Syntax:
```
DATE<sp><qstring>
DATE?
```

Range: The format of `<qstring>` is <dd-mon-yy>
dd is the day of the month,
mon is the first three letters of the month, and
yy is the last two digits of the year.

Examples

`DATE '03-SEP-90'`
sets the date.

DCOpy

<qstring>
FPS<ui>
STO<ui>

Set only. DCOpy copies waveforms and settings from one location to another.

Syntax: `DCOpy<sp><link>:<arg>`

<qstring>

<qstring> copies the specified file to either a stored waveform location (STO<*ui*> in RAM), a stored setting location (FPS<*ui*> in RAM), or another file. The current directory will be used, unless a full pathname is given.

Syntax:
```
DCOPY<sp><qstring>:{FPS<ui>|STO<ui>}
DCOPY<sp><qstring>,<qstring>
```

Range:
```
STO<ui> = 1-420† or 1-918‡
FPS<ui> = 1-20
```

†*Without the disk drive, the range is 1-453.*

‡*With Option 4C (NVRAM) installed.*

Examples

`DCO "A:\WAVEFORM\STO12.WFB":STO1`
copies the waveform STO12.WFB (on the disk) into STO1 (in RAM).

FPS<ui>

FPS<ui> copies the specified stored setting (in RAM) to a file on the disk. The current directory will be used, unless a full pathname is given.

Syntax: `DCOPY<sp>FPS<ui>:<qstring>`

Range: `FPS<ui> = 1-20`

Examples

`DCO FPS1:"A:\FPSETS\FPSET1.FPB"`
copies the stored setting in FPS1 to a file on the disk in the FPSETS directory.

STO<ui>

STO<ui> copies the specified stored waveform (in RAM) to a file on the disk. The current directory will be used, unless a full pathname is given.

Syntax: `DCOPY<sp>STO<ui>:<qstring>`

Range: `STO<ui> = 1-420† or 1-918†`

†Without the disk drive, the range is 1-453.

†With Option 4C (NVRAM) installed.

Examples

```
DCO STO11:"A:\WAVES\ELEVEN.WFA"
```

copies the stored setting in STO11 to a file on the disk in the WAVES directory.

DEBug

GPIb
RS232

DEBug copies input data from the specified interface to the front panel display for program development troubleshooting. The incoming ASCII commands are displayed on the top four lines of the screen.

Syntax:
```
DEBug<sp><link>:<arg>
DEBug? [<sp><link>]
```

Note: Setting DEBug to ON for either interface slows system throughput considerably.

GPIb

GPIb sets DEBug to ON or OFF for the GPIb interface.

Syntax:
```
DEBug<sp>GPIb:{OFF|ON}
DEBug?<sp>GPIb
```

Examples

`DEB GPI:OFF`
turns debug mode off for the GPIB interface.

RS232

RS232 sets DEBug to ON or OFF for the RS-232-C interface.

Syntax:
```
DEBug<sp>RS232:{OFF|ON}
DEBug?<sp>RS232
```

Examples

`DEB RS232:ON`
turns debug mode on for the RS-232-C interface.

DEF

DEF defines a logical name to substitute for a DSA command string.

Syntax: `DEF<sp><qstring>,<qstring>`
`DEF?`

Range: The first `<qstring>` is the logical name; the second `<qstring>` is the command string that is executed.

DEF Usage. Here are some rules and suggestions for using DEF:

- The first character of the logical name must be alphabetic. Case is ignored.
- You cannot use logical names as <qstring> input for other commands.
- You cannot have a command string that is null, such as ''. Also, the first character of an expansion string cannot be any of the following six characters:

Restricted Expansion String Characters

Character	Character
colon (:)	space (octal 40)
comma (,)	linefeed (octal 12)
semicolon (;)	carriage return (octal 15)

- You can define a short name for a group of concatenated commands, or you can rename a command to one or two letters. However, do not redefine the single characters **L**, **C**, or **R**. These characters represent the plug-in compartments in various commands. If L, C, or R are redefined, the commands that contain them will always return a syntax error.
- Recursive DEF logical names are acceptable only when recursion occurs to the right of an unquoted semicolon. All other recursive definitions are illegal.

Acceptable and Illegal Recursion

Acceptable Recursion	Illegal Recursion
DEF 'z','TBMain?;z'	DEF 'z','z?'
DEF 'j','ABStouch 3,10;j'	DEF 'j','TEXt j'

Note: A valid recursive logical name causes an infinite command processing loop. Thus, once a recursive logical name is transmitted, the DSA will not respond to command input until a DCL (Device Clear) signal is sent to the port that received the recursive logical name. (Refer also to the FEOi command.)

Note: Logical names and expansion strings are not stored in nonvolatile RAM. Therefore, they are lost when the DSA is powered off.

Predefined Logical Names. Each time the DSA is turned on, the following two logical names are automatically placed in the definition table:

Predefined Logical Names

Logical Name	Expansion String
e	RS232 ECHo:ON
v	RS232 VERBose:ON

Examples

```
DEF 'TB?','TBM?;TBW?'
```
defines a new query that is the concatenation of the TBMain and TBWin queries.

Once the logical name has been defined with DEF, you enter the logical name without quotes as in other commands.

```
DEF?
```
returns a list of all commands defined by DEF and their expansions.

DELAy

Query Only. DELAy returns the time between the first and last MESial crossing of a waveform within the measurement zone, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `DELAy?`

Returns: `<NRx> or <bblock>`

Examples

```
DELA?
```

returns the time between the first and last MESial crossing, such as:

```
DELAY 1.954E-6,EQ
```

DELete

ALL

Set Only. DELete removes stored front panel settings and stored waveforms from memory or disk.

Syntax: `DELete<sp>{<link>:<arg>|<qstring>}`

Arguments

- **<qstring>** – deletes the stored waveform, or front panel setting, that matches the label. If <qstring> starts with "A:" then the file is assumed to be on the disk. If a full path is not specified, the current working directory is searched and files that match the <qstring> are deleted. Wildcard characters are interpreted; refer to page 181 for wildcard definitions. If the label matches both a stored waveform and a front panel setting, the stored waveform is deleted. To delete the labeled front panel setting, you must send DELete <qstring> again.

Note: You cannot delete a stored waveform that is a combined component of an active waveform. (However, you can delete a stored waveform if it is the only active waveform.)

Examples

```
DEL STO150
```
deletes the stored waveform 150.

Range: `FPS<ui>` = 1 to 20, and specifies the front panel setting.
`STO<ui>` = `1-420†` or `1-918†`

†Without the disk drive, the range is 1-453.

†With Option 4C (NVRAM) installed.

ALL

Deletes all stored front panel settings or all stored waveforms from memory or disk. If SETDev is set to RAM, then files in memory are deleted. If SETDev is set to DISK, then files on the disk are deleted. See SETDev for further details.

Syntax: `DELete<sp>ALL:{FPS|STO}`

Arguments

- **FPS** – deletes all stored front panel settings.

- **STO**—deletes all stored waveforms.

Note: It is not an error to issue DELete ALL:FPS or DELete ALL:STO when no settings or waveforms are stored.

Examples

`DEL ALL:FPS`
deletes all stored front panel settings.

DELTa

CHIme
CONSecpts
COPy
DEScription
EVENTS
NUMPts
REPeat
SAVe
SHOWPts
SRQ
STAtus
TOTalpts

DELTa compares a displayed (test) waveform against an enveloped reference waveform. If specified conditions are met (e.g., the required number of points occur outside the reference envelope), a delta event occurs and specified actions are performed. Possible actions include sounding a beep, making a hardcopy of the display, signaling the GPIB SRQ line, or saving the acquired waveform as a stored waveform.

Syntax:
```
DELTa<sp><link>:<arg>
DELTa?[<sp><link>]
```

CHIme

CHIme determines whether the DSA beeps when a delta event occurs.

Syntax:
```
DELTa<sp>CHIme:{OFF|ON}
DELTa?<sp>CHIme
```

Examples

```
DELT CHI:OFF
```

CONSecpts

CONSecpts selects the number of consecutive points of the test waveform that fall outside the reference waveform envelope that must be acquired for a delta event to occur. Both CONSecpts and DELTa TOTalpts must be satisfied for the event to occur.

Syntax:
```
DELTa<sp>CONSecpts:<NRx>
DELTa?<sp>CONSecpts
```

Range: `<NRx>` = 1 to the record length of the test waveform.

Examples

```
DELT CONS:10
```

COPy

COPy selects whether a hardcopy of the current display and menus is spooled to the printer when a delta event occurs. If DELTa COPy and DELTa REPeat are both set to ON, the digitizer is re-armed before the copy is spooled. However, subsequent delta events will not result in a hardcopy until the previous hard-copy has finished spooling.

Syntax:
```
DELTa<sp>COPy:{OFF|ON}
DELTa?<sp>COPy
```

Examples

```
DELT COP:ON
```

DEScription

DEScription defines the delta comparison.

Syntax: `DELTa<sp>DEScription:<qstring>`
`DELTa?<sp>DEScription`

Range: `<qstring>` is in the form:

WFM<*ui*> OUTSIDE {WFM<*ui*>|STO<*ui*>}

where WFM<*ui*> is a defined waveform (normally referred to in the form TRAce<*ui*>); and OUTSIDE is the keyword for delta comparison. The first WFM<*ui*> is the test trace and the second WFM<*ui*> or STO<*ui*> is the reference waveform.

Examples

```
DELT DES:'WFM6 OUTSIDE STO55'
```

EVENTS

Query only.EVENTS returns the number of delta events that have occured. The value is reset each time the digitzer is started. When REPeat is set to ON, EVENTS returns the number of delta events that have occured since the digitzer was started.

Syntax: `DELTa?<sp>EVENTS`

Examples

```
DELT? EVENTS
```

returns the number of delta events, such as,

```
DELTA EVENTS:17
```

NUMPts

Query only. NUMPts returns the number of points outside the reference waveform.

Syntax: `DELTa?<sp>NUMPts`

Examples

```
DELT? NUMP
```

returns the number of points outside the reference, such as,

```
DELTA NUMPTS:117
```

REPeat

REPeat selects whether the DSA halts after the first delta event or if it performs the specified action(s) and re-arms the digitizer. If REPeat is set to ON, the DSA continues to test for delta conditions until REPeat is set OFF or it receives DIGitizer STOP, or the front panel **DIGITIZER** button is pressed.

Syntax:
```
DELTa<sp>REPeat:{OFF|ON}
DELTa?<sp>REPeat
```

Examples

```
DELT REP:OFF
```

SAVe

SAVe selects whether to save the acquisition that caused the delta event as a stored waveform. The waveforms specified by AUTOAcq are stored. Up to eight waveforms can be stored (See AUTOAcq). If SAVe is set to ON, the waveforms are labeled using a base label and an index with a time and date stamp. (Refer to the LABel command.)

Syntax:
```
DELTa<sp>SAVe:{OFF|ON}
DELTa?<sp>SAVe
```

Examples

```
DELT SAV:ON
```

SHOWPts

SHOWPts determines which waveform is used for producing the delta points display.

Syntax:
```
DELTa<sp>SHOWPts:{SELECTEd|TESt}
DELTa?<sp>SHOWPts
```

Arguments

- SELECTEed – specifies that points on the selected waveform that fall outside the template waveform are displayed in a different color.
- TESt – specifies that points on the test waveform (defined by DELTa DEScription) that fall outside the template waveform are displayed in a different color.

Examples

```
DELT SHOWP:SELECTE
```

SRQ

SRQ selects whether the SRQ line is signaled for each delta event when DELTa REPeat is set to ON. SRQMASK OPCMPL must be set to ON for SRQ to be transmitted.

Syntax:
```
DELTa<sp>SRQ:{OFF|ON}
DELTa?<sp>SRQ
```

Examples

```
DELT SRQ:OFF
```

STAtus

STAtus selects the information that is displayed in the Act on Delta selector of the Waveform major menu.

Syntax:
```
DELTa<sp>STAtus:{ACTions|NUMPts}
DELTa?<sp>STAtus
```

Arguments

- ACTions – specifies that the currently selected delta actions will appear in the status area.
- NUMPts – specifies that the number of points that fell outside the template waveform will appear in the status area.

Examples

```
DELT STA:ACT
```

TOTalpts

TOTalpts specifies the total number of points that must be outside the envelope; both CONSecpts and TOTalpts must be satisfied for the delta event to occur.

Syntax:
```
DELTa<sp>TOTalpts:<NRx>
DELTa?<sp>TOTalpts
```

Range: `<NRx>` = 1 to the record length of the test waveform.

Examples

```
DELT TOT:15
```

DIAg

Query Only. DIAg returns pass/fail information from Self-tests Diagnostics or Extended Diagnostics. Power-on Diagnostics are always performed unless bypassed with hardware jumpers.

Syntax: `DIAg?`

Query Responses

- **DIAg PASsed:"NONe"**—means that all tests were run and all passed.
- **DIAg PASsed:"<omitted test>"**—means that all tests were run and all passed. <omitted test> is a comma-delimited list of tests that were not performed because of missing (optional) hardware.
- **DIAg FAIled:"<failed test>"**—means that one or all of the tests failed. <failed test> is a comma-delimited list of the tests that failed diagnostics.
- **DIAg FAIled:"<omitted test>"**—means that one or all of the tests failed. <omitted test> is a comma-delimited list of tests that were not performed because of missing (optional) hardware.
- **DIAg BYPassed**—means power-on Self-test Diagnostics were bypassed with hardware jumpers.

Note: The DIAg? FAIled response can include both failed and omitted tests.

Refer to the *DSA 601A and DSA 602A Service Reference* for information on the syntax and meaning of omitted tests and failed tests.

Examples

`DIA?`
returns pass/fail information, such as:
`DIAG FAILED:"DI62X,DI22X,R????"` where `DI62X` and `DI22X` are failed tests and `R????` is an omitted test.

DIGitizer

DIGitizer starts and stops waveform acquisition (digitizing). At least one waveform must be defined and at least one component must be acquired. Both the RUN and ARMed arguments enable waveform acquisition. A DIGitizer? query returns ARMEd, if CONDacq TYPe is set to SINgle, SEQuence, or REPTrig, and the DSA has received a DIGitizer RUN or DIGitizer ARMed command, but has not yet received a trigger signal to begin acquisition.

Syntax:
```
DIGitizer<sp>{ARMed|RUN|STOP}
DIGitizer?
```

Arguments

- **ARMed** – digitizer is armed.
- **RUN** – digitizer running.
- **STOP** – digitizer stopped.

Examples

```
DIG RUN
```
starts waveform acquisition.

DIR

Query Only. DIR displays a list of files and directories in the current working directory on the floppy disk.

Syntax: `DIR?`

Returns: a directory listing of the current stored waveforms as a list of comma separated stings in the following format:
```
"FILE.EXT SIZE READ/WRITE  TIME DATE ",
"FILE.EXT...
```

Examples

```
DIR?
   "ELEVEN.WFA 2342 +r  12:49:06 12-MAY-91 ",
   "SETS <DIR> 13:31:23 29-JAN-91"
```

DISPlay

GRAticule
INTensity
INTERPolation
MODe
PERSistence

DISPlay sets the number of graticules, the display intensity, and the display mode.

Syntax:
```
DISPlay<sp><link>:<arg>
DISPlay?[<sp><link>]
```

GRAticule

GRAticule selects dual or single graticules.

Syntax:
```
DISPlay<sp>GRAticule:{DUAl|SINgle}
DISPlay?<sp>GRAticule
```

Examples

```
DISP GRA:SIN
```
selects a single graticule.

INTensity

INTensity sets the overall display intensity.

Syntax:
```
DISPlay<sp>INTensity:<NRx>
DISPlay?<sp>INTensity
```

Range: `<NRx>` = 0 to 100 percent.

Examples

```
DISP INT:65
```
sets the display intensity.

INTERPolation

INTERPolation selects the type of interpolation when pan/zoom is enabled and HMAG is greater than one.

Syntax:
```
DISPlay<sp>INTERPolation:{LINear|NONe|
                          PSINc|SINC}
DISPlay?<sp>INTERPolation
```

Arguments

- **LINear**—displays points between acquired data points using linear interpolation.

- **NONe** – displays only acquired data points.
- **PSINc** – displays points between acquired data points using the same method as SINC, but the data is prefiltered to minimize overshoot and ringing for under-sampled edges.
- **SINC** – displays points between acquired data points using sin(x)/(x) interpolation.

Examples

```
DISP INTERP:NON
```
sets interpolation to none.

MODe

MODe selects a DOTs or VECtors type display.

Syntax:
```
DISPlay<sp>MODe:{DOTs|VECtors}
DISPlay?<sp>MODe
```

Arguments

- **DOTs** – displays individual data points.
- **VECtors** – connects adjacent data points.

Note: When more than 512 data points are acquired, the points are compressed to fit the 512-point scan line of the display. The largest and smallest adjacent vertical values are displayed as a single scan line connected with a vector. Thus, to get a true dots display, you may need to set TBMain LENgth or TBWin LENgth to 512.

Examples

```
DISP MOD:VEC
```
selects a vector display.

PERSistence

PERSistence sets the persistence time for waveforms displayed in variable persistence mode (see the discussion of ACCumulate under the TRA<*ui*> command). Persistence time is in seconds.

Syntax:
```
DISPlay<sp>PERSistence:<NRx>
DISPlay?<sp>PERSistence
```

Range: `<NRx>` = .2 to 60

Examples

```
DISP PERS:2.2E+0
```
sets data point persist time.

DISTal

DISTal sets the distal (furthest from origin) level used by RISetime? and FALltime? measurements.

Syntax:
```
DISTal<sp><NRx>
DISTal?
```

Range: `<NRx>` = 0 to 100, and specifies a percentage of the difference between the TOPline and BASEline values.

Examples

```
DIST 85
```
sets the furthest from origin level.

DLYtrace

DLYtrace specifies the delayed waveform used with the PDElay? measurement.

Syntax: `DLYtrace<sp>TRAce<ui>`
`DLYtrace?`

Range: `<ui>` = 0 to 8, and specifies the waveform.

The valid `<ui>` setting range is 1 to 8. However, if one waveform is defined, DLYtrace? returns TRAce1. If no traces are defined, DLYtrace? returns error 250. If you send DLYtrace TRAce0 to the DSA, it is ignored.

The GAIn? and PHAse? measurements use a reference trace set by the REFtrace command.

Each waveform has an associated delayed waveform; when you change the selected waveform, you may need to change the delayed waveform. Measurements are taken from the selected waveform to the delayed waveform. You cannot specify the selected waveform as the delayed waveform.

Changing Measurement Parameters on the Delayed Waveform. The PDElay? measurement returns the time from the first mesial crossing of the selected waveform to the first mesial crossing of the delayed waveform. Every waveform has its own measurement parameters (e.g., MESial, LMZone) which can be changed only when that waveform is the selected waveform. Therefore, use the following procedure if you need to change measurement parameters on the delayed waveform:

1. Use the SELect command to make the delayed waveform the selected waveform.
2. Change the measurement parameters.
3. Use the SELect command to reassign the correct selected waveform.

Here is an example of the entire process of taking a PDElay measurement. Assume you want to measure PDElay between TRAce2 (the selected waveform) and TRAce4 (its delayed waveform). The required MESial values are 40% and 45%, respectively.

■	SELect TRAce2	/* Specify selected waveform */
■	MESial 40	/* Specify its mesial value */
■	DLYtrace TRAce4	/* Specify its delayed waveform */
■	SELect TRAce4	/* Select trace 4 to change its parameters*/
■	MESial 45	/* Specify its mesial value */
■	SELect TRAce2	/* Return to the original selected waveform */
■	PDElay?	/* Measure PDElay from trace 2 to trace 4 */

Examples

`DLY TRA2`
selects the delayed waveform that is used with the PDElay measurement.

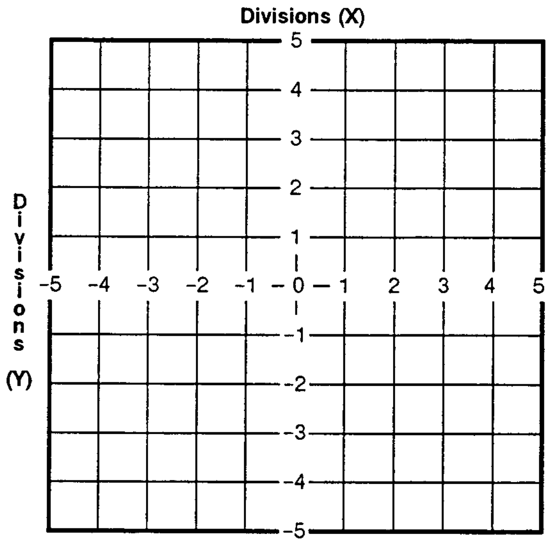
DOT1Abs DOT2Abs

PCTg
XCOord
XDIv
XQUal
YCOord
YDIv
YQUal

DOT1Abs and DOT2Abs set absolute horizontal and vertical positions (with respect to the waveform record) for split or paired (dot) cursors. DOT1Abs and DOT2Abs have the same parameters.

Syntax:
```
DOT{1|2}Abs<sp><link>:<arg>
DOT{1|2}Abs?[<sp><link>]
```

The following figure illustrates the graticule coordinates:



Graticule X, Y Coordinates

PCTg

PCTG positions the first or second dot cursor as a percentage of the waveform record.

Syntax:
```
DOT{1|2}Abs<sp>PCTg:<NRx>
DOT{1|2}Abs?<sp>PCTg
```

Range: <NRx> = 0 to 100 percent

XY Note: You must use the PCTg link to position the cursors for XY waveforms. Attempting to use XCOord or XDIv will give unpredictable results.

Examples

`DOT2A PCT:10`
positions the second dot cursor as a percentage.

XCOord

XCOord positions the first or second dot cursor with respect to horizontal units of the selected waveform.

Syntax:
```
DOT{1|2}Abs<sp>XCOord:<NRx>
DOT{1|2}Abs?<sp>XCOord
```

Range: The following range formulas assume ADJtrace PANzoom is set to OFF and the waveform is acquired. Refer to the cursor positioning discussion on page 139 for calculating XCOord range when PANzoom is set to ON or the waveform is unacquired. Refer to page 274 for formulas to calculate *duration*.

XCOord range when the selected waveform record is MAIN:

MAINPos to (MAINPos + *main_duration*)

XCOord range when the selected waveform record is WIN1:

WIN1Pos to (WIN1Pos + *win_duration*)

XCOord range when the selected waveform record is WIN2:

WIN2Pos to (WIN2Pos + *win_duration*)

Examples

`DOT1A XCO:1.2E-2`
positions the first dot cursor with respect to the horizontal units.

XDIv

XDIv positions the first or second dot cursor in graticule divisions (refer to the graticule illustration on page 135).

Syntax:
```
DOT{1|2}Abs<sp>XDIv:<NRx>
DOT{1|2}Abs?<sp>XDIv
```

Range: Range depends on record (TBMain or TBWin) length.

XDIV Ranges

Record LENgth	XDIV Range (<NRx>)
4096, 8192, or 16384	-5.12 to +3.07
32768	-5.12 to +1.42
Any other LENGTH	-5.12 to +5.10

These ranges are valid only when ADJtrace PANzoom is OFF and the selected waveform is acquired. (Refer to the Range of Cursor Positioning discussion on page 139 for calculating XCOord range when PANzoom is set to ON or the waveform is unacquired.)

Examples

```
DOT1A XDI:2.85
```
positions the first dot cursor in graticule divisions.

XQUal

Query Only. XQUal returns the accuracy of XCOord or XDIv positioning information.

Syntax:
```
DOT{1|2}Abs?<sp>XQUal
```

Query Responses

- **EQ**—the true position and response are equal. YT waveforms always return the EQ qualifier because the cursor horizontal position is always known.
- **GT**—the true position is greater than response (i.e. the cursor is above the top of the screen).

- **LT** – the true position is lower than the response (i.e. the cursor is below the bottom of the screen).
- **UN** – the true position is uncertain (i.e. the cursor is on an unacquired waveform point).

Examples

`DOT1A? XQU`
returns the accuracy of positioning information, such as:
`DOT1ABS XQUAL:EQ`

YCOord

Query Only. YCOord returns the vertical position of the first or second dot cursor, in units of the selected waveform.

Syntax: `DOT{1|2}Abs?<sp>YCOord`

Examples

`DOT1A? YCO`
returns the vertical position of the first dot cursor, such as:
`DOT2ABS YCOORD:2.22E-4`

YDIv

Query Only. YDIv returns the vertical position of the first or second dot cursor in graticule divisions. (Refer to the graticule illustration on page 135.)

Syntax: `DOT{1|2}Abs?<sp>YDIv`

Examples

`DOT2A? YDI`
returns the vertical position of the second dot cursor in graticule divisions, such as:
`DOT1ABS YDIV:-1.4`

YQUal

Query Only. YQUal returns the accuracy of YCOord or YDIv positioning information.

Syntax: `DOT{1|2}Abs?<sp>YQUal`

Query Responses

- **EQ** – the true position and response are equal.
- **GT** – the true position is greater than response (i.e. the cursor is above the top of the screen).
- **LT** – the true position is lower than response (i.e. the cursor is below the bottom of the screen).
- **UN** – the true position is uncertain (i.e. the cursor is on an unacquired waveform point).

Range of Cursor Positioning. Under some circumstances, such as when PANzoom is set to ON, you cannot conveniently compute the valid range of cursor positions. However, you can force the cursors to their minimum and maximum values (use the PCTg:0 and PCTg:100 links) and then query the DSA for the cursor positions. These new positions constitute the valid range of cursor positions for that particular DSA setup.

The following example demonstrates this technique. This method applies to both dot and bar cursors and is always successful, regardless of DSA settings.

Examples

```
DOT1A PCT:0
DOT1A PCT:100
DOT1A? XCO;DOT2A? XCO
DOT1ABS XCOORD:-6.0E-6
DOT1ABS XCOORD:5.055E-4
```

DOT1Rel DOT2Rel

PCTg
XCOord
XDIv

Set Only. DOT1Rel and DOT2Rel set the paired or split (dot) cursor position relative to the current absolute cursor location. DOT1Rel and DOT2Rel have the same links.

Syntax: `DOT{1|2}Rel<sp><link>:<arg>`

Note: These commands position the dot cursors relative to their current position. This means the range is twice as large as the DOT{1|2}Abs command range. The cursor will never be positioned outside the legal range as defined for the DOT{1|2}Abs command.

PCTg

PCTg positions the first or second dot cursor as a percentage of the waveform record, relative to the DOT1Abs/DOT2Abs value.

Syntax: `DOT{1|2}Rel<sp>PCTg:<NRx>`

Range: `-current postion`≤`<NRx>`≤ 100–current postion

Examples

`DOT1R PCT:50`
positions the first dot cursor as a percentage of the waveform record.

XCOord

XCOord positions the first or second dot cursor with respect to the units of the selected waveform, relative to the DOT1Abs/DOT2Abs value.

Syntax: `DOT{1|2}Rel<sp>XCOord:<NRx>`

Range: `-duration-current XCOord`≤`<NRx>`≤+duration–current `XCOord`

Examples

`DOT2R XCO:0.5`
positions the second dot cursor.

XDIv

XDIv positions the first or second dot cursor in graticule divisions with respect to the selected waveform, relative to, but not exceeding, the DOT1Abs/DOT2Abs value.

Syntax: `DOT{1|2}Rel<sp>XDIv:<NRx>`

Range: `-5.12-current XCOord`≤`<NRx>`≤+5.10–current `XCOord`

Examples

`DOT2R XDI:2.85`
positions the second dot cursor in graticule divisions.

DSYmenu

Query Only. DSYmenu returns the major menu active on the front panel display.

Syntax: `DSYmenu?`

Query Responses

- ALL_Wavfrm – the All Waveforms Status menu.
- CURSor – the Cursor menu.
- MEAS – the Measurement menu.
- STATHIST – the Statistics/Histogram menu.
- STORE_Recall – the Store/Recall menu.
- TRIgger – the Trigger menu.
- UTILITY1 – the Utility1 menu.
- UTILITY2 – the Utility2 menu.
- UTILITY3 – the Utility3 menu.
- WAVfrm – the Waveform menu.
- WFMSCAN – the stored Waveform Scan menu.

Examples

```
DSY?
```

returns the active front panel menu, such as:

```
DSYMENU CURSOR
```

DSYSTOFmt

DSYSTOFmt determines the format of the stored waveform timestamp displayed in the menus. Both date and hundredths of seconds are recorded whenever a waveform is stored, but only one appears in the timestamp.

Syntax: `DSYSTOFmt<sp>{HUNdredths|DATE}`

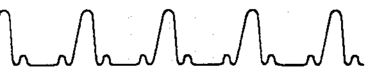
Arguments

- **HUNdredths** – selects hours, minutes, seconds, and hundredths of seconds. This is especially useful when a number of waveforms have been stored using repetitive single trigger or Act-on-Delta acquisition in quick succession.
- **DATE** – selects hours, minutes, seconds, and date.

Note: The DSYSTOFmt setting when the waveform was stored does not affect the available timestamp information, so either DATE or HUNdredths may be selected at any time.

Examples

```
DSYSTOF HUN
```



DUTy

Query Only. DUTy returns the percentage of a period that a waveform spends above the MESial level, followed by an accuracy qualifier. (See page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `DUTy?`

Returns: <NRx> or <bblock>

Examples

```
DUT?
```

returns the percentage of a period that a trace spends above the MSEial level, such as:

```
DUTY 5.071E+1,EQ
```

ENCdg

HISTogram
MEAs
SET
WAVfrm

ENCdg determines the data encoding for information returned by CURVe?, HISTogram?, WAVfrm?, <meas>?, and SET? queries.

Syntax:
```
ENCdg<sp><link>:<arg>
ENCdg?[<sp><link>]
```

HISTogram

HISTogram sets the encoding for data points in a histogram transferred with the HISTogram? DATA query.

Syntax:
```
ENCdg<sp>HISTogram:{ASCii|BINary}
```

Examples

`ENC HIST:ASC`
selects ASCII encoding for data transfer.

SET

SET sets the encoding for front panel setting (FPS) transfers with the SET? query.

Syntax:
```
ENCdg<sp>SET:{ASCii|BINary}
ENCdg?<sp>SET
```

Examples

`ENC SET:ASC`
selects ASCII encoding for SET query data transfer.

WAVfrm

WAVfrm sets the encoding for waveform transfers with the CURVe? and WAVfrm? queries.

Syntax:
```
ENCdg<sp>WAVfrm:{ASCii|BINary}
ENCdg?<sp>WAVfrm
```

Examples

`ENC WAV:BIN`
selects binary encoding for data transfer.

MEAs

MEAs sets the encoding for measurement data transfers with any measurement command.

Syntax:
```
ENCdg<sp>MEAs:{ASCii|BINary}
ENCdg?<sp>MEAs
```

Examples

`ENC MEA:BIN`
selects binary encoding for data transfer.

ENV

ENV sets enveloping ON or OFF for the vertical expression component <y exp> (e.g., L1) of the waveform description of the selected waveform. (Refer also to the TRACe and AVG commands.)

Syntax: `ENV<sp>{OFF|ON}`
`ENV?`

Arguments

- **OFF**—removes the enclosing ENV() when <y exp> is enclosed with ENV(). You cannot set ENV to OFF when the <y exp> is not enclosed with ENV().

- **ON**—encloses <y exp> with ENV(), or ENV() replaces AVG() when <y exp> is enclosed with AVG(). You cannot set ENV to ON if the selected waveform is XY or has only stored and/or scalar components.

Query Responses

- **ENV?**—returns the state of enveloping. ENV ON means the entire <y exp> is enclosed by ENV. ENV OFF means the entire <y exp> is not enclosed by ENV, though the ENV() function may be embedded within the description.

Examples of ENV Usage

<y exp> **Before**	**Command**	<y exp> **After**
L2	**ENV ON**	ENV(L2)
L1	**ENV OFF**	-error-
AVG(C1-C2)	**ENV ON**	ENV(C1-C2)
ENV(R1)	**ENV OFF**	R1
ENV(C4)	**ENV ON**	ENV(ENV(C4))

Examples

```
ENV ON
```
turns enveloping on.

EVENT

Query Only. EVENT returns the event code <NR1> if LONgform is set to OFF, or returns the event code and a descriptive <qstring> if LONgform is set to ON.

Syntax: `EVENT?`

Refer to Event Reporting, later in this manual, for a list of event codes.

Examples

```
EVENT?
```

returns event code information, such as:

```
EVENT 269,"NO SUCH TRACE".
```

FALltime

Query Only. FALltime returns the transition time of a falling pulse edge, from the DISTal to PROXimal level, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `FALltime?`

Returns: `<NRx>` or `<bblock>`

Examples

`FAL?`

returns the transition time of a falling pulse edge, such as:

`FALLTIME 5.883E-9,EQ`

FEOi

Set Only. FEOi forces the DSA to output a message terminator for any pending query response. (The message terminator for GPIB is an EOI signal; the message terminator for RS232 is the EOL string. Refer to the RS232 command for the EOL options.) FEOi is useful to force the output of a recursive query (created with the DEF command) onto individual lines.

Syntax: `FEOi`

Note: FEOi has no argument.

Examples

`FEO`

forces the output of a message terminator for any pending query response.

FFT

AVG
DCSUP
FORMat
PHAse
WINDow

FFT controls the Fast Fourier Transform (FFT) parameters. The FFT function is part of the waveform description. (Refer to the TRAce DEScription command.)

Syntax:

```
FFT<sp><link><arg>
FFT? [<sp><link>]
```

AVG

AVG controls averaging of the FFT source. When averaging is on, it is applied to all FFT calculations (rather than on a per-waveform basis) and is done prior to the FFT calculation. FFT AVG does not affect the waveform description.

Syntax:

```
FFT<sp>AVG:{OFF|ON}
FFT?<sp>AVG
```

Examples

```
FFT AVG:ON
```

DCSUP

DCSUP turns DC suppression ON or OFF. Some DC components may remain, even when DCSUP is on.

Syntax:

```
FFT<sp>DCSUP:{OFF|ON}
FFT?<sp>DCSUP
```

Examples

```
FFT DCSUP:ON
```

FORMat

FORMat specifies the magnitude output format.

Syntax:

```
FFT<sp>FORMat:{DBM|DBFund|DBVPeak|DBVRms|
               VPEak|VRMs}
FFT?<sp>FORMat
```

Arguments

- **DBM** – causes the FFT magnitude to be displayed in dB, decibel units relative to 1 mW; for example, a sine wave of 0.316 V peak (0.224 V rms) will give 1 mW into 50 Ω and will display an FFT magnitude of 0 dB. Signals of a lesser magnitude have a negative dB value.
- **DBFund** – displays logarithmic magnitudes relative to the fundamental.
- **DBVPeak** – displays logarithmic magnitude based on peak volts.
- **DBVRms** – displays logarithmic magnitude based on RMS volts.
- **VPEak** – displays linear magnitude based on peak volts. This is equivalent to LINEAR(in earlier releases of the software).
- **VRMs** – displays linear magnitude based on RMS volts.

Examples

```
FFT FORM:VPE
```

PHAse

PHAse specifies the phase output format. When PHAse is set to WRAp, FFT phase output is constrained to +180° to -180°. Phase data is wrapped from -180° to +180°. When PHAse is set to UNWrap, no constraints are placed on the phase data.

Syntax:
```
FFT<sp>PHAse:{UNWrap|WRAp}
FFT?<sp>PHAse
```

Examples

```
FFT PHA:UNW
```

WINDow

WINDOW specifies the window (or taper) function used to remove the effects of time domain discontinuities. The algorithms associated with these windows are included in the *DSA 601A and DSA 602A User Reference.*

Syntax:

```
FFT<sp>WINDow:{BLAckman|BLHarris|HAMming
               |HANning|RECTangular|
               TRIAngular}
FFT?<sp>WINDow
```

Examples

```
FFT WIND:BLH
```

FILTer

FILTer controls anti-alias filter mode.

Syntax: `FILTer<sp>{DISAble|ENAble}`
`FILTer?`

Arguments

- **DISAble** – the digitizer bandwidth is not limited.
- **ENAble** – the digitizer bandwidth is limited to approximately 100 MHz. When FILTer is set to ENAble, the following conditions are forced:

  Sample rate for a single-channel acquisition of <1 GSamples/s for a DSA 601A or <2 GSamples/s for a DSA 602A.

  Sample rate for two-channel acquisitions of ≤1 GSamples/s for a DSA 602A.

Note: Refer to the CH command to set the system bandwidth.

Examples

```
FILT ENA
```

FORMAT

Set only. The FORMAT command formats and labels floppy disks.The label can be eight characters or less.

Syntax: `FORMAT<sp><qstring>[,<qstring>]`

Range: `<qstring>` = "A:", and specifies the drive that contains the disk to be formatted. The second (optional) <qstring> is the volume label for the disk.

Examples

```
FORM 'A:','test1'
```

FPAnel

FPanel OFF functionally mimics the GPIB RWLS (Remote With Lockout State) and FPAnel ON mimics the GPIB LOCS (Local State).

Syntax:
```
FPAnel<sp>{OFF|ON}
FPAnel?
```

Arguments

- **OFF** – the front panel is locked out and only these controls are operable:

  RQS icon, if it was enabled (displayed) with the SRQMask USEr:ON command. (The RQS icon is not displayed at power on.) If enabled, you can disable the RQS icon with SRQMask USEr:OFF.

  Probe ID button, if SRQMask PROBE is set to ON. When FPAnel is set to OFF, the only effect of pressing the button is that event code 457 will be returned to both the GPIB and RS-232-C ports.

- **ON** – all front panel controls are operable, assuming the **TOUCH PANEL ON/OFF** button is set to ON.

The differences between the FPAnel command and the **TOUCH PANEL ON/OFF** button are:

- FPAnel provides a way to lock out active front panel controls (knobs, buttons, and screen touches) from the remote interfaces. There is no front panel equivalent to FPAnel.

- The **TOUCH PANEL ON/OFF** button only locks out screen touches. No command mimics the effect of this button. However, you can use the ABStouch command to simulate a touch to this button from the remote interfaces.

Examples

```
FPA ON
```
makes all front panel controls operable.

FPSList

Query Only. FPSList returns a list of all front panel settings stored on the disk or in nonvolatile RAM (NVRAM). If RAM is the device, the amount of memory used to store each setting is also displayed.

Syntax: `FPSList?`

Query Responses

- **FPS<ui>:<seq>,<bytes>**... – is the format of the list of front panel settings that are stored.
- **<qstring>**[,**<qstring>**,...] – file name(s) on the disk.
- **EMPty** – means no settings are stored.

Examples

`FPSL?`

returns front panel settings stored in memory, such as:

```
FPSLIST FPS2:1,1056,FPS5:2,979
```

Note: If you get unexpected results, make sure the device is set properly (to RAM or DISK) with the SETDev command.

FPSNum

Query Only. FPSNum returns the number of front panel settings (FPS) stored in the current disk directory or in nonvolatile RAM.

Syntax: `FPSNum?`

Returns: `<NR1>`

Examples

`FPSN?`

returns the number of front panel settings stored, such as:

```
FPSNUM 2
```

FPUpdate

FPUpdate determines whether the front panel display readouts will be updated following set command execution. The power-on default is FPUpdate EMPty.

Syntax: `FPUpdate<sp>{ALWays|EMPty|NEVer}`
`FPUpdate?`

Arguments

- **ALWays** – the front panel display is updated after each successful set command.
- **EMPty** – the front panel display is only updated when:

  The DSA receives DCL or SDC.

  The DSA receives a syntactically or semantically incorrect query or set command.

  The DSA input buffer is empty after a successful set or query execution.

- **NEVer** – the front panel display is not updated until FPUpdate is changed to ALWays or EMPty, or DSA power is cycled off and on. (However, data will be written to the display by the DEBug or TEXt commands.)

Note: Front panel controls function with FPUpdate ALWays or FPUpdate EMPty, but do not function with FPUpdate NEVer.

Note: Command throughput is faster with FPUpdate set to EMPty and is fastest with FPUpdate set to NEVer.

The links ON and OFF are included for compatibility with 11401 and 11402 Mainframes and will not be returned to a query. ON is equivalent to ALWays; OFF is equivalent to EMPty.

Examples

```
FPU EMP
```

FREq

Query Only. FREq returns the frequency of the signal, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `FREq?`

Returns: `<NRx> or <bblock>`

Examples

```
FRE?
```

returns the signal frequency, such as:

```
FREQ 1.024E+6,EQ
```

GAIn

Query Only. GAIn? returns the ratio of the peak-to-peak amplitude of the reference waveform to the peak-to-peak amplitude of the selected waveform, followed by an accuracy qualifier. Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

(Refer to page 192 for qualifier definitions.)

Syntax: `GAIn[<ui>]?`

Returns: `<NRx> or <bblock>`

Examples

`GAI?`

returns the peak-to-peak amplitude ratio of the reference and selected waveforms, such as:

```
GAIN 1.007E+0,EQ
```

H1Bar H2Bar

YCOord
YDIv

H1Bar and H2Bar sets the absolute vertical position of horizontal bar cursors. H1Bar and H2Bar have the same parameters.

Syntax:
```
H{1|2}Bar<sp><link>:<arg>
H{1|2}Bar?[<sp><link>]
```

YCOord

YCOord positions the first or second horizontal bar cursor with respect to the units of the selected waveform. The range depends on whether the waveform was created in integer mode or floating-point mode.

Syntax:
```
H{1|2}Bar<sp>YCOord:<NRx>
H{1|2}Bar?<sp>YCOord
```

Range: The YCOord range for an integer mode waveform is:

(SEN * -5.12 + OFFS) to (SEN * 5.10 + OFFS)

where SEN and OFFS are the channel sensitivity and offset (CH<slot><ui>? SEN,OFFS) of the channel(s) in the integer mode waveform.

The YCOord range for a floating-point mode waveform is:

(VSI * -5.12 + VPO) to (VSI * 5.10 + VPO)

where VSI and VPO are the vertical size and vertical position (ADJ<ui>? VSI,VPO) of the floating-point waveform.

Note: For information on waveform modes, see the WFMScaling command.

Examples

```
HlBar YCO:0.75
```
positions the first bar cursor with respect to units of the selected trace.

YDIv

YDIv positions the first or second horizontal bar cursor in graticule divisions.

Syntax: `H{1|2}Bar<sp>YDIv:<NRx>`
`H{1|2}Bar?<sp>YDIv`

Range: `<NRx>` = -5.12 to +5.10

Examples

`H2Bar YDI:-4.0`
positions the second bar cursor in graticule divisions.

HISTogram

DATA
HISTScaling
C.WINBottom
C.WINLeft
C.WINRight
C.WINTop
D.WINBottom
D.WINLeft
D.WINRight
D.WINTop
NR.pt
TYPe

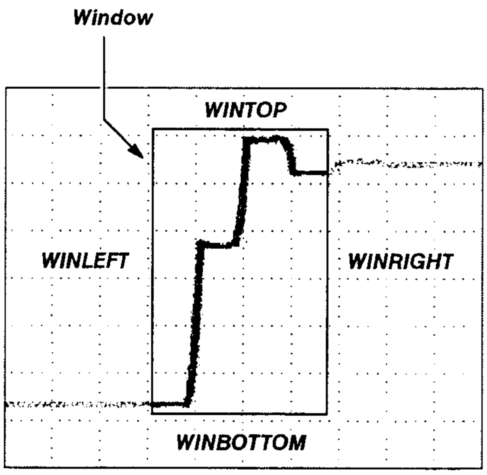
HISTogram initiates a vertical or horizontal histogram display for the selected trace. It also sets a variety of histogram parameters including the dimensions of a displayed histogram window.

Syntax:
```
HISTogram<sp>{CLEar|<link>:<arg>}
HISTogram?[<sp>{DATA|<link>}]
```

The histogram window selects a portion of the trace on which to perform the histogram algorithm. The histogram window appears on the display when the HISTogram TYPe is set to HORiz or VERt, and the second page of the MEASURE major menu is selected (see the DSYmenu command on page 142).

Each displayed trace has a unique histogram window and a unique HISTogram TYPe. Once a histogram is started on a trace, selecting another trace will activate the histogram window for that trace.

The C.WIN links specify the sides in the current vertical and horizontal scale units (i.e., volts, seconds, hertz, etc.) of the selected trace. The D.WIN links specify the window in absolute screen divisions independent of the current scale settings. The histogram window can be defined with C.WIN links then queried with D.WIN links, and vice versa. The following illustration shows the four histogram window parameters.



Histogram Window Parameters

For more information on the use of the histogram function, refer to the *User Reference* for your instrument.

Arguments

- **CLEar** – removes all waveform and histogram data from the display and restarts all acquisitions.

Examples

```
HIST CLE
```

removes all histogram and waveform data from the display.

Query Responses

DATA

- **HISTogram? DATA** – transfers the value of each point on the histogram to the controller in binary or ASCII format. The histogram window determines what portion of the waveform will be incorporated in the histogram. It also determines how much histogram data will be transferred.

Histogram data points are sent as unsigned 32-bit values starting from the left of the screen for horizontal histograms and from the bottom for vertical histograms.

<Histogram data> can be in ASCII (<asc data>) or binary (<bblock>) format. The format is set by the ENCdg HISTogram command. Use the HISTogram? NR.pt query to get the number of histogram points to expect from the DATA query.

ASCII Transfer. Data transferred as an <asc data> use the following format:

```
<asc data> ::= <NR1> [ ,<NR1> ] ...
```

where <NR1> values are histogram data points within the range 0 to 4294967295.

Binary Transfer. Data is transferred as a binary block (`<bblock>`) where:

```
<bblock> ::= %<byte cnt><bin pt>...<checksum>
```

<bin pt> is a four-byte unsigned binary integer (MSB first) and <byte cnt> is an arbitrary number of binary bytes. This binary format is similar to that used for trace transfers with the CURVe command which is discussed on page 111.

The order of bytes within a bin count value is set with the BYT.OR command. You can set either the least significant byte (LSB) to be sent first followed by bytes of greater significance or the most significant byte (MSB) first followed by bytes of lesser significance. There are no separators (such as commas) between binary bin counts.

C.WINBottom

C.WINBottom specifies the bottom edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>C.WINBottom:<NRx>
HISTogram?<sp>C.WINBottom
```

Range: `<NRx>` is a vertical value in the current units of the vertical scale. The range is defined by the vertical graticule limits. See the D.WINBottom link for the default setting. C.WINBottom can never be greater than C.WINTop.

Examples

`HIST C.WINB:-2.5`
defines the bottom edge of the histogram window.

C.WINLeft

C.WINLeft specifies the left edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>C.WINLeft:<NRx>
HISTogram?<sp>C.WINLeft
```

Range: `<NRx>` is a horizontal value in units of the current horizontal scale. The range is defined by the end points of the trace record. See the D.WINLeft link for the default setting. C.WINLeft can never be greater than C.WINRight.

Examples

`HIST C.WINL:1.15`
defines the left edge of the histogram window.

C.WINRight

C.WINRight specifies the right edge of the histogram window for the selected waveform.

Syntax:
```
HISTogram<sp>C.WINRight:<NRx>
HISTogram?<sp>C.WINRight
```

Range: `<NRx>` is a horizontal value in units of the current horizontal scale. The range is defined by the end points of the waveform record. See the D.WINRight link for the default setting. C.WINRight can never be less than C.WINLeft.

Examples

`HIST C.WINR:4.05`
defines the right edge of the histogram window.

C.WINTop

C.WINTop specifies the top edge of the histogram window for the selected waveform.

Syntax:
```
HISTogram<sp>C.WINTop:<NRx>
HISTogram?<sp>C.WINTop
```

Range: `<NRx>` is a vertical value in units of the vertical scale. The range is defined by the graticule limits. See the D.WINTop link for the default setting. C.WINTop can never be less than C.WINBottom.

Examples

`HIST C.WINT:1.5`
defines the top edge of the histogram window.

D.WINBottom

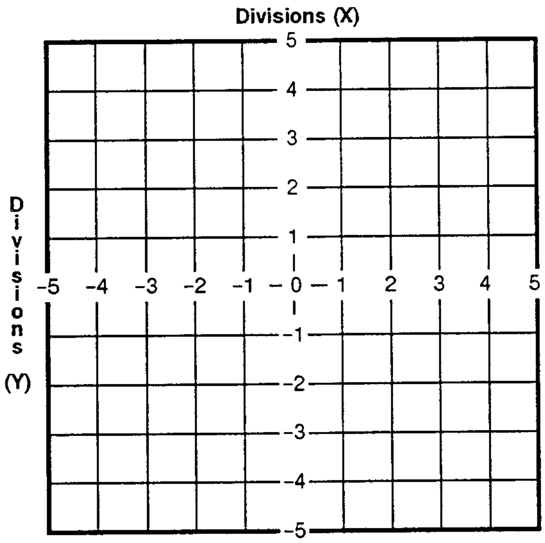
D.WINBottom specifies the bottom edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>D.WINBottom:<NRx>
HISTogram?<sp>D.WINBottom
```

Range: <NRx> is a vertical value in divisions within the range of -5.12 to + 5.10 divisions, though D.WINBottom can never be greater than D.WINTop.

The following illustration shows the coordinate system used to define D.WIN parameters. Because waveform records extend slightly beyond the left and right graticule limits, the D.WIN limits slightly exceed the -5 and +5 values shown. The illustration on page 162 shows the data window parameters and their associated WIN link. The default value is -4.

See the histogram discussion on page 162 for more information on the data selection window.



Graticule X, Y Coordinates

Examples

```
HIST D.WINB:-2.5
```
defines the bottom window of the histogram window.

D.WINLeft

D.WINLeft specifies the left edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>D.WINLeft<NRx>
HISTogram?<sp>D.WINLeft
```

Range: `<NRx>` is a horizontal value in divisions within the range of -5.12 to +5.10. D.WINLeft can never be greater than D.WINRight.The default value is -4.

Examples

```
HIST D.WINL:-1.15
```
defines the left edge of the histogram window.

D.WINRight

D.WINRight specifies the right edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>D.WINRight<NRx>
HISTogram?<sp>D.WINRight
```

Range: `<NRx>` is a horizontal value in divisions within the range of -5.12 to +5.10. D.WINRight can never be less than D.WINLeft. The default value is +4.

Examples

```
HIST D.WINR:4.05
```
defines the right edge of the histogram window.

D.WINTop

D.WINTop specifies the top edge of the histogram window for the selected trace.

Syntax:
```
HISTogram<sp>D.WINTop<NRx>
HISTogram?<sp>D.WINTop
```

Range: `<NRx>` is a vertical value in divisions within the range of -5.12 to +5.10. D.WINTop can never be less than D.WINBottom. The default value is +4.

Examples

```
HIST D.WINT:1.5
```
defines the top edge of the histogram window.

HISTScaling

HISTScaling selects either linear or logarithm base 10 scaling for the histogram display. All waveforms are affected. The default is LINear.

Syntax:
```
HISTogram<sp>HISTScaling:{LINear|LOG10}
HISTogram?<sp>HISTScaling
```

Examples

```
HIST HISTS:LOG10
```
selects logarithm base 10 scaling for the histogram display.

NR.pt

Query only. NR.pt returns the number of histogram points that will be returned by HISTogram? DATA.

Examples

```
HISTogram? NR.pt
```

TYPe

TYPE selects the type of histogram display for the selected trace.

Syntax:
```
HISTogram<sp>TYPe:{HORiz|NONe|VERt}
HISTogram?<sp>TYPe
```

Arguments

- **HORiz** – accumulates bin counts for each data point along the horizontal axis.
- **NONe** – disables the histogram function for the selected trace.
- **VERt** – accumulates histogram data for each data point along the vertical axis.

A histogram TYPe can be specified for each trace. The histogram window (C.WIN and D.WIN links) determines what portion of the trace is included in the histogram calculation.

Notes. If HISTogram is started when the ADJtrace ACCumulate type is NORmal or VARiable, the ADJtrace ACCumulate TYPe will change to INFinite persistence. Starting histograms will set the record length of the timebase used by the selected trace to 512 points.

Examples

```
HIST TYP:VER
```

selects a display that will accumulate histogram data for each point along the vertical axis.

HNUmber

HNUmber selects the harmonic number on which the SMAgnitude and SFRequency measurements are made when SMOde is set to HARmonic.

Syntax:
```
HNUmber<sp><NR1>
HNUmber?
```

Range: `<NR1>` = 1 to 1000

Examples

`HNU 3`
sets the harmonic number.

HPGl

COLor
FORMat
GRAticlue
PORt
SECUre

HPGl specifies printing parameters for the Tek HC100 plotter or other devices that conform to the HPGL format.

Syntax:
```
HPGl<sp><link>:<arg>
HPGl?[<sp><link>]
```

COLor

COLor assigns plotter pens to the DSA color index. Refer to page 92 for color index meanings.

Syntax:
```
HPGl<sp>COLor<ui>:<NRx>
HPGl<sp>COLor:DEFAult
HPGl?<sp>COLor
```

Range: `<ui>` = 0 to 7, and specifies the color.
`<NRx>` = 0 to 8, and specifies the pen.

Arguments

- **DEFAult**—assigns the following default pens to the color index:

Default Plotter Pen Assignments

Color Index	Pen No.	Color Index	Pen No.
0	1	4	5
1	2	5	6
2	3	6	7
3	4	7	8

Note: Assigning pen 0 to the color index means that color is not plotted (no pen is assigned).

Examples

```
HPG COL3:1
```
assigns a new color to pen 3.

FORMat

FORMat selects the output format. Pop-up menus are not plotted.

Syntax:
```
HPGl<sp>FORMat:{DRAft|HIRes|SCReen}
HPGl?<sp>FORMat
```

Arguments

- **DRAft**—is the same as SCReen except the front panel status menu is not plotted.
- **HIRes**—plots the entire screen, including every waveform point.
- **SCReen**—plots the entire screen, but includes only the min/max point-pairs of each YT waveform column (XY and PA waveforms are not affected.) This is the default mode.

Note: Plotting infinte or variable persistence waveforms is very time-consuming and tends to wear down plotter pen points more rapidly than other types of plots.

Examples

```
HPG FORM:HIR
```
specifies that the entire screen be output.

GRAticule

GRAticule selects the type of graticule printed on hardcopy output.

Syntax:
```
HPGl<sp>GRAticule:{CROSSHair|FULl}
HPGl?<sp>GRAticule
```

Arguments

- **CROSSHair**—prints a small crosshair (+) at graticule intersection points.
- **FULl**—prints the entire graticule. This is the default mode.

Examples

```
HPG GRA:CROSSH
```
selects crosshair graticule.

PORt

PORt specifies the output port for the plotter.

Syntax:
```
HPGl<sp>PORt:{CENTRonics|GPIb|RS232|DISk}
HPGl?<sp>PORt
```

Examples

```
HPG POR:CENTR
```
selects the Centronics port for plotter output. The Centronics port is the default output port.

SECUre

SECUre specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax:

```
HPGl<sp>SECUre:{ON|OFF}
HPGl?<sp>SECUre
```

Examples

```
HPG SECU:OFF
```

turns off the SECUre function.

HSYs

HSYs turns the Histogram system ON or OFF on the front panel display. HSYs must be ON to make histogram measurements.

Syntax:

```
HSYs<sp>{OFF|ON}
HSYs?
```

Set HSYs to ON when you need to make histogram measurements. Set HSYs to OFF for faster remote system throughput.

Examples

```
HSY OFF
```

turns the Histogram system off.

ID

Query Only. ID returns identifying information about the DSA and its firmware, delimited by commas.

Syntax: `ID?`

The query response includes the following items:

- **<model number>**–the DSA model number.
- **V<NR2>** –the TEK Codes & Formats version number.
- **DIG/<NR2>** –the digitizer processor (DIG) firmware version.
- **DSY/<NR2>** –the display processor (DSY) firmware version.
- **EXP/<NR2>** –the executive processor (EXP) firmware version.

Examples

`ID?`
returns DSA information, such as:
```
ID TEK/DSA602A,V81.1,DIG/1.0,DSY/1.0,EXP/1.0
```

IDProbe

Query Only. IDProbe returns the channel number (<slot><ui>) of the last probe ID button pressed by the operator. IDPRobe? returns L0 if no probe ID button was pressed.

Syntax: `IDProbe?`

Note: IDProbe? does not distinguish between the plus and minus probes of a differential amplifier.

Examples

`IDP?`
returns the last probe ID button pressed, such as:
```
IDPROBE C2
```

INCAcq

INCAcq controls incremental acquire mode of the digitizer.

Syntax: `INCAcq<sp>{DISAble|ENAble}`

In addition to INCAcq being set to ENAble, incremental acquire mode requires the following:

- No windows are being acquired.
- Main time base is greater than 5 ms/point.
- Total number of samples is 32,256 for all acquired waveforms.
- No calculated waveforms (e.g., L1*L2) are being acquired.
- CONDAcq TYPe is not DELta.
- No stored waveforms are displayed.

Examples

```
INCA ENA
```

INIt

Set Only. INIt initializes the DSA to its factory-assigned default parameters and settings. Completion of INIt is signaled by event code 474, "INIt complete."

Syntax: `INIt`

For both GPIB and RS-232-C, the defaults are:

- ABStouch FIFO buffer is empty.
- DEBug is OFF.
- IDProbe button press is cleared.
- SRQMask USEr is OFF; this removes the RQS icon if it was displayed.
- All pending events except Power On are discarded.
- All user TEXt is cleared from the display.
- For GPIB only, RQS is set to ON.

Note: INIt has no argument.

Examples

`INI`

initializes the DSA to its default settings.

INPut

INPut selects the destination for preamble and waveform data sent to the DSA by the WFMpre and CURVe commands.

Syntax: `INPut<sp>{STO<ui>|<qstring>}`
`INPut?`

Range: `<ui>` = 1 to 420† or 918‡, and specifies the stored waveform location.

†Without the disk drive, the range is 1-453.

‡The range is 1 to 918 when Option 4C, Nonvolatile RAM, is installed.

Arguments

- **STO<ui>** – a stored waveform destination. The power-on default INPut location is STO1.
- **<qstring>** – a label that identifies the stored waveform destination.

Query Note: INPut? always returns STO<ui>, even if the location was specified with a label.

Examples

`INP STO92`
selects the specified destination for data sent to the DSA by the WFMpre and CURVe queries.

INTERleave

INTERleave controls digitizer interleave mode. Interleave mode must be enabled to achieve a sample rate of 1 Gsamples/s for a DSA 601A or 2 Gsamples/s for a DSA 602A. However, the sample rate is not *forced* to any specific rate; this mode only *allows* these rates to be attained when other conditions are met.

Syntax:
```
INTERleave<sp>{DISAble|ENAble}
INTERleave?
```

Examples

```
INTER ENA
```

LABAbs

PCTg
XCOord
YDIv

LABAbs positions the label associated with the selected waveform.

Syntax:
```
LABAbs<sp><link>:<arg>
LABAbs?[<sp><link>]
```

Examples

```
LABA?
```
returns waveform label position information, such as:
```
LABABS XCOORD:-9.8E-5,PCTG:1.369863E+0,
   YDIV:3.0E-1
```

PCTg

PCTg sets the horizontal position of the label as a percentage of the waveform record.

Syntax:
```
LABAbs<sp>PCTg:<NRx>
LABAbs?<sp>PCTg
```

Range: `<NRx>` = 0 to 100

Examples

```
LABA PCT:50
```
specifies the horizontal position of the label.

XCOord

XCOord sets the horizontal position of the label in horizontal units. The label maintains the specified position, tracking changes in the waveform.

Syntax:
```
LABAbs<sp>XCOord:<NRx>
LABAbs?<sp>XCOord
```

Range: The following range formulas assume ADJtrace PANzoom is set to OFF and the waveform is acquired. Refer to the discussion on cursor positioning on page 139 for a method to calculate XCOord range when PANzoom is set to ON or the waveform is unacquired. Refer to page 274 for formulas to calculate *duration*.

XCOord range when the selected waveform record is MAIN is calculated:

MAINPos to (MAINPos + *main_duration*)

XCOord range when the selected waveform record is WIN1 is calculated:

WIN1Pos to (WIN1Pos + *win_duration*)

XCOord range when the selected waveform record is WIN2 is calculated:

WIN2Pos to (WIN2Pos + *win_duration*)

Examples

`LABA XCO:0.5`
horizontally positions the waveform label.

YDIv

YDIV sets the vertical position of the label in divisions, relative to the point specified by the XCOord link. The label maintains the specified vertical distance, tracking changes in the waveform.

Syntax:
```
LABAbs<sp>YDIv:<NRx>
LABAbs?<sp>YDIv
```

Range: `<NRx>` = −10.22 to +10.22

Examples

`LABA YDI:2.85`
vertically positions the waveform label.

LABel

BASELAbel
DELete
DISK
DISPlay
FPS
NEXTRep
STO
TRAce

LABel defines and deletes labels, and controls label display. LABel also labels disks.

Syntax: `LABel<sp><link>:<arg>`
`LABel?[<sp><link>]`

Label Wildcard Characters. For some commands that take labels, the characters **?** and * have a special meaning in a <qstring> when searching for a matching label. The **?** will match any single character. The * will match any number (including 0) of any character. To search for a literal ? or *, use a backslash \ in front of the ? or *.

Examples of Wildcard Usage

a?c	*matches*	abc, axc, a2c, aEc, etc.
rep1?	*matches*	rep11, rep12, rep1b, etc.
rep*	*matches*	rep, rep65, rep1a92, repZZ, etc.
a*c	*matches*	abc, a3478c, axyzc, etc.
a*c	*matches*	a*c

BASELAbel

BASELAbel defines the base part of the label generated for stored waveforms created in repetitive single trigger acquisition mode or through Act-On-Delta. (Refer to the CONDacq and DELTa commands.) An index value is appended to this base label to form the full stored waveform label. Numerals are not permitted in BASELAbel.

Syntax: `LABel<sp>BASELAbel:<qstring>`
`LABel?<sp>BASELAbel`

Range: `<qstring>` is $\leq$ 5 characters.

Examples

`LAB BASELA:'TESTA'`
sets the baselabel.

DELete

DELete deletes labels for active waveforms, stored waveforms, stored settings, or ALL labels. Waveforms and stored settings on the disk are not affected.

Syntax: `LABel<sp>DELete:{ALL|FPS[<ui>]|STO[<ui>]|TRAce[<ui>]|<qstring>}`

Range: The range for `FPS<ui>` is from 1 to 20
The range for `STO<ui>` is 1 to 420† or 1 to 918‡
The range for `TRAce<ui>` is 1 to 8.

†Without the disk drive, the range is 1-453.

‡The range is 1 to 918 when Option 4C, Nonvolatile RAM, is installed.

Arguments

- **ALL**—deletes all labels.
- **FPS**—deletes one or all stored front panel setting labels. Specifying FPS with <ui> deletes the label associated with the specified argument.
- **STO**—deletes one or all stored waveform labels. Specifying STO with <ui> deletes the label associated with the specified argument.
- **TRAce**—deletes one or all active waveform labels. Specifying TRAce with <ui> deletes the label associated with the specified argument.
- **<qstring>**—deletes that label. Wildcard characters are interpreted. (Refer to page 181 for wildcards.)

Note: Setting a label to a null string is the same as deleting a label.

Examples

`LAB DEL:TRA2`
deletes the label for waveform 2.

DISK

DISK specifies a disk label.

Syntax:
```
LABel<sp>DISK:<qstring>
LABel?<sp>DISK
```

Arguments

- **<qstring>** – specifies the disk label (eight characters or less, no spaces).

Examples

```
LAB DISK:"HDTV_ONE"
```
labels the disk.

DISPlay

DISPlay controls the display of labels associated with active waveforms.

Syntax:
```
LABel<sp>DISPlay:{OFF|ON}
LABel?<sp>DISPlay
```

Arguments

- **OFF** – labels are not displayed but all labels are retained.
- **ON** – labels are displayed.

Examples

```
LAB DISP:ON
```
turns label display on.

FPS

FPS defines a label for a stored front panel setting.

Syntax: `LABel<sp>FPS<ui>:<qstring>`
`LABel?<sp>FPS<ui>`

Range: `<ui>` = 1 to 20
`<qstring>` is $\leq$10 characters.

Examples

`LAB FPS1:'SETUP1'`
defines a label for the first stored front panel setting.

NEXTRep

Query Only. NEXTRep returns the value of the next label to be used by the Repetitive Trigger acquisition mode, or Act-on-Delta mode.

Syntax: `LABel<sp>NEXTRep?`

Returns: `<qstring>`

Examples

`LAB NEXTR?`
returns the next label, such as:

```
LABEL NEXTREP:"REP2"
```

STO

STO defines the label for a stored waveform.

Syntax:
```
LABel<sp>STO<ui>:<qstring>
LABel?<sp>STO<ui>
```

Range: <ui> = 1 to 420† or 1 to 918‡
<qstring> is ≤10 characters.

†*Without the disk drive, the range is 1-453.*

‡*The range is 1 to 918 when Option 4C, Nonvolatile RAM, is installed.*

Examples

```
LAB STO2:'DATA1'
```
defines a label for the second stored waveform.

TRAce

TRAce defines the label for an active waveform.

Syntax:
```
LABel<sp>TRAce<ui>:<qstring>
LABel?<sp>TRAce<ui>
```

Range: <ui> = 1 to 8
<qstring> is ≤10 characters.

Examples

```
LAB TRA1:'CLOCK'
```
defines a label for waveform1.

LABRel

PCTg
XCOord
YDIv

LABRel positions the label of the selected waveform relative to its position prior to the command.

Syntax: `LABRel<sp><link>:<arg>`

Note: These commands position the label relative to their current position. This means the range is twice as large as the LABAbs command range. The label will never be positioned outside the legal range as defined for the LABAbs command.

PCTg

Set Only. PCTg changes the horizontal position of the label, relative to its previous horizontal position, in units of percent of record length, but not exceeding the LABAbs PCTg range.

Syntax: `LABRel<sp>PCTg:<NRx>`

Range: `-current postion`≤`<NRx>`≤ 100–current postion

Examples

`LABR PCT:50`
positions the label relative to its previous position.

XCOord

Set Only. XCOord changes the horizontal position of the label, relative to its previous horizontal position, but not exceeding the LABAbs XCOord range.

Syntax: `LABRel<sp>XCOord:<NRx>`

Range: `-duration-current XCOord`≤`<NRx>`≤+duration–current `XCOord`

Examples

`LABR XCO:0.5`
positions the label horizontally relative to its previous position.

YDIv

Set Only. YDIv changes the vertical position of the label relative to its previous vertical position, but not exceeding the LABAbs YDIv range.

Syntax: `LABRel<sp>YDIv:<NRx>`

Range: `-5.12-current XCOord`≤`<NRx>`≤+5.10-current `XCOord`

Examples

```
LABR YDI:2.85
```

positions the label vertically relative to its previous position.

LCAlconstants

LCAlconstants sets or queries the calibration constants of the left plug-in unit.

Syntax:
```
LCAlconstants<sp><ui>:<NRx>
LCAlconstants?[<sp><ui>]
```

Range: `<ui>` is the constant (range is plug-in unit specific)
`<NRx>` is the value of the constant.

Note: You can only set LCAlconstants when an internal jumper has been installed by a qualified service person.

Examples

```
LCAL? 12
```

returns the calibration constant for the left plug-in unit, such as:

```
LCALCONSTANTS 12:-1.011494E-2
```

LMZone

LMZone sets the left measurement zone limiter as a percentage of the waveform record, or in scaled units of the horizontal time-base. See the MTIme command.

Syntax:

```
LMZone<sp><NRx>
LMZone?
```

Range: `<NRx>` depends on the current MTIme value. When MTIme is set to RELative, LMZone is a percentage of the waveform record. When MTIme is set to ABSOlute, LMZone is an absolute position in horizontal units of the selected waveform.

RMZone Ranges

With MTIme RELative	With MTIme ABSOlute
0 to 100 %	XZE to (XZE + XIN * (NR.pt -1))

The MTIme ABSOlute range is calculated using XZEro, XINcr, and NR.PT values from the waveform preamble (WFMpre) of the selected trace.

Examples

`LMZ 0`
sets the left measurement zone limiter.

LONgform

LONgform controls the return of the longer versions of query responses. The power-on default is LONgform ON.

Syntax:
```
LONgform<sp>{OFF|ON}
LONgform?
```

Arguments

- **OFF** – query responses are in abbreviated form, and EVENT? and RS232 VERBose:ON responses include only the event codes.
- **ON** – queries respond with full header and link spellings; the EVENT? and RS232 VERBose:ON commands return a descriptive <qstring> in addition to the event code.

Examples

`LON ON`
returns query responses with unabbreviated headers and links.

MAINPos

MAINPos sets the horizontal position of the Main waveform record with respect to the Main trigger.

Syntax: `MAINPos<sp><NRx>`
`MAINPos?`

Range: `<NRx>` = -(main duration) to 0 seconds

Refer to page 274 for formulas to calculate *duration.*

Examples

```
MAINP -7.9E-6
```
sets the main waveforms horizontal position.

MAX

Query Only. MAX returns the maximum amplitude (most positive peak voltage) of the selected waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `MAX?`

Returns: `<NR3> or <bblock>`

Examples

```
MAX?
```
returns the maximum amplitude, such as:

```
MAX 5.04E-1,EQ
```

MCAIconstants

MCAIconstants sets or queries DSA calibration constants.

Syntax: `MCAlconstants<sp><ui>:<NRx>`
`MCAlconstants?[<sp><ui>]`

Range: `<ui>` = 1 to x, where x depends on the current firmware.
`<NRx>` = -2^{31} to $2^{31}-1$ and is the value of the constant.

Note: You can only set MCAIconstants after an internal jumper has been installed by a qualified service person.

Examples

```
MCA? 12
```
returns the calibration constant, such as:
```
MCA 12:2048
```

MEAN

Query Only. MEAN returns the average amplitude (arithmetic mean voltage) of the selected waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `MEAN?`

Returns: `<NR3> or <bblock>`

Examples

```
MEAN?
```
returns the average amplitude, such as:
```
MEAN 2.212E-1,EQ
```

MEAS

Query Only. MEAS executes the measurements (<meas>) in the current measurement list (MSLIst).

Syntax: `MEAS?`

Query Responses

- MEAS? returns a scalar value followed by an accuracy qualifier (<qual>) for each measurement in the list. The format is:

  ```
  MEAS {<meas>:<NR3>,<qual>
  [,{<meas>:<NR3>,<qual>...}]}
  ```

- EMPTY – if MSLIst contains no measurements.

The <qual> accuracy qualifier indicates whether or not the underlying waveform data contain null, overrange, or underrange values.

The measurement <qual> accuracy qualifiers are defined in the following table:

Measurement Accuracy Qualifiers (<qual>)

<qual>	**Meaning**
EQ	True measurement equals value returned
LT	True measurement is less than value returned
GT	True measurement is greater than value returned
UN	True measurement is uncertain
ER	Error occurred; value returned is meaningless

The UN qualifier is returned for the following conditions:

- Attempted a timing measurement when the measurement zone of the selected waveform contained null (unacquired) values.
- Attempted a FALltime?, FREq?, PERiod?, RISetime?, WIDth?, or an area/energy measurement when the waveform description for the selected waveform is enveloped or contains enveloped components.
- Attempted a MEAN? or RMS? measurement when DAInt was set to SINgle and the waveform description of the selected waveform was enveloped or contained enveloped components.

The ER qualifier is returned for the following conditions:

- Attempted FREq? or PERiod? measurement and no period was found within the specified measurement zone.
- Attempted a MEAN?, RMS?, YTPls_area?, YTMns_area?, or YTEnergy? measurement when DAInt was SINgle and no period was found within the specified measurement zone.
- Attempted a CROss? measurement and no transition of the specified slope was found.
- Attempted a CROss? measurement and REFLevel did not fall within the computed MAX and MIN of the specified measurement zone.
- Attempted a RISetime? measurement and the measurement system could not compute a valid PROXimal time, followed by a valid DISTal time, within the specified measurement zone.
- Attempted a FALltime? measurement and the measurement system could not compute a valid DISTal time followed by a valid PROXimal time, within the specified measurement zone.

- Attempted a WIDTh? measurement and two MESial crossings of opposite slope could not be found within the specified measurement zone.
- Attempted a GAIn?, PDElay?, or PHAse? measurement when only one waveform was defined.
- Attempted any measurement when the selected waveform was an XY waveform or in infinite or variable persistence mode.
- Attempted a frequency domain measurement on a non-FFTmag waveform.
- Attempted any measurement when no waveforms were displayed.

Examples

`MEAS?`

executes the measurement commands in MSList and returns the results, such as:

```
MEAS MEAN:7.3333E-4,EQ,CROSS:7.6685E-4,EQ
```

Binary Block Format

Measurement data can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

<bblock> ::= <byte cnt><double><qual><chksum>

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<double> ::= an 8 byte binary value in 64-bit IEEE floating point format.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values and their ASCII qualifiers.

<chksum> ::= (as defined for CURVe.)

The <qual> values and their ASCII qualifiers are listed below:

<qual> Measurement Values

Hex Value	ASCII qualifier	Hex Value	ASCII qualifier
00	ER	0B	UN
01	EQ	0C	UN
02	LT	0D	UN
03	GT	0E	UN
04	LT	0F	UN
05	GT	10	UN
06	LT	11	UN
07	GT	12	UN
08	GT	13	ER
09	UN	14	ER
0A	UN	15	ER

<meas>

Query Only. *<meas>*? is shorthand for a query of any of the measurements listed below. Querying a specific measurement executes the measurement and returns its value followed by an accuracy qualifier. (Refer to the MEAS? command for the list of qualifiers.)

Syntax: `<meas>?`

The *<meas>* measurements are listed by function below:

<meas> Measurement Types

Amplitude	Area/Energy	Frequency Domain	Timing/ Frequency
GAIn	YTEnergy	SFReq	CROss
MAX	YTMns_area	SMAg	DELay
MEAN	YTPLs_area	THD	DUTy
MID			FALltime
MIN			FREq
OVErshoot			PDElay
PP			PERiod
RMS			PHAse
UNDershoot			RISetime
			SKEW
			TTRig †
			WIDth

† TTRig? sends event code 463, "Measurements complete," when it is queried or MEAS? is queried and TTRig is on the measurement list.

Refer to each measurement entry for information.

Examples

`MEAN?`

returns the the average amplitude of the selected waveform followed by a qualifier, such as:

```
MEAN 7.3333E-4,EQ
```

MESial

MESial sets the mesial (middle) reference level (i.e., the endpoint of the waveform period) for DELAy?, DUTy?, FREq?, MEAN?, PERiod?, PDElay?, PHAse?, RMS?, SKEW?, and WIDth? measurements; and when DAInt is set to SINgle, for YTEnergy?, YTMns_area?, YTPls_area? measurements.

Syntax:
```
MESial<sp><NRx>
MESial?
```

Range: MESial range depends on the current argument to MLEvel. When MLEvel is RELative, the range is a percentage of the difference between the TOPline and BASeline values. When MLEvel is ABSOlute, the range is in vertical units of the selected waveform:

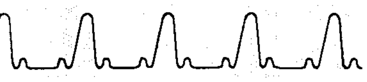
MESial Ranges

With MLEvel RELative	With MLEvel ABSOlute
0 to 100 %	-5.0E+20 to +5.0E+20

The MESial range when the MLEvel argument is BASEDelta or TOPDelta is the same as MLEvel ABSOlute.

Examples

`MES 50`
sets the middle reference level.



MID

Query Only. MID returns the amplitude midpoint, halfway between the maximum amplitude and the minimum amplitude of the selected waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `MID?`

Returns: `<NR3> or <bblock>`

Examples

`MID?`
returns the amplitude midpoint, such as:
`MID 2.2E-1,EQ`

MIN

Query Only. MIN returns the minimum amplitude (most negative peak voltage) of the selected waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `MIN?`

Returns: `<NR3> or <bblock>`

Examples

`MIN?`
returns the minimum amplitude, such as:
`MIN -6.398E-2,EQ`

MKDIR

Set only. MKDIR creates a new directory on the floppy disk.

Syntax: `MKDIR<sp><qstring>`

Arguments

- **<qstring>** – specifies the name of the new directory. If the full pathname is not specified the directory is created within the current working directory.

Examples

```
MKDIR: "A:\HARDCPY\NDATA"
```
makes a directory called NDATA within the directory A:\HARDCPY.

MLEvel

MLEvel controls how ranges are determined for DISTal, MESial, and PROXimal commands.

Syntax:
```
MLEvel<sp>{ABSOlute|BASEDelta|RELative|
                TOPDelta}
MLEvel?
```

Arguments

- **ABSOlute** – makes the DISTal, MESial, and PROXimal ranges absolute values scaled in vertical units (typically volts) of the selected waveform.
- **BASEDelta** – makes DISTal, MESial, and PROximal ranges "delta" values which are added to the current BASeline value to give the DISTal, MESial, or PROXimal value used for measurements. BASEDelta is an absolute value scaled in vertical units.
- **RELative** – makes DISTal, MESial, and PROXimal ranges a percentage of the difference between the current TOPline and BASeline values.

- **TOPDelta**—makes DISTal, MESial, and PROximal ranges "delta" values which are added to the current TOPline value to give the DISTal, MESial, or PROXimal value used for measurements. TOPDelta is an absolute value scaled in vertical units.

Here are some examples, assuming BASeline is 0 V and TOPline is 10 V:

Examples of MLEvel Usage

MLEvel Argument	Desired Parameter	Command To Use
RELative	MESial 4.5 V	MESial 45
ABSOlute	MESial 4.5 V	MESial 4.5
TOPDelta	PROximal 1.1 V	PROximal -8.9
BASEDelta	DISTal 8.7 V	DISTal 8.7

Examples

`MLE ABSO`
selects ranges that are absolute values.

MSCount

MSCount specifies the number of samples to be used in computing all measurement statistics.

Syntax: `MSCount<sp><NRx>`
`MSCount?`

Range: `<NRx>` = 2–5000

Note: Intermediate results are not computed. Each time a statistics query is entered, the entire MSCount number of samples will be acquired and the computations completed before results are returned to the interface.

Examples

`MSC 10`
specifies that 10 samples will be used when calculating measurement statistics.

MSList

MSList selects up to six measurements (<meas>) that are executed continuously in the Measure major menu. (The values of these measurements are returned with a MEAS? query.) EMPty deletes all measurements from the list and all measurements are cleared from the Measure major menu.

Syntax:
```
MSList<sp>{<meas>[<ui>][,<meas>[<ui>]...]
          |EMPty}
MSList?
```

<meas> Measurement Types

Amplitude	Area/Energy	Frequency Domain	Timing/ Frequency
GAIn	YTEnergy	SFReq	CROss
MAX	YTMns_area	SMAg	DELay
MEAN	YTPLs_area	THD	DUTy
MID			FALltime
MIN			FREq
OVErshoot			PDElay
PP			PERiod
RMS			PHAse
UNDershoot			RISetime
			SKEW
			TTRig †
			WIDth

† *TTRig? sends event code 463, "Measurements complete," when it is queried or MEAS? is queried and TTRig is on the measurement list.*

Examples

`MSLI PP,FRE,WID,PER`

executes the peak-to-peak, frequency, width, and period measurements on the selected waveform.

MSLOpe

Set Only. MSLOpe sets the crossing slope for the CROss measurement.

Syntax: `MSLOpe<sp>{MINUs|PLUs}`

Examples

```
MSLO PLU
```

selects a positive crossing slope.

MS<meas>

Query Only. MS<meas> returns the measurement statistics (minimum, maximum, mean, and standard deviation) of the measurement specified by <meas>. STATIstics must be set to ON. Completion of MS<meas>? is signaled with event code 463, "Measurements completed." Output encoding is determined by the ENCDG MEAS command.

Syntax: `MS<meas>?`

Returns: `<NR3> or <bblock>`

<meas> Measurement Types

Amplitude	Area/Energy	Frequency Domain	Timing/ Frequency
GAIn	YTEnergy	SFReq	CROss
MAX	YTMns_area	SMAg	DELay
MEAN	YTPLs_area	THD	DUTy
MID			FALltime
MIN			FREq
OVErshoot			PDElay
PP			PERiod
RMS			PHAse
UNDershoot			RISetime
			SKEW
			TTRig
			WIDth

Note: Intermediate results are not computed. Each time MS<meas>? is entered, the required number of samples is acquired and the computations completed before results are returned.

Examples

`MSRMS?`

returns the measurement statistics for RMS, such as:

```
MSRMS 5.085E+0,EQ,5.116E+0,EQ,5.102E+0,EQ,
    5.976E-3,EQ
```

Measurement statistics data can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

<bblock> ::= <byt cnt> <min> <qual> <max> <qual>
<mean> <qual> <std> <qual> <chksum>

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<min> ::= (minimum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values.

<max> ::= (maximum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<mean> ::= (mean of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<std> ::= (standard deviation of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<chksum> ::= (as defined for CURVe.)

MSNum

Query Only. MSNum returns the number of items in the current MSList. The range is 0 to 6 items.

Syntax: `MSNum?`

Returns: `<ui>`

Examples

`MSN?`

returns the number of items in the measurement list, such as:

`MSNUM 4`

MSREP<meas>

Set only. MSREP<meas> generates measurement statistics (minimum, maximum, mean, and standard deviation) for the measurement specified by *<meas>* for repeated single shot acquisitions. Output encoding is determined by the ENCDG MEAS command.

Syntax: `MSREP<meas><sp>START`

Returns: The measurement values are returned in the following form:

MSREP*<meas> <NR3>*,*<qual>*,*<NR3>*,
<qual>,*<NR3>*,*<qual>*,*<NR3>*,
<qual>

or <bblock>

The order of the values is minimum, maximum, mean, and standard deviation, followed by the number of the stored waveforms that produced the maximum and minimum values. Each value is followed by a qualifier.

Arguments

- **START** – starts acquisition. On each trigger, the trace specified by the SELECT link of the REPMEAS command (and any other trace required by the specified measurement) will be acquired. Measurement statistics will be generated on groups of acquisitions. The number of acquisitions in each statistical group is specified by the MSCount command. Acquisitions stop when the count specified by REPMEAS NREPMeas command is reached, or when the instrument receives a DCL. Total number of acquisitions is equal to MSCount times REPMEAS NREPMeas.

Examples

```
MSREPPP START
```

Measurement statistics data for repeated single shot acquistions can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

```
<bblock> ::= <byt cnt> <min> <max> <mean> <std>
             <qual> <chksum>
```

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<min> ::= (minimum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<max> ::= (maximum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<mean> ::= (mean of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<std> ::= (standard deviation of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values.

<chksum> ::= (as defined for CURVe.)

MSREPmeas

Set only. MSREPmeas generates measurement statistics (minimum, maximum, mean, and standard deviation) of the measurements specified by the current MSLIST for repeated single shot acquisitions. Output encoding is determined by the ENCDG MEAS command.

Syntax: `MSREPmeas<sp>START`

Returns: The measurement values are returned in the following form:

MSREP <*meas*> <*NR3*>, <*qual*>, <*NR3*>, <*qual*>, <*NR3*>, <*qual*>, <*NR3*>, <*qual*>

or <bblock>

The order of the values is minimum, maximum, mean, and standard deviation, followed by the number of the stored waveforms that produced the maximum and minimum values. Each value is followed by a qualifier.

The order of the values is minimum, maximum, mean, and standard deviation. Each value is followed by a qualifier. For a list of qualifier definitions refer to page 192.

Arguments

- **START** – starts acquisition. On each trigger, the trace specified by the SELECT link of the REPMEAS command (and any other trace required by the specified measurements) will be acquired. Measurement statistics will be generated on groups of acquisitions. The number of acquisitions in each statistical group is specified by the MSCount command. Acquisitions stop when the count specified by REPMEAS NREPMeas command is reached, or when the instrument receives a DCL. Total number of acquisitions is equal to MSCount times REPMEAS NREPMeas. At the end of each acquisition event code 463 is given "Measurements complete." When MSREPMEas is complete, event code 450 is given "Conditonal acquire complete."

Examples

`MSREP START`
starts measurement statistics for repeated single-shot acquisitions.

Measurement statistics data for repeated single shot acquistions can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

<bblock> ::= <byt cnt> <min> <max> <mean> <std>
<qual> <chksum>

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<min> ::= (minimum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<max> ::= (maximum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<mean> ::= (mean of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<std> ::= (standard deviation of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values.

<chksum> ::= (as defined for CURVe.)

MSTat

Query Only. MSTat returns the measurement statistics (minimum, maximum, mean, and standard deviation) of the measurement(s) on the measurement list (MSList). STATIstics must be set to ON. Completion of MSTat? is signaled with event code 463, "Measurements completed."

Syntax: `MSTAT?`

Note: Intermediate results are not computed. Each time MSTat? is entered, the required number of samples is acquired and the computations completed before results are returned.

Examples

```
MSTAT?
```

returns the statistics for all measurements in MSList, such as:

```
MSTAT RMS:5.085E+0,EQ,5.116E+0,EQ,
    5.102E+0,EQ,5.976E-3,EQ,OVERSHOOT:
    0.0E+0,EQ,1.429E+0,EQ,5.991E-1,EQ,
    3.432E-1,EQ,
```

MSTO

FROM
TO
USING

MSTO defines parameters for making measurements on groups of stored waveforms.

Syntax:
```
MSTO<sp><link>:<arg>
MSTO?[<sp><link>]
```

FROM

FROM specifies the starting stored waveform. This must be an existing stored waveform.

Syntax:
```
MSTO<sp>FROM:<ui>
MSTO?<sp>FROM
```

Range: <ui> = 1 to 420† or 1 to 918 ‡

† *The range without a disk drive is 1 to 453.*

‡ *The range with Option 4C, Nonvolatile RAM, installed.*

Examples

```
MSTO FROM:1
```

TO

TO specifies the ending stored waveform. This must be an existing stored waveform.

Syntax:
```
MSTO<sp>TO:<ui>
MSTO?<sp>TO
```

Range: <ui> = 1 to 420† or 1 to 918 ‡

† *The range without a disk drive is 1 to 453.*

‡ *The range with Option 4C, Nonvolatile RAM, installed.*

Examples

```
MSTO TO:10
```

USING

USING specifies a list of stored waveforms to be measured. ALL indicates that all stored waveforms will be used. If a <*qstring*> is given, all waveforms with a baselabel that matches <*qstring*> will be used. Only the alpha character portion of the baselabels will be matched. <*qstring*> must not contain digits. To change baselabel names refer to the BASELabel command.

Syntax:
```
MSTO<sp>USING:{ALL|<qstring>}
MSTO?<sp>USING
```

Examples

```
MSTO USING:"REP"
```

MSTO<meas>

Query only. MSTO<*meas*> causes the specified measurement to be made on the specified list of stored waveforms (see MSTO). Output encoding is determined by the ENCDG MEAS command.

Syntax: `MSTO<meas>?`

Returns: The measurement values are returned in the follow form:

MSTO<*meas*> <*NR3*>,<*qual*>,<*NR3*>,
<*qual*>,<*NR3*>,<*qual*>,<*NR3*>,
<*qual*>,<*NR1*>,<*NR1*>

or <bblock>

The order of the values is minimum, maximum, mean, and standard deviation, followed by the number of the stored waveforms that produced the maximum and minimum values. Each value is followed by a qualifier. Any measurement can be made on stored waveforms except the TTRig.

Examples

```
MSTOPHASE?
```

returns the phase measurement on stored waveforms (in the specified list), such as:

```
MSTOPHASE
    3.597E+2,EQ,-1.149E+2,EQ,-8.157+1,EQ,
    -2.476+2,EQ,3,1
```

Measurement statistics data for stored waveforms can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

<bblock> ::= <byt cnt> <min> <qual> <max> <qual>
<mean> <qual> <std> <qual> <sto max>
<sto min> <chksum>

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<min> ::= (minimum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<max> ::= (maximum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<mean> ::= (mean of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<std> ::= (standard deviation of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<sto min> ::= (the stored waveform number that measured the minimum value) a 2 byte unsigned integer.

<sto max> ::= (the stored waveform number that measured the maximum value) a 2 byte unsigned integer.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values.

<chksum> ::= (as defined for CURVe.)

MSTOMEAS

Query only. MSTOMEAS causes the measurements specified by MSLIst to be made on the specified list of stored waveforms (see MSTO). Output encoding is determined by the ENCDG MEAS command.

Syntax: `MSTOMEAS?`

Returns: The measurement values are returned in the follow form:

MSTOMEAS <*meas*> <*NR3*>,<*qual*>, <*NR3*>, <*qual*>,<*NR3*>,<*qual*>,<*NR3*>, <*qual*>,<*NR1*>,<*NR1*>
[<*meas*> <*NR3*>,<*qual*>, <*NR3*>, <*qual*>,<*NR3*>,<*qual*>,<*NR3*>, <*qual*>,<*NR1*>,<*NR1*>...]

or <bblock>

The order of the values is minimum, maximum, mean, and standard deviation, followed by the number of the stored waveforms that produced the maximum and minimum values. Each value is followed by a qualifier. For a list of qualifier definitions refer to page 192. Any measurement can be made on stored waveforms except the TTRig.

Examples

```
MSTOMEAS?
```

returns the measurements specified by MSLIst to be made on the specified list of stored waveforms, such as:

```
MSTOPMEAS PHASE 3.597E+2,EQ,-1.149E+2,EQ,
    -8.157+1,EQ,-2.476+2,EQ,3,1
```

Measurement statistics data for stored waveforms can also be output in binary block <bblock> format. The <bblock> measurement data is defined as followed:

<bblock> ::= <byt cnt> <min> <qual> <max> <qual>
<mean> <qual> <std> <qual> <sto max>
<sto min> <chksum>

where

<byte cnt> ::= a two byte binary integer (MSB first) giving the length (in bytes) of the <bblock>, including the checksum.

<min> ::= (minimum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<max> ::= (maximum measurement value) an 8 byte binary value in 64-bit IEEE floating point format.

<mean> ::= (mean of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<std> ::= (standard deviation of all measurement values) an 8 byte binary value in 64-bit IEEE floating point format.

<sto min> ::= (the stored waveform number that measured the minimum value) a 2 byte unsigned integer.

<sto max> ::= (the stored waveform number that measured the maximum value) a 2 byte unsigned integer.

<qual> ::= a 1 byte binary qualifier value. See the table on page195 for qualifier values.

<chksum> ::= (as defined for CURVe.)

MSYs

MSYs sets the measurement system ON or OFF at the front panel display. In effect, MSYs presses the front panel **MEASURE** button. Whether MSYs is ON or OFF has no effect on measurements taken with MEAS? or if you query a specific measurement.

Syntax:
```
MSYs<sp>{OFF|ON}
MSYs?
```

Set MSYs to ON when you need to use the front panel in conjunction with remote commands (e.g., semi-automatic ATE applications). Set MSYs to OFF for faster remote system throughput.

Examples

`MSY OFF`
turns the measurement system off at the front panel.

MTIme

MTIme determines the left and right measurement zone operation modes.

Syntax:
```
MTIme<sp>{ABSOlute|RELative}
MTIme?
```

Arguments

- **ABSOlute**—scales the LMZone and RMZone values in units of the horizontal time base.
- **RELative**— sets the LMZone and RMZone values as a percentage of the waveform record.

Examples

`MTI REL`
sets the zone values as a percentage of the trace record.

MTRack

MTRack controls measurement tracking.

Syntax: `MTRack<sp>{BASeline|BOTh|OFF|ON|TOPline}`
`MTRack?`

Arguments

- **BASeline** – the DSA determines the BASeline and you set the TOPline value.
- **BOTh** – the DSA determines both BASeline and TOPline values.
- **OFF** – you set both BASeline and TOPline values.
- **ON** – may be substituted for BOTh when MTRack is used to set measurement tracking, but the query MTRack? will return BOTh.
- **TOPline** – the DSA determines the TOPline and you set the BASeline value.

Examples

`MTR OFF`
lets the user set BASeline and TOPline values.

NAVg

NAVg sets the number of waveform samples to be averaged when averaging is enabled either in the waveform description (refer to the TRAce command) or as an acquisition condition (refer to the CONDacq command).

Syntax:
```
NAVg<sp><NRx>
NAVg?
```

Range: `<NRx>` = 2 to 65534

Examples

`NAV 50`
specifies the number of trace samples that are averaged.

NENV

NENV sets the number of waveform samples to be enveloped when enveloping is enabled either in the waveform description (refer to the TRAce command) or as an acquisition condition (refer to the CONDacq command).

Syntax:
```
NENV<sp><NRx>
NENV?
```

Range: `<NRx>` = 2 to 4096

Examples

`NEN 300`
sets the number of trace samples that are enveloped.

NEXTFps

NEXTFps sets the next index for stored front panel settings. The SETDev setting (disk or RAM) affects the index range. The number of characters in the BASEName affects the disk index range, the disk index range shown is with a default BASEName of three characters.

Syntax:
```
NEXTFps<sp><NRx>
NEXTFps?
```

RAM Range: `<NRx>` = 1 to 20

Disk Range: `<NRx>` = 1 to 99999

Examples

`NEXTF 300`
sets the next stored setting index to 300.

NEXTSto

NEXTSto sets the next index for stored waveforms. The SETDev setting (disk or RAM) affects the index range.The number of characters in the BASEName affects the disk index range, the disk index range shown is with a default BASEName of three characters.

Syntax:
```
NEXTFps<sp><NRx>
NEXTFps?
```

RAM Range: `<NRx>` = 1 to 918†

† The range with Option 4C, Nonvolatile RAM, installed.

Disk Range: `<NRx>` = 1 to 99999

Examples

`NEXTS 123`
sets the next stored waveform index to 123.

NHISt.pt

NHISt.pt sets the number of points that must be acquired in a histogram to stop conditional acquisition (refer to the CONDacq TYPe:HIST.pt command).

Syntax:
```
NHISt.pt<sp><NRx>
NHISt.pt?
```

Range: `<NRx>` = 1 to 4294967295

Examples

`NHIS 330`
specifies that 330 points must be acquired before acquisition is stopped.

NREptrig

NREptrig sets the number of repetitive triggers to be acquired when CONDacq TYPe is set to REPtrig.

Note: NREptrig value is ignored when AUTOAcq MEMWrap is set to ON.

Syntax:
```
NREptrig<sp><NRx>
NREptrig?
```

Range: Minimum NREptrig value is 1. Maximum value depends on whether Option 4C, Nonvolatile RAM, is installed. If Option 4C is installed, maximum is 918. If Option 4C is not installed, maximum is 416 (or 449 if the disk drive is not installed).

Examples

`NRE 500`
Specifies 500 waveforms must be acquired before acquistion is stopped.

NVRam

Query Only. NVRam returns the number of bytes of unallocated nonvolatile RAM (NVRam) available for storing front panel settings, or if option 4C is installed, for storing front panel settings and waveforms.

Syntax: `NVRam?`

Returns: `<NR1>`

Examples

```
NVR?
```

returns the number of available bytes, such as:

```
NVRAM 104723
```

NWAVfrm

NWAVfrm sets the number of waveforms that must be processed into histograms to stop conditional acquisition (refer to the CONDacq TYPe:WAVfrm command).

Syntax:

```
NWAVfrm<sp><NRx>
NWAVfrm?
```

Range: `<NRx>` = 1-4294967295

Examples

```
NWAV 100
```

set the number of traces that will be processed before acquisition is stopped.

OPTIONS

Query Only. OPTIONS returns the number of options installed, and if more than one, returns a *<qstring>* list of the options delimited by commas.

Syntax: `OPTIONS?`

Examples

`OPTIONS?`
returns the number of options installed, such as:
`OPTIONS 1,"Option 4C - Non-volatile RAM"`

OUTput

Set Only. OUTput selects the source of data returned by CURVe, WAVfrm?, or WFMpre? queries. The power-on default is STO1.

Syntax: `OUTput<sp>{STO<ui>|TRAce<ui>|<qstring>}`

Range: `STO<ui>` = 1 to 420† or 1 to 918 ‡
`TRAce<ui>` = 1 to 8

† *The range without a disk drive is 1 to 453.*

‡ *The range with Option 4C, Nonvolatile RAM, installed.*

Arguments

- **STO<ui>** – identifies the data source as the specified stored waveform.
- **TRAce<ui>** – identifies the data source as the specified displayed waveform.
- **<qstring>** – identifies the data source as the specified labeled waveform. If the label matches both a stored waveform and a displayed waveform, the displayed waveform is used by OUTput. Wildcards are legal. All displayed and stored waveforms with labels matching <qstring> will be output for WFMPRE? or CURve? queries from the lowest to highest numbered displayed trace, followed by the lowest to highest numbered stored waveform.

Examples

```
OUT 'CTRL44'
```
specifies the data source.

OVErshoot

Query Only. OVErshoot returns the difference between the maximum signal amplitude and the TOPline value, given as a percentage of the difference between the TOPline and BASeline values, and followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `OVErshoot?`

Returns: `<NR3> or <bblock>`

Examples

```
OVE?
```
returns the difference between the maximum amplitude and the topline value, such as:

```
OVERSHOOT 6.221E-1,EQ
```

PATh

PATh controls whether queries return headers, links, and arguments or just arguments. When PATh is set to OFF, only the arguments are returned to a query. The default state is PATh ON.

Syntax:
```
PATh<sp>{OFF|ON}
PATh<sp>?
```

- PATh does not affect the ASCII or binary SET? query response. Headers and links are returned regardless of the setting of PATh.
- When PATh is set to OFF, some commands continue to return their links for clarity (for example, STONum?, DIAg?, FPSList?).
- When PATh is set to OFF, data returned from a query is not acceptable as set command input and will generate error(s) if returned to the DSA.

Examples

`PAT ON`

specifies that headers, links, and arguments will be included in query responses.

PDElay

Query Only. PDElay returns the propagation delay between MESial crossings of the selected waveform and the waveform specified with the DLYtrace command, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `PDElay?`

Returns: `<NR3> or <bblock>`

Examples

`PDE?`

returns the propagation delay, such as:

```
PDELAY 6.9E-11,EQ
```

PERiod

Query Only. PERiod returns the time taken for one complete signal cycle, defined by the MESiaL crossing level, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) PERiod is the reciprocal of the frequency (FREq). Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `PERiod?`

Returns: `<NR3> or <bblock>`

Examples

`PER?`

returns the time required for one complete cycle, such as:

```
PERIOD 9.766E-7,EQ
```

PHAse

Query Only. PHAse returns the phase relationship (from 0 to 360°) of the selected waveform to the reference waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `PHAse?`

Returns: `<NR3> or <bblock>`

Examples

```
PHA?
```

returns the phase of the selected waveform, such as:

```
PHASE 1.064E+2,EQ
```

PIN8

FORMat
PORt
SECUre

PIN8 specifies parameters for printers that support standard Epson 8-pin Bit Image Graphics commands, such as the Tektronix 4644 and Epson EX-800.

Syntax:
```
PIN8<sp><link>:<arg>
PIN8?[<sp><link>]
```

FORMat

FORMat selects the output format.

Syntax:
```
PIN8<sp>FORMat:{DRAft|HIRes|REDuced}
PIN8?<sp>FORMat
```

Arguments

- **DRAft**—prints black-on-white background except for selected icons or text which are printed white-on-black background.
- **HIRes**—shows front panel intensified regions by dithering icon and text backgrounds and increasing foreground saturation.

- **REDuced** – is a quarter-size version of DRAft and prints black-on-white background only.

Note: Use FORMat:HIRes for IBM Proprinter and Epson RX80 printers.

Examples

`PIN8 FORM:DRA`
selects a print format that prints both black-on-white and white-on-black.

PORt

PORt specifies the output port for the plotter.

Syntax:
```
PIN8<sp>PORt:{CENTRonics|GPIb|RS232
                          |DISk}
PIN8?<sp>PORt
```

Examples

`PIN8 POR:CENTR`
selects the Centronics port for output. CENTRonics is the default.

SECUre

SECUre, when set to ON, specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax:
```
PIN8<sp>SECUre:{ON|OFF}
PIN8?<sp>SECUre
```

Examples

`PIN8 SECU:OFF`
Turns off the SECUre function.

PIN24

FORMat
PORt
SECUre

PIN24 specifies parameters for printers that support extended Epson 24-pin Dot Graphics commands, such as the Epson LQ-1500.

Syntax:
```
PIN24<sp><link>:<arg>
PIN24?[<sp><link>]
```

FORMat

FORMat selects the output format.

Syntax:
```
PIN24<sp>FORMat:{DRAft|HIRes|REDuced}
PIN24?<sp>FORMat
```

Arguments

- **DRAft**—prints black-on-white background except for selected icons or text which are printed white-on-black background.
- **HIRes**—shows front panel intensified regions by dithering icon and text backgrounds and increasing foreground saturation.
- **REDuced**—is a quarter-size version of DRAft and prints black-on-white background only.

Examples

`PIN24 FORM:RED`
prints black-on-white only.

PORt

PORt specifies the output port for the plotter.

Syntax:
```
PIN24<sp>PORt:{CENTRonics|GPIb|RS232
                |DISk}
PIN24?<sp>PORt
```

Examples

`PIN24 POR:GPI`
sets the GPIB port for output. CENTRonics is the default.

SECUre

SECUre specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax: `PIN24<sp>SECUre:{ON|OFF}`
`PIN24?<sp>SECUre`

Examples

`PIN24 SECU:OFF`
Turns off the SECUre function.

PINdex

PINdex selects the peak index on which the SMAgnitude and SFRequence measurements are made when SMOde is set to PEAK.

Syntax: `PINdex<sp><ui>`
`PINdex?`

Range: `<ui>` = 1 to 1000

Examples

```
PIN 530
```
sets the peak index.

PIVersion

Query Only. PIVersion returns identifying information about plug-in unit firmware version numbers. If a plug-in compartment is empty, it returns "N/7K."

Syntax: `PIVersion?`

Examples

```
PIV?
```
returns the firmware version number of the plug-in unit, such as:

```
PIVERSION LEFT:"3.7",CENTER:"3.7",RIGHT:"N/7K"
```

POWeron

Query Only. POWeron returns the total number of times the DSA has been powered on.

Syntax: `POWeron?`

Examples

`POW?`

returns the number of times the unit has been powered on, such as:

```
POWERON 149
```

PP

Query Only. PP returns the peak-to-peak voltage value (i.e., the difference between the MAX? and MIN? measurement values), followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `PP?`

Returns: `<NR3> or <bblock>`

Examples

`PP?`

returns peak-to-peak voltage, such as:

```
PP 5.72E-1,EQ
```

PROBe

PROBe selects the function performed when the ID button of an 11000 Series probe is pressed.

Syntax: `PROBe<sp>{NT|NTAuto|SETSeq|STO}`
`PROBe?`

Arguments

- **NT** – selects either a displayed waveform that includes the probe input channel or, if no displayed waveform includes the probe channel, creates a new waveform that contains only the probe channel.
- **NTAuto** – is similar to PROBe NT except that AUTOSet is executed on the selected waveform or on the new waveform created.
- **SETSeq** – causes a probe button press to recall the next set of stored front panel settings from memory. You can sequentially recall all stored settings by repeated button presses.
- **STO** – causes a button press to store in memory all traces with the same plug-in channel component as the probe channel. If no trace matches the probe, a new trace is created and then stored.

Examples

`PROB NTA`
executes an AUTOSet on the selected waveform.

PROXimal

PROXimal sets the proximal (near to origin) level for RISetime? and FALltime? measurements.

Syntax:
```
PROXimal<sp><NRx>
PROXimal?
```

Range: `<NRx>` depends on the current argument to MLEvel. When MLEvel is RELative, the range is a percentage of the difference between the TOPline and BASeline. When MLEvel is ABSOlute, the range is in vertical units of the selected waveform.

PROXimal Ranges

With MLEvel RELative	With MLEvel ABSOlute
0 to 100%	-5.0E+20 to +5.0E+20

The PROXimal range when the MLEvel argument is BASEDelta or TOPDelta is the same as for MLEvel ABSOlute.

Examples

`PROX 5`
sets the near-to-origin level.

PZMode

MULTitrace
PIVot

PZMode controls multiple waveform panning and zooming and selects the pivot point for Pan/Zoom.

Syntax:
```
PZMode<sp><link>:<arg>
PZMode? [<sp><link>]
```

MULTitrace

MULTitrace sets multiwaveform Pan/Zoom to ON or OFF.

Syntax:
```
PZMode<sp>MULTitrace:{OFF|ON}
PZMode?<sp>MULTitrace
```

Arguments

- **OFF** – the Pan/Zoom controls affect only the selected waveform.
- **ON** – all waveforms of the same record length on the same graticule share HMAg and HPOsition values. If you Change the HMAg or HPOsition of a selected waveform, all waveforms (on the same graticule) change to the new HMAg or HPOsition settings. This also occurs when multiple windows are combined into one graticule.

Examples

`PZM MULT:ON`
turns on multiwaveform pan/zoom mode.

PIVot

PIVot selects the pivot point for zooming. Changing the pivot point does not change the HMAG value nor the position of any waveforms.

Syntax:
```
PZMode<sp>PIVot:{CENter|LEFt|RIGht}
PZMode?<sp>PIVot
```

Arguments

- **CENter** – selects the center of the display.
- **LEFt** – selects the left side of the display.

- **RIGht**—selects the right side of the display.

Examples

```
PZM PIV:CEN
```
selects the center of the display for the pivot point.

RCAlconstants

RCAlconstants sets or queries the calibration constants of the right plug-in unit.

Syntax: `RCAlconstants<sp><ui>:<NRx>`
`RCAlconstants?[<sp><ui>]`

Range: `<ui>` is the constant (range is plug-in unit specific), `<NRx>` is the value of the constant.

Note: You can only set RCAlconstants after an internal jumper has been installed by a qualified service person.

Examples

```
RCA? 12
```
returns the calibration constant for the right plug-in unit, such as:

```
RCALCONSTANTS 12:-1.011494E-2
```

RECall

Set Only. RECall recalls stored front panel settings from memory or disk. Completion of RECall is signaled with event code 473, "Recall complete."

Syntax: `RECall<sp>{FPNext|FPS<ui>|<qstring>}`

Range: `<ui>` = 1 to 20 for RAM.

Arguments

- **FPS<ui>** – recalls from memory the front panel settings specified by <ui>.
- **FPNext** – recalls from memory the next front panel setting in sequence. (The SETSeq command must be set to ON.)
- **<qstring>** – recalls from memory or disk the front panel settings labeled by <qstring>.

Examples

```
REC FPN
```
brings up the next front panel setting.

RECOVER STO

Set only. RECOVER STO attempts to recover stored waveforms that have been deleted. Stored waveform memory is searched for headers and data.

Syntax: `RECOVER STO`

Note the following:

- The operation can take up to 45 seconds.
- Header/Data integrity is not guaranteed.
- The waveform number is assigned in the order the waveforms are found.
- If an out-of-memory error occurs, remove displayed traces and try again.

Examples

`RECOVER STO`
searches the NVRAM for headers and data that has been deleted.

REFLevel

REFLevel sets the signal reference level for CROss?, YTEnergy?, YTPls_area?, and YTMns_area? measurements.

Syntax:
```
REFLevel<sp><NRx>
REFLevel?
```

Range: `<NRx>` = Any legal value.

Examples

`REFL 55`
sets the signal reference level.

REFset

CURRent
<meas>

REFset sets reference value(s) for comparison measurements returned when COMpare is set to ON. (Refer to the COMpare command.)

Syntax:
```
REFset<sp><link>:<arg>
REFset?[<sp><link>]
```

CURRent

Set Only. CURRent executes the specified measurement (<meas>), and stores the resulting value as the measurement reference.

Syntax: `REFset<sp>CURRent:<meas>`

Note: Completion of REFset CURRent:TTRig is signaled with event code 463, "Measurements complete." No other CURRent argument generates an operation complete.

Examples

`REF CURR:PP`
executes the peak-to-peak measurement.

<meas>

<meas> sets the reference value for the specified measurement.

Syntax: `REFset<sp><meas>:<NRx>`

Range: `<NRx>` = Any legal value.

Query Note: The general REFset? query returns all reference values, whether assigned a reference value or not. A measurement without an assigned reference value returns 0.0E+0.

Examples

`REF PP:2.0`
sets the reference level for the measurement.

REFTrace

REFTrace specifies the reference (delayed) waveform used with the GAIn?, PHAse?, and SKEw? measurements. The reference waveform is used by all three measurements, and is independent of the selected waveform. The measurement is taken *from* the reference waveform *to* the selected waveform. The reference waveform can be the selected waveform. When the reference waveform is the selected waveform, GAIn? returns 1.0, PHAse? returns 0.0, and SKEw? returns 0.0.

Syntax:
```
REFTrace<sp>TRAce<ui>
REFTrace?
```

Range: `<ui>` = 0 to 8, and specifies the waveform.

The valid `<ui>` *setting* range is 1 to 8. However, REFTrace? returns TRAce0 when no waveforms are displayed; REFTrace TRAce0 is ignored when sent back to the DSA.

Changing Measurement Parameters on the Reference Waveform. The GAIn?, PHAse? and SKEw? measurements compare the reference waveform to the selected waveform. Every waveform has its own measurement parameters (e.g., MESial, LMZone) which can be changed only when that waveform is the selected waveform. Therefore, if you need to change measurement parameters on the reference waveform:

1. Use the SELect command to make the reference waveform the selected waveform.

2. Change the measurement parameters.

3. Use the SELect command to reassign the correct selected waveform.

Here is an example of the process of taking a SKEw measurement between TRAce2, the reference waveform, and TRAce4, the selected waveform. The required MESial values are 40% and 45%, respectively.

■ SELect TRAce2	/* Select Trace 2 to change its parameters. */
■ MESial 40	/* Specify its mesial value. */
■ REFTrace TRAce2	/* Make Trace 2 the reference waveform.*/
■ SELect TRAce4	/* Select Trace 4 to change its parameters. */
■ MESial 45	/* Specify its mesial value. */
■ SKEw?	/* Measure SKEw from trace 2 to Trace 4. */

Examples

`REFT TRA2`

selects the reference trace that is used with other measurements.

REMove

Set Only. REMove discards existing data and waveform definitions to remove waveforms from the display. If a waveform is also stored in memory, the stored waveform is not removed. (Use the DELete command to remove stored waveforms.)

Syntax: `REMove<sp>{ALL|TRAce<ui>|<qstring>}`

Range: `<ui>` = 1 to 8, and specifies the waveform.

Arguments

- **ALL** – removes all displayed waveforms. It is not an error to specify ALL when no waveforms are defined.
- **TRAce<ui>** – removes the specified waveform from the display only, not from memory.
- **<qstring>** – The <qstring> argument removes the waveform labeled <qstring> from the display only. Wildcard characters are interpreted. (Refer to page 181 for wildcard definitions.)

Examples

```
REM 'SAMPLE16'
```
removes the data for the waveform labeled 'SAMPLE16.'

RENAme

Set only.RENAme changes file names.

Syntax: `RENAme<sp><qstring>,<qstring>`

Arguments

- **<qstring>** – the first <qstring> is the original filename. The second <qstring> is the new filename. The full pathname must be given, unless the file is in the current working directory.

Note: Files cannot be renamed to different directories; use DCOpy to copy a file from one directory to another.

Examples

`RENA 'SAMPLE16.WFA','SAMPLE23.WFA'`
changes the name of a file in the current directory.

RENDir

Set only.RENDir changes directory names.

Syntax: `RENDir<sp><qstring>,<qstring>`

Arguments

- **<qstring>** – the first <qstring> is the original directory name. The second <qstring> is the new directory name. The full pathname must be given, unless the directory is in the current working directory.

Note: Directories cannot be renamed to different directories.

Examples

`REND 'A:\ICTEST\WAVES','A:\ICTEST\OLDDATA'`
changes the name of a directory in the ICTEST directory.

REPCurve

NREPCurve
REPS

REPCurve controls fast transfer of trace data from the DSA to the controller.

Syntax:

```
REPCurve<sp>{STARt|<link>:<arg>}
REPCurve? [<sp><link>]
```

Arguments

- STARt starts acquisition. On each trigger, the traces specified by the AUTOAcq command will be acquired and transferred over the bus. Acquisitions will stop when either the count specified (by NREPCurve) is reached or when the DSA receives a DCL.

Note: It must be possible to acquire all defined traces concurrently in real time. Therefore, no more than four channels for the DSA 602A or two channels for the DSA 601A may be used in defined traces. The channels which may be used together are also restricted. See the *DSA 601A and DSA 602A User Reference* for information on concurrent acquisition.

Query Responses

- **REPCurve?** – returns the number of current acquisitions made and the number of waveforms transferred.

Examples

```
REPC STAR
```

NREPCurve

NREPCurve specifies the number of acquisitions to be transferred. If 0 is specified, acquisition will continue indefinitely until the DSA receives a DCL.

Syntax:

```
REPCurve<sp>NREPCurve:<NRx>
REPCurve?<sp>NREPCurve
```

Range: `<NRx>` = 0 to 32767

Query Note: REPCurve? NREPCurve returns the current number of acquisitions (the number selected for transfer.)

Examples

```
REPC NREPC:64
```

REPS

Query only. REPS specifies the number of waveforms transferred during the previous REPCurve command.

Syntax:

```
REPCurve?<sp>REPS
```

Examples

```
REPC? REPS:34
```

REP<meas>

Set only. REP makes the specified measurement for repeated single-shot acquisitions.

Syntax: `REP<meas><sp>START`

Arguments

- **START** – starts acquisition. On each trigger, the trace specified by the SELECT link of the REPMEAS command (and any other trace required by the specified measurement) will be acquired and measured. Acquisitions stop when the count specified by REPMEAS NREPMEAS command is reached, or when the instrument receives a DCL. The measurement data is sent to the controller.

Examples

```
REPPP START
```

REPMEAS

NREPMeas
REPS
SELECT
START

REPMEAS controls measurements that are made on repeated single shot acquisitions.

Syntax: `REPMEAS<sp><link>:<arg>`
`REPMEAS? [<sp><link>]`

NREPMeas

NREPMeas specifies the number of acquisitions to be measured. If 0 is specified, acquisitions continue until a DCL is received.

Syntax: `REPMEAS<sp>NREPMeas:<ui>`
`REPMEAS?<sp>NREPMeas`

Range: `<ui>` = 0 to 32767

Examples

```
REPMEAS NREPM:22
```

REPS

Query Only. REPS returns the number of acquisitions that have been made.

Syntax: `REPMEAS?<sp>REPS`

Examples

```
REPMEAS? REPS
```
returns the number of acquisitions, such as:
```
REPMEAS REPS:45
```

SELECT

SELECT:TRAce<ui> specifies the trace to be measured. If necessary, the traces specified by DLYTRACE and REFTRACE will also be acquired.

Syntax: `REPMEAS<sp>SELECT:TRAce<ui>`
`REPMEAS?<sp>SELECT`

Range: `<ui>` = 1 to 8

Examples

```
REPMEAS SELECT:TRA1
```

START

START starts acquisition. On each trigger, the trace specified by the SELECT link of the REPMEAS command (and any other trace required by the specified measurements) will be acquired. Acquisitions stop when the count specified by the REPMEAS NREPMEAS command is reached, or when the instrument receives a DCL.

Note: The traces specified by DLYTRACE and REFTRACE commands must be permitted to be acquired concurrently with the trace specified by the SELECT link, if measurements requiring those traces appear in MSLIST.

Examples

```
REPMEAS START
```

RISetime

Query Only. RISEtime returns the transition time of a rising-pulse edge, from the PROXimal to DISTal level, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `RISetime?`

Returns: `<NR3> or <bblock>`

Examples

`RIS?`

returns the transition time of the rising-pulse edge, such as:

```
RISETIME 7.922E-9,EQ
```

RMDIR

Set only. RMDIR removes a directory from the floppy disk. If the directory is not empty, it cannot be removed.

Syntax: `RMDIR <qstring>`

Range: `<qstring>` is the pathname of the directory to be removed.

Examples

`RMDIR "A:\WAVFORMS"`

removes a directory called "WAVFORMS" from the disk.

RMS

Query Only. RMS returns the true root-mean-square voltage, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `RMS?`

Returns: `<NR3> or <bblock>`

Examples

```
RMS?
```
returns the true root mean square voltage, such as:
```
RMS 3.516E-1,EQ
```

RMZone

RMZone sets the right measurement zone limit.

Syntax:
```
RMZone<sp><NRx>
RMZone?
```

Range: <NRx> depends on the current MTIme value. When MTIme is set to RELative, RMZone is a percentage of the waveform record. When MTIme is set to ABSOlute, RMZone is an absolute position in horizontal units of the selected waveform.

RMZone Ranges

With MTIme RELative	With MTIme ABSOlute
0 to 100%	XZE to (XZE + XIN * (NR.pt -1))

The MTIme ABSOlute range is calculated using XZEro, XINcr, and NR.PT values from the waveform preamble (WFMpre) of the selected trace.

Examples

```
RMZ 75
```
sets the right measurement zone limit.

RQS

RQS determines the DSA response to events detected during DSA operation. The power-on default for GPIB is RQS set to ON.

Syntax: `RQS<sp>{OFF|ON}`
`RQS?`

Arguments

- **OFF** – the DSA does not assert an SRQ after an event.
- **ON** – the DSA asserts SRQ after an event.

Note: RQS is meaningless for the RS-232-C port; the RQS command is always set to OFF for RS-232-C.

Examples

`RQS ON`
asserts an SRQ after an event.

RS232

BAUd
DELAy
ECHo
EOL
FLAgging
PARity
STOPBits
VERBose

RS232 sets parameters for the RS-232-C interface.

Syntax:
```
RS232<sp><link>:<arg>
RS232?[<sp><link>]
```

BAUd

BAUd sets both the transmit and receive baud rates.

Syntax:
```
RS232<sp>BAUd:<NRx>
RS232?<sp>BAUd
```

Range: `<NRx>` = 110, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200

Note: Set the baud rate on the DSA before setting the baud rate on the controller.

Examples

`RS232 BAU:9600`
Sets the RS-232-C transmission rate to 9600 baud.

DELAy

DELAy sets the minimum delay from receipt of a query to its response, with 20 ms granularity.

Syntax:
```
RS232<sp>DELAy:<NRx>
RS232?<sp>DELAy
```

Range: `<NRx>` = 0 to 60 seconds.

Examples

`RS232 DELA:0.5`
sets minimum delay.

ECHo

ECHo determines whether characters are echoed on the controller screen.

Syntax:
```
RS232<sp>ECHo:{OFF|ON}
RS232?<sp>ECHo
```

Note: You cannot send binary data to the DSA when ECHo is set to ON.

Examples

`RS232 ECH:ON`
turns character echo on.

EOL

EOL selects the end-of-line output message terminator:

Syntax:
```
RS232<sp>EOL:{CR|CRLf|LF|LFCr}
RS232?<sp>EOL
```

Arguments

- **CR** – a carriage return.
- **LF** – a line feed.
- **CRLF** – a carriage return followed by a line feed.
- **LFCR** – a line feed followed by a carriage return.

All of the above are accepted as an input message terminator.

Examples

`RS232 EOL:CRL`
sets the message terminator.

FLAgging

FLAgging controls I/O flagging.

Syntax:
```
RS232<sp>FLAgging:{HARd|OFF|SOFt}
RS232?<sp>FLAgging
```

Arguments

- **HARd**—uses the DTR and CTS control lines. Input is halted when the buffer is three-quarters full, and is restarted when the buffer is one-quarter full.
- **OFF**—means there is no transmission control.
- **SOFt**—uses XON (DC1) and XOFF (DC3) handshaking. Input is halted when the buffer is three-quarters full, and is restarted when the buffer is one-quarter full.

Note: SOFt flagging is usually not used with binary transfers because the binary data may contain unintended XON or XOFF controls.

Examples

```
RS232 FLA:SOF
```
turns on XON/XOFF handshaking.

PARity

PARity sets the parity used for all RS-232-C data transfers. The DSA generates parity on output data and checks the parity on input data. An input parity error produces event code 653, "RS-232-C input parity error."

Syntax:
```
RS232<sp>PARity:{EVEN|NONe|ODD}
RS232?<sp>PARity
```

Examples

```
RS232 PAR:EVEN
```
selects even parity.

STOPBits

STOPBits selects the number of transmission stop bits sent with each character to identify the end of data.

Syntax:
```
RS232<sp>STOPBits:<NRx>
RS232?<sp>STOPBits
```

Range: `<NRx>` = 1, 1.5, 2

Examples

```
RS232 STOPB:1.5
```
specifies the number of stop bits transmitted.

VERBose

When VERBose is set to ON, the DSA returns error and warning messages to the controller at the time they occur. When VERBose is set to OFF, the controller must query the DSA for event messages.

Syntax:
```
RS232<sp><VERBose:{OFF|ON}
RS232?<sp><VERBose
```

Examples

```
RS232 VERB:ON
```
turns verbose mode on.

SAVEFactory

Set only. SAVEFactory saves certain calibration constants from RAM to NVRAM in the digitizer. This command is intended for authorized service personnel only, and cannot be done unless an internal hardware connection is made.

Syntax: `SAVEFactory`

Examples

```
SAVEF
```
saves calibration constants in NVRAM.

SCANStowfm

FROm
MODe
RATe
TO
USIng

SCANStowfm controls scanning of stored waveforms.

Syntax:
```
SCANStowfm<sp>{KEEp|NEXt|PREvious|
                    <link>:<arg>}
SCANStowfm?[<sp>{CURRent|NEXt|PREvious|
                    <link>:<arg>}]
```

Arguments

- **KEEp** – causes the current stored waveform to be kept as a displayed waveform.
- **NEXt** – causes the next stored waveform (if any) to become the current waveform and updates the display. When queried, NEXt returns the number of the next stored waveform in the scan list.
- **PREvious** – causes the previous stored waveform (if any) to become the current waveform and updates the display. When queried, PREvious returns the number of the previous stored waveform in the scan list.

Query Responses

- **SCANStowfm?** – returns all information regarding scanning of stored waveforms.
- **SCANStowfm? CURRent** – returns the stored waveform number of the current waveform, or returns -1 if the current waveform is not defined.
- **SCANStowfm? NEXt** – returns the stored waveform number of the next waveform, or returns -1 if the next waveform is not defined.
- **SCANStowfm? PREvious** – returns the stored waveform number of the previous waveform, or returns -1 if the previous waveform is not defined.

Examples

```
SCANS? CURR
```
returns the number of the current waveform, such as:
```
SCANSTOWFM CURRENT:12
```

FROm

FROm specifies the starting stored waveform, which must exist. Event code 229, "No Stored Waveforms," is returned if the specified waveform does not exist.

Syntax:
```
SCANStowfm<sp>FROm:<ui>
SCANStowfm?<sp>FROm
```

Examples

```
SCANS FRO:153
```

MODe

MODe starts or stops stored waveform scanning.

Syntax:
```
SCANStowfm<sp>MODe:{SCAn|STOP}
SCANStowfm?<sp>MODe
```

Examples

```
SCANS MOD:SCA
```

RATe

RATe sets the rate (number of waveforms per second) at which waveforms are scanned.

Syntax:
```
SCANStowfm<sp>RATe:<NRx>
SCANStowfm?<sp>RATe
```

Range: <NRx> = 0.1 to 10

Examples

```
SCANS RAT:2
```

TO TO specifies the ending stored waveform, which must exist. Event code 229, "No Stored Waveforms," is returned if the specified waveform does not exist.

Syntax:
```
SCANStowfm<sp>TO:<ui>
SCANStowfm?<sp>TO
```

Examples

```
SCANS TO:350
```

USIng USIng specifies the list of waveforms to be scanned, either ALL stored waveforms or those whose BASELabel is specified by *<qstring>*.

Syntax:
```
SCANStowfm<sp>USIng:{ALL|<qstring>}
SCANStowfm?<sp>USIng
```

Examples

```
SCANS USI:ALL
```

SCLockd

SCLockd controls whether or not the sample clock is dithered.

Syntax:
```
SCLockd<sp>{DISAble|ENAble}
SCLockd?
```

Arguments

- **DISAble** – maximizes single-shot timing accuracy.
- **ENAble** – improves equivalent time repetitive signal capture (this is the default state).

Examples

`SCL DISA`
disables the sample clock.

SELect

SELect specifies the waveform used by many commands, including AUTOSet, measurement, histogram, and cursor commands. By default, the most recently created waveform is the selected waveform until changed with SELect.

Syntax:

```
SELect<sp>{TRAce<ui>|<qstring>}
SELect?
```

Range: `<ui>` = 0 to 8, and specifies the waveform.

The valid SELect TRAce<ui> *setting* range is 1 to 8. However, SELect? returns TRAce0 when no waveforms are defined. You can send SELect TRAce0 to the DSA without an error; it is ignored.

Arguments

- **TRAce<ui>** – selects the waveform specified by <ui> to be the selected waveform.
- **<qstring>** – designates the waveform labeled with <qstring> as the selected waveform.

Examples

`SEL TRA8`
assigns the selected waveform.

SELFcal

MODe

SELFcal either forces a self-calibration or selects the mode when self-calibration will occur.

Syntax: `SELFcal<sp>{FORce|<link>:<arg>}`
`SELFcal? [<sp><link>]`

Arguments

- **FORce** – causes an immediate self-calibration to occur.

Examples

`SELF FOR`
performs self-calibration immediately.

MODe

MODe selects whether self-calibration is performed automatically when due (always after a five degree change in the internal temperature) or is performed manually using SELFcal FORce.

Syntax: `SELFcal<sp>MODe:{AUTO|MANual}`
`SELFcal?<sp>MODe`

Examples

`SELF MOD:MAN`
lets the user specify when calibration will take place.

SET

SET returns front panel settings to the controller in ASCII or binary format, depending on the state of the ENCdg SET command.

Syntax: `SET?`

Note: SET? is *not* query-only. You can send settings back to the DSA (with some restrictions) to restore a previously defined DSA state. However, the header SET is used only when sending binary data.

ASCII SET? Response. SET? returns strings of DSA commands separated by semicolons.

Binary SET? Response. SET? returns binary data in the following format:

```
<bblock>::= %<byte cnt><settings><checksum>
```

where <byte cnt> is a two-byte integer (MSB first) giving the length in bytes of the remainder of the binary block, including checksum; <settings> are binary-encoded data; and <checksum> is an 8-bit, twos complement of the modulo 256 sum of <byte cnt> and <settings> data.

Sending Settings Back to the DSA. Send settings as a complete set; do not edit or modify the data. For ASCII settings, simply send the entire set of strings. The binary SET? response returns the SET header at the beginning of the response; you must include the SET header when sending binary settings to the DSA. Completion of binary-settings recall is signaled with event code 473, "Front panel recall complete."

Examples

```
SET?
```

returns front panel settings, such as:

```
REM ALL;CHL1 COU:DC,OFFS:0.0E+0,BW:3.5E+8,
    IMP:5.0E+1,PROB:"LEVEL 2/P6231/B011623",
    SEN:1.0E+1,UNI:"VOL";CHL2 COU:DC,
    OFFS:-2.5E-3,BW...
```

SETDev

FPS
STO

SETDev controls whether RAM or DISK storage is accessed. Waveforms and settings may be sent to disk or RAM independently.

Syntax:
```
SETDev<sp><link>:<arg>
SETDev?
```

FPS

FPS controls whether settings are stored in RAM or on disk.

Syntax:
```
SETDev<sp>FPS:{DISK|RAM}
SETDev?<sp> FPS
```

Arguments

- **DISK** – settings will be stored on disk.
- **RAM** – settings will be stored in RAM.

Note: The SETDev command affects the following commands:

Commands Affected By SETDev

Command	SETDEV FPS:RAM	SETDEV FPS:DISK
DELete	Deletes RAM data only.	Deletes files on the disk.
FPSList?	Returns settings stored in RAM.	Returns settings stored on the disk.
FPSNum?	Returns the number of settings stored in RAM.	Returns the number of settings stored on the disk.
NEXTFps	Index range is 1-20	Index range is 1 to 9999999, if BASEName is only 1 character.
STORe	Stores setting in RAM	Stores settings on the disk.

Examples

```
SETD FPS:DISK
```
sets the settings storage device to DISK.

STO

STO controls whether waveforms are stored in RAM or on disk.

Syntax:
```
SETDev<sp>STO:{DISK|RAM}
SETDev?<sp> STO
```

Arguments

- **DISK** – waveforms will be stored on disk.
- **RAM** – waveforms will be stored in RAM.

Note: The SETDev command affects the following commands:

Commands Affected By SETDev

Command	SETDEV STO:RAM	SETDEV STO:DISK
DELete	Deletes RAM data only.	Deletes files on the disk.
NEXTSto	Index range is 1-453 or 918 (*with Option 4C Nonvolatile RAM, installed.)*	Index range is 1 to 9999999, if BASEName is only 1 character.
STOList?	Returns waveforms stored in RAM.	Returns waveforms stored on the disk.
STONum?	Returns the number of waveforms stored in RAM.	Returns the number of waveforms stored on disk.
STORe	Stores waveforms in RAM.	Stores waveforms on disk.

Examples

```
SETD STO:DISK
```
sets the waveform storage device to DISK.

SETSeq

SETSeq controls the sequencing of front panel settings. When SETSeq is set to ON, the settings are sequenced and the RECall FPNext or PROBe SETSeq commands recall the next set of stored front panel settings from memory.

Syntax:
```
SETSeq<sp>{OFF|ON}
SETSeq?
```

Note: If SETSeq is set to ON and all stored settings are deleted, SETSeq is set to OFF. If SETSeq is set to OFF and PROBe SETSeq is issued, SETSeq is set to ON.

Examples

`SETS ON`
sequences front panel settings.

SFReq

Query Only. SFReq returns the spectral frequency (harmonic or spectral peak), followed by an accuracy qualifier. (Refer to page 192 or qualifier definition.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `SFReq?`

Returns: `<NR3> or <bblock>`

Examples

`SFR?`
returns the spectral frequency, such as:
```
SFREQ 2.33E+2,EQ
```

SKEw

Query Only. SKEw returns the propagation (time) delay between MESial crossings of the selected waveform and the reference waveform set with the REFTrace command, followed by an accuracy qualifier. (Refer to page 192 for qualifier definition.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `SKEw?`

Returns: `<NR3> or <bblock>`

Examples

```
SKE?
```
returns the propagation delay, such as:
```
SKEW 4.228E-8,EQ
```

SMAg

Query Only. SMAg returns the spectral magnitude (harmonic or spectral peak), followed by an accuracy qualifier. (Refer to page 192 for qualifier definition.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `SMAgnitude?`

Returns: `<NR3> or <bblock>`

Examples

```
SMA?
```
returns the spectral magnitude, such as:
```
SMAG 2.33E+2,EQ
```

SMOde

SMOde selects whether the SMAgnitude or SFRequency measurements are made on a selected harmonic or spectral peak.

Syntax:
```
SMOde<sp>{HARM|PEAK}
SMOde?
```

Examples

`SMO HAR`
selects harmonic for spectral measurements.

SNRatio

SNRatio sets the signal-to-noise ratio for a noise rejection band for measurements. The reciprocal of the number selected is the fraction of the TOPline-to-BASeline distance the noise-rejection band extends above and below the MESial level.

Syntax:
```
SNRatio<sp><NRx>
SNRatio?
```

Range: <NRx> = 1 to 99

Examples

`SNR 50`
sets the signal-to-noise ratio.

SPEaker

SPEaker controls the DSA audio feedback (i.e., whether you hear a click when you touch the front panel).

Syntax:
```
SPEaker<sp>{OFF|ON}
SPEaker?
```

Examples

`SPE ON`
turns audio feedback on.

SRQMask

SRQMask controls the reporting of selected classes of events, regardless of the state of the RQS command. If an SRQMask link is set OFF, that class of events is not reported. At power-on, all SRQMask links are set to ON, except ABStouch, IDProbe, and USEr. The following table lists all SRQMask links, their meanings, and associated event code(s).

Syntax:
```
SRQMask<sp><link>:{OFF|ON}
SRQMask?[<sp><link>]
```

SRQMask Links

Link	Meaning	Event Code(s)
ABStouch:	Controls reporting of front panel touches either via the ABStouch command or screen touches	451
CALDue:	Controls reporting of calibration-due events	465-472
CMDerr:	Controls reporting of command errors	100-199
EXErr:	Controls reporting of execution errors	200-299
EXWarn:	Controls reporting of execution warnings	500-599
IDProbe:	Controls reporting of probe ID button presses	457
INErr:	Controls reporting of internal errors	300-399
INWarn:	Controls reporting of internal warnings	600-699
OPCmpl:	Controls reporting of operation-complete events	450, 460-464, 473-475
USEr:	Controls whether the RQS icon is displayed and whether RQS icon touches are reported	403

Examples

```
SRQM ABS:ON
```
turns on event reporting for all front panel interactions.

STATHist

Query Only. STATHist provides a number of query links to access the statistical information created by the histogram function. Refer also to the HISTogram command.

Syntax:
```
STATHist? [<sp>{HIST.pt|MEAN|NWFm|PP|
                RSMDev|SIGMA1|SIGMA2|
                SIGMA3}]
```

Arguments

- **STATHist? HIST.pt** – returns the number of sample points processed into the histogram data.
- **STATHist? MEAN** – returns the statistical mean value for the histogram data.
- **STATHist? NWFm** – returns the number of traces processed into the histogram data.
- **STATHist? PP** – returns the peak-to-peak measurement for the histogram data.
- **STATHist? RMSDev** – returns the RMS (standard deviation) value for the histogram data.
- **STATHist? SIGMA1** – returns the percentage of points in the histogram that are within the area that is one STD of the MEAN.
- **STATHist? SIGMA2** – returns the percentage of points in the histogram that are within two STDs of the MEAN.
- **STATHist? SIGMA3** – returns the percentage of points in the histogram that are within three STDs of the MEAN.

Examples

```
STATH? NWF
```
returns the number of traces processed in the histogram data, such as:
```
STATHIST NWFM:197610
```

STATIstics

STATIstics controls whether measurement statistics are computed. When STATIstics is set to ON, measurement statistics are computed and measurement queries return mean values. Also, STATIstics must be ON to use MSTAT?, MS<meas>?, MSREP<meas>, and MSREPMEAS.

Syntax: `STATIstics<sp>{OFF|ON}`
`STATIstics?`

Examples

```
STATI ON
```

computes measurement statistics.

STByte

Query Only, RS-232-C Only. STByte enables an RS-232-C controller to read the status byte of the current RS-232-C event by mimicking a GPIB serial poll at the RS-232-C port. STByte? is not valid at the GPIB port.

Syntax: `STByte?`

Returns: `<ui>`

Note: The status byte is defined in the section on Event Reporting later in this document.

Examples

```
STB?
```

returns the status byte for the RS-232-C port, such as:

```
STBYTE 2
```

STOFmt

STOFmt selects the data format for stored waveforms.

Syntax: `STOFmt<sp>{ASCii|BINary|EKUtil|WORKSheet}`

Arguments

- **ASCii** – selects ASCII format data.
- **BINary** – selects binary format data.
- **EKUtil** – selects 11000 Series Utility Software Package format data.
- **WORKSheet** – selects Lotus format data.

Examples

`STOF ASC`
  selects ASCII data format for stored waveforms.

STOList

Query Only. STOList returns a list of all stored waveforms, or EMPTY if there are no stored waveforms.

Syntax: `STOList?`

Returns: `STO<ui>[,STO<ui>...]`

Examples

`STOL?`
  returns all stored waveforms, such as:
  `STOLIST STO2,STO9,STO56,STO200`

Note: If you get unexpected results, make sure the device is set properly (to RAM or DISK) with the SETDev command.

STONum

Query Only. STONum returns the number of waveforms stored in memory or stored in the current working directory of the disk.

Syntax: `STONum?`

Returns: `<ui>`

Examples

```
STON?
```
returns the number of stored waveforms, such as:

```
STONUM 4
```

STORe

FPS<ui>
TRAce<ui>
<qstring>

Set Only. STORe saves front panel settings (FPS) in nonvolatile RAM or disk. STORe also copies a displayed waveform to memory or disk; the waveform is not removed from the display.

Syntax: `STORe<sp><link>:<arg>`

STORe Constraints: You cannot store an XY waveform. An existing STO<ui> location can be overwritten only if the record lengths of the new and stored waveforms are the same; the previous waveform data is destroyed. If the previously stored waveform was a component of a displayed waveform, the displayed waveform changes to include the newly stored waveform.

FPS<ui>

FPS<ui> stores the current front panel settings using FPS and the specified number as an identifier. The setting is stored in RAM. If SETDev FPS is set to DISK, this link is not valid. If <ui> is an existing FPS number, the new data overwrites the previous data.

Syntax: `STORe<sp>FPS<ui>`

Range: <ui> 1–20 when SETDev FPS is RAM.

Arguments

- **FPS<ui>** – specifies a RAM stored setting number.

Examples

`STOR FPS2`
saves the front panel settings in RAM.

TRAce<ui>

TRAce stores a copy of the TRAce<ui> waveform in memory at the location specified by STO<ui>. If SETDev STO is DISK, then <qstring> specifies a filename. If SETDev STO is RAM, then <qstring> is a label for the waveform.Wildcard characters are not interpreted. If the label or filename identify an existing STO location or file, the new data overwrites the previous data. If the label does not identify an existing STO location, the data is stored in the next available STO location with that label assigned to it.

Note: If <qstring> begins with "A:", the waveform is stored on disk regardless of the status of SETDEV.

Syntax: `STORe<sp>TRAce<ui>:{STO<ui>|<qstring>}`

Range: `TRAce<ui>` = 1 to 8;
`STO<ui>` = 1 to 420†; or if Option 4C, Nonvolatile RAM, is installed, the range is 1 to 918.

† *The range without a disk drive is 1 to 453.*

Examples

STOR TRA1:STO10
saves a copy of waveform 1 in memory.

<qstring>

<qstring> stores the front panel settings on the disk or in RAM depending on the SETDev setting. If SETDev FPS is DISK, then the front panel setting is stored in the file specified by <qstring>. If SETDev FPS is RAM, then <qstring> is a label for the front panel setting.

Note: If <qstring> begins with "A:", the front panel setting is stored on disk regardless of the status of SETDEV.

Syntax: `STORe<sp><qstring>`

Examples

STOR 'A:\SETT.FPB'
saves the front panel settings on the disk.(SETDev FPS is disk.)

STOR 'var1'
the front panel settings is stored in RAM with the label "var1".(SETDev FPS is RAM.)

TBMain TBWin

LENgth
TIMe
XINcr

TBMain sets the Main time base parameters and TBWin sets the Window time base parameters. Both commands use the same links and arguments.

Syntax:
```
TB{Main|Win}<sp><link>:<arg>
TB{Main|Win}?[<sp><link>]
```

LENgth

LENgth sets the selected time base to the specified record length, scaled in points per waveform.

Syntax:
```
TB{Main|Win}<sp>LENgth:<NRx>
TB{Main|Win}?[<sp>LENgth]
```

Range: <NRx> = 512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768

Examples

`TBM LEN:1024; TBW LEN:512`
sets the record length for the main and window time bases.

TIMe

TIMe sets the horizontal scale (time per division).

Syntax:
```
TB{Main|Win}<sp>TIMe:<NRx>
TB{Main|Win}?[<sp>TIMe]
```

Range: <NRx> =50E-12 to 100 sec†

† *Maximum TBWin TIMe cannot exceed (must be less than or equal to) TBMain TIMe*

Examples

`TBM TIM:20E-3; TBW TIM:5.0E-3`
sets the time per division for the main and window time bases.

The following table lists which LENgth values you can use with each TIMe value. (All LENgth values can be used when TIMe is between 100 μs and 100s.)

TIMe & LENgth Requirements

TIMe	LENgth Values
50 ps	512
100 ps	512, 1024
200 ps	512,1024,2048
400 ps	2048
500 ps	512, 1024, 4096, 5120
1 ns	512, 1024, 2048, 4096, 5120, 8192, 10240
2 ns	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464
4 ns	16384, 20464
5 ns	512, 1024, 2048, 4096, 5120, 8192, 10240, 32768
10 ns	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768
20 ns	1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768
25 ns	512
50 ns	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768
100 ns	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768
200 ns	1024, 2048, 8192, 10240, 16384, 20464, 32768
250 ns	4096, 5120
400 ns	1024, 2048
500 ns	512, 4096, 5120, 8192, 10240, 16384, 20464, 32768
800 ns	2048
1 μs	512, 1024, 4096, 5120, 8192, 10240, 16384, 20464, 32768
2 μs	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464
2.5 μs	32768

TIMe & LENgth Requirements (Cont.)

TIMe	LENgth Values
4 µs	2048, 8192, 10240, 16384, 20464
5 µs	512, 1024, 4096, 5120, 32768
8 µs	16384, 20464
10 µs	512, 1024, 2048, 4096, 5120, 8192, 10240, 32768
20 µs	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768
40 µs	16384, 20464
50 µs	512, 1024, 2048, 4096, 5120, 8192, 10240, 32768
100 µs to 100 s	512, 1024, 2048, 4096, 5120, 8192, 10240, 16384, 20464, 32768

XINcr

Query Only. XINcr returns the sample interval of the selected time base, in seconds per point.

Syntax: `TB{Main|Win}?<sp>XINcr`

Returns: `<NR3>`

Calculating Duration. Duration is used when calculating the range of other commands, such as MAINpos.

Use the following formula for *main duration*:

(TBMain XINcr) * (TBMain LENgth – 1)

Use the following formula for *window duration*:

(TBWin XINcr) * (TBWin LENgth – 1)

Examples

```
TBM? XIN; TBW? XIN
```

returns the sample interval of the main and window time bases, such as:

```
TBMAIN XINCR:2.0E-10;TBWIN XINCR:4.0E-9
```

TEK4692

COLor
DIRection
FORMat
PORt
SECUre

TEK4692 specifies parameters for the Tektronix 4692 color graphics copier and Tektronix 4693D color wax printer operating in 4692 emulation mode.

Syntax:
```
TEK4692<sp><link>:<arg>
TEK4692?[<sp><link>]
```

COLor

COLor assigns copier colors to the DSA color index.

Syntax:
```
TEK4692<sp>COLor{:DEFAult|:SCReen|
                         <ui>:<NRx>}
TEK4692?<sp>COLor
```

Range: `<ui>` = 0 to 7, and specifies the color.
`<NRx>` = 0 to 4095, and specifies the printer color.

Arguments

- **DEFAult**—assigns default copier colors to the DSA color index as shown below.
- **SCReen**—assigns copier colors to match the current colors on the display.

  The color assignments for the original color system differ from those for the standard color system.

Default TEK4692 Color Assignments-Original Color System

Color Index	4692 Color	Color Index	4692 Color
0	4095 (0xFFF)	4	1020 (0x3FC)
1	243 (0x0F3)	5	0 (0x000)
2	1638 (0x666)	6	207 (0x0CF)
3	972 (0x3CC)	7	3840 (0xF00)

Default TEK4692 Color Assignments-Standard Color System

Color Index	4692 Color	Color Index	4692 Color
0	4095 (0xFFF)	4	2362 (0x93C)
1	0 (0x000)	5	1020 (0x3FC)
2	3945 (0xF69)	6	2457 (0x999)
3	1776 (0x6F0)	7	3840 (0xF00)

Note: Refer to page 90 for the color index.

- **<ui>:<NRx>** – assigns copier colors to the DSA color index.

Examples of 4692 Index Coding

4692 Color	Maps to
4095 (0xFFF)	White
240 (0x0F0)	Green
4080 (0xFF0)	Yellow
15 (0x00F)	Purple
0 (0x000)	Black
255 (0x0FF)	Blue
3840 (0x0FF)	Red

Note: RGB color charts are included in the *4692 Color Graphics Copier Device Driver Development Guide* (Tektronix part number 070-4818-00).

Examples

```
TEK4692 COL:DEFA
```

assigns default colors to the color index.

DIRection

DIRection selects the printing orientation.

Syntax:
```
TEK4692<sp>DIRection:{HORiz|VERt}
TEK4692?<sp>DIRection
```

Arguments

- **HORiz** – prints rows left to right and from top to bottom.
- **VERt** – prints columns bottom to top and from left to right.

Examples

`TEK4692 DIR:VER`
selects a horizontal printing orientation.

FORMat

FORMat selects the output format. Use SCReen for the 4693D printer in 4692 emulation mode.

Syntax:
```
TEK4692<sp>FORMat:{DIThered|DRAft|HIRes|
                   SCReen}
TEK4692?<sp>FORMat
```

Arguments

- **DIThered** – modifies print contrast for TEK4692.
- **DRAft** – prints monochrome.
- **HIRes** – prints front panel intensified regions.
- **SCReen** – is a one-to-one mapping of 3-bit pixel information.

Examples

`TEK4692 FORM:DIT`
selects a print format that alters the print contrast for the Tektronix 4692 printer.

PORt

PORt specifies the output port for the printer.

Syntax:
```
TEK4692<sp>PORt:{CENTRonics|GPIb|RS232
                           |DISk}
TEK4692?<sp>PORt
```

Examples

`TEK4692 POR:CENTR`
specifies the Centronix port for hardcopy output.

SECUre

SECUre specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax:
```
TEK4692<sp>SECUre:{ON|OFF}
TEK4692?<sp>SECUre
```

Examples

`TEK4692 SECU:OFF`
Turns off the SECUre function.

TEK4696

COLor
DIRection
FORMat
PORt
SECUre

TEK4696 specifies parameters for the Tektronix 4696 and Tektronix 4695 color inkjet printers.

Syntax:
```
TEK4696<sp><link>:<arg>
TEK4696?[<sp><link>]
```

COLor

COLor assigns inkjet colors to the DSA color index.

Syntax:
```
TEK4696<sp>COLor{:DEFAult|<ui>:<NRx>}
TEK4696?<sp>COLor
```

Range: `<ui>` = 0 to 7, and specifies the color.
`<NRx>` = 0 to 12, and specifies the printer color.

Arguments

- **DEFAult**—assigns default inkjet colors to the DSA color index as shown below.

  The color assignments for the original color system differ from those for the standard color system.

Default Inkjet Color Assignments—Original Color System

Color Index	4696 Color	Color Index	4696 Color
0	White	4	Blue
1	Green	5	Black
2	Cyan	6	Magenta
3	Cyan	7	Red

Default Inkjet Colors Assignments—Standard Color System

Color Index	4696 Color	Color Index	4696 Color
0	White	4	Blue
1	Black	5	Cyan
2	Magenta	6	Black
3	Green	7	Red

Note: Refer to page 90 for definitions of the color index.

- **<ui>:<NRx>**—assigns inkjet colors to the DSA color index.

The colors associated with each 4696 Printer color number are listed below:

Colors Associated With 4696 Color Numbers

4696 No.	Actual Color	4696 No.	Actual Color
0	white	7	purple
1	cyan	8	black
2	yellow	9	black & cyan
3	green	10	black & yellow
4	magenta	11	black, cyan, yellow
5	blue	12	black & magenta
6	red		

Examples

```
TEK4696 COL3:3
```
assigns a color to the color index.

DIRection

DIRection selects the printing orientation.

Syntax:
```
TEK4696<sp>DIRection:{HORiz|VERt}
TEK4696?<sp>DIRection
```

Arguments

- **HORiz**—prints rows left to right and from top to bottom.
- **VERt**—prints columns bottom to top and from left to right.

Examples

```
TEK4696 DIR:HOR
```
selects a horizontal printing orientation.

FORMat

FORMat selects the output format.

Syntax:
```
TEK4696<sp>FORMat:{DIThered|DRAft|HIRes|
                   REDuced|SCReen}
TEK4696?<sp>FORMat
```

Arguments

- **DIThered** – improves print contrast for TEK4696.
- **DRAft** – prints monochrome.
- **HIRes** – prints front panel intensified regions.
- **REDuced** – is a quarter-size version of DRAft.
- **SCReen** – is a one-to-one mapping of 3-bit pixel information.

Examples

`TEK4696 FORM:SCR`
uses a one-to-one mapping of 3-bit pixel data for printer output.

PORt

PORt specifies the output port for the printer.

Syntax:
```
TEK4696<sp>PORt:{CENTRonics|GPIb|RS232
                 |DISk}
TEK4696?<sp>PORt
```

Examples

`TEK4696 POR:RS232`
selects the RS-232-C port for hardcopy output.

SECUre

SECUre specifies that only the waveform(s) and the graticule are sent to the plotter.

Syntax:
```
TEK4696<sp>SECUre:{ON|OFF}
TEK4696?<sp>SECUre
```

Examples

```
TEK4696 SECU:OFF
```
Turns off the SECUre function.

TEK4697

TEK4697 specifies parameters for the Tektronix 4697 color graphics copier.

The syntax for TEK4697 is identical to that for TEK4696, described previously. Color assignments match those for TEK4692.

TEKSECURE

TEKSECURE resets the instrument and erases all of the internal memory. All settings and waveforms stored in memory will be lost.Calibration constants generated by Enhanced Accuracy will also be lost. Factory settings will be retained. All other memory is erased. The floppy disk is not affected by this command (use the FORMat command to erase a floppy disk).

Syntax:
```
TEKSECURE
TEKSECURE?
```

Query note: TEKSECURE? responds with ERASED only after the command has been executed (otherwise NOTERASED is returned).

Examples

`TEKSECURE`
    Erases all memory.

TESt

Set Only. TESt initiates the Self-test diagnostics or, with the XTNd argument, the Extended Diagnostics.

Syntax: `TESt [<sp>XTNd]`

Completion of diagnostics is signaled with either event code 460, successful completion of tests, or event code 394, completion with failed tests.

Note: TESt destroys user-defined expansion strings created with the DEF command, resets the TEXt X,Y coordinates to 0,0, and removes user-entered text from the display. If option 4C (NVRAM) is not installed, TESt destroys all stored waveforms.

Examples

`TES XTN`
    runs extended diagnostics.

TEXt

STRing
X
Y

Set Only. TEXt writes character(s) to the selected area of the screen.

Note: TEXt<ui> is an upgraded version of the TEXt command. Strings written with the TEXt command will disappear when the number of graticules is changed, and when TEXt<ui> is used.

Syntax: `TEXt<sp>{CLEar|<link>:<arg>}`

Arguments

- **CLEar**—removes all user-defined text from the display.

Examples

`TEX CLE`
removes all user-defined text from the display.

STRing

Set Only. STRing specifies the text that is to be displayed at the X and Y coordinates. This automatically changes the X position so subsequent strings appear to the right of previous strings.

Syntax: `TEXt<sp>STRing:<qstring>`

Examples

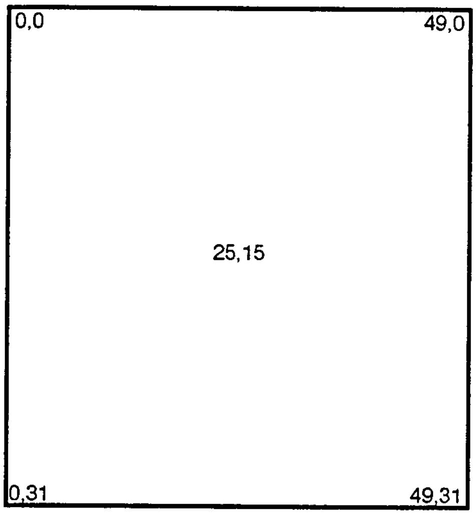
`TEX STR:'Select a waveform'`
defines a string to write to the display.

X **Set Only.** X specifies the horizontal position (X coordinate) of a character in discrete character cells.

Syntax: `TEXt<sp>X:<ui>`

Range: `<ui>` = 0 to 49

The following figure shows the XY cell coordinates.



TEXT X,Y Display Coordinates

Examples

`TEX X:10`
specifies the x coordinate of the screen position.

Y **Set Only.** Y specifies the vertical position (Y coordinate) of a character in discrete character cells.

Syntax: `TEXt<sp>Y:<ui>`

Range: `<ui>` = 0 to 31 (See drawing above)

Examples

```
TEX Y:20
```
specifies the y coordinate of the screen position.

TEXt<ui>

COLor
STRing
X
Y

TEXt<*ui*> specifies up to 12 string buffers that can be displayed anywhere within the graticule display area (except on line −2). TEXt<*ui*> is more versatile than TEXt. The string buffers are permanent until removed (remove them by entering a null string), and their locations may be adjusted.

Syntax:
```
TEXt<ui><sp><link>:<arg>
TEXt<ui>?[<sp><link>]
```

Range: `<ui>` = 0 to 11

COLor

Set Only. COLOR specifies the color of the text that is to be displayed at the X: and Y: coordinates. <*ui*> can range from 1 to 7. The actual color depends on the colors set by the color command.

Syntax: `TEXt<ui><sp>COLor:<ui>`

Range: `<ui>` = 0 to 7

Examples

```
TEX1 COL:3
```

Note: You can change the color within a string by embedding the appropriate escape sequence in the string. See Appendix C.

STRing

STRing specifies the text that is to be displayed at the X: and Y: coordinates.

Syntax: `TEXt<ui><sp>STRing:<qstring>`
`TEXt<ui>?<sp>STRing`

Examples

`TEX1 STR:'Select a waveform'`
defines a string to write to the display.

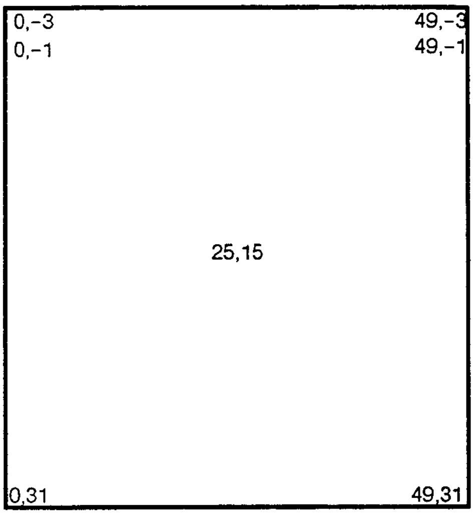
X

X specifies the horizontal position (X coordinate) of a character in discrete character cells. The range is 0 (left edge of the graticule) to 49 (right edge of the graticule). If the position of the first character causes subsequent characters to be pushed off the right side of the screen, these characters will not be displayed (the string will not wrap around to the first position on the next line).

Syntax: `TEXt<ui><sp>X:<ui>`
`TEXt<ui>?<sp>X`

Range: <ui> = 0 to 49

The following figure shows some TEXt<*ui*> X,Y cell coordinates.



TEXT<ui> X:,Y: Display Coordinates

Examples

`TEXO X:10`
specifies the x coordinate of the screen position.

Y Y specifies the vertical position (Y coordinate) of a character in discrete character cells. The range is -3 (top edge of the screen), and -1 to 31 (bottom edge of the graticule). You are not permitted to write on the icon line (y=-2).

Syntax: `TEXt<ui><sp>Y:<ui>`
`TEXt?<sp>Y`

Range: `<ui>` = -3, -1 to 31 (See drawing above)

Examples

```
TEXO Y:20
```

specifies the y coordinate of the screen position.

THD

Query Only. THD returns the total harmonic distortion, that is, the RMS value of the harmonics divided by the fundamental in decibels or percent depending on the FFT magnitude format (logarithmic or linear), followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `THD?`

Returns: `<NR3> or <bblock>`

Examples

```
THD?
```

returns the total harmonic distortion measurement, such as:

```
THD -6.084E+0,EQ
```

TIMe

TIMe sets the time of day on the internal clock.

Syntax:
```
TIMe<sp><qstring>
TIMe?
```

Range: `<qstring>` = <hh:mm:ss>

where hh is hour,
mm is minutes, and
ss is seconds in 24-hour format.

Examples

```
TIM '07:25:30'
```
sets the time of day.

TOPline

The TOPLINE command sets the vertical topline level for measurements.

Syntax:
```
TOPline<sp><NRx>
TOPline?
```

Range: `<NRx>` = -5.0E+20 to 5.0E+20

TOPLINE sets the topline level when MTRACK (measurement tracking) is set to OFF or BASELINE.

Note: The TOPline command is ignored when MTRack is set to BOTh or TOPline.

Examples

```
TOP 2.0
```
sets the top vertical level for measurements.

TR

Query Only. The TR query is equivalent to entering: TRMain?;TRWin?.

Syntax: `TR?`

The response is:

```
TRMAIN MODE:<arg>,ALEVEL:<NR3>,
   ANLEVEL:<NR3>,{DIVS|VOLTS},
   COUPLING:<arg>,SLOPE:<arg>,
   SOURCE:<qstring>,STATUS:<arg>,
   TIHOLDOFF:<NR3>;
   TRWIN MODE:<arg>,ALEVEL:<NR3>,
   COUPLING:<arg>,EVHOLDOFF:<NR1>,
   NLEVEL:<NR3>,{DIVS|VOLTS},
   SLOPE:<arg>,SOURCE:<qstring>,
   STATUS:<arg>,TIHOLDOFF:<NR3>
```

Note: The TR header is not part of the response.

Examples

`TR?`
returns the main and window time base parameters.

TRAce

ACCumulate
ACState
DEScription
GRLocation
GRType
WFMCalc
XUNit
YUNit

TRAce defines a waveform and its characteristics.

Syntax:

```
TRAce<ui><sp><link>:<arg>
TRAce?
TRAce<ui>?[<sp><link>]
```

Range: `<ui>` = 1 to 8, and specifies the waveform.

Query Responses

- **TRAce?** – returns waveform characteristics for all defined waveforms in low-to-high order.
- **TRAce<ui>?** – returns the links and arguments of the specified waveform in the following order:

```
TRACE<ui> DESCRIPTION:<qstring>,
   ACCUMULATE:<arg>,ACSTATE:<arg>,
   GRLOCATION:<arg>,GRTYPE:<arg>,
   WFMCALC:<arg>,XUNIT:<arg>,YUNIT:<arg>
```

ACCumulate

ACCumulate controls the display persistence of the specified trace.

Syntax:

```
TRAce<ui><sp>ACCumulate:{INFPersist|OFF|
                         VARPersist}
TRAce<ui>?<sp>ACCumulate
```

Arguments

- **INFPersist** – selects infinite persistence. In this mode, waveform record points remain on the display indefinitely until some event clears the trace display.
- **OFF** – returns the trace to normal display mode. In normal display mode, waveform record points are cleared from the display each time a new waveform record is displayed.

- **VARPERSIST**—selects variable persistence mode. In this mode, waveform record points remain on the display for the length of time specified by DISPLAY PERSISTENCE before being cleared from the display.

You cannot set ACCumulate to VARPersist or INFPersist in the following cases:

- For a stored or scalar waveform (e.g., STO9)
- When the record length is greater than 2048

You cannot mix INFPersist and VARPersist waveforms on the same graticule. Changing one waveform from one persist mode to the other automatically changes all persist mode waveforms on the same graticule (waveforms in normal display mode are not affected).

Note: You can take automated measurements of traces in the normal display mode only.

Examples

```
TRA3 ACC:VARP
```
sets the variable persistence mode.

ACState

Query Only. ACState returns the accuracy mode in which the specified waveform was created.

Syntax: `TRAce<ui>?<sp>ACState`

Query Responses

- **ENHanced**—the DSA was in enhanced accuracy state when the waveform was acquired.
- **NENHanced**—the DSA was not in enhanced accuracy state when the waveform was acquired.

Examples

```
TRA3? ACS
```
returns the accuracy mode, such as:
```
TRACE3 ACSTATE:ENHANCED
```

DEScription

DEScription defines the source expression(s) of the selected waveform.

Syntax:
```
TRAce<ui><sp>DEScription:<qstring>
TRAce<ui>?<sp>DEScription
```

Range: `<qstring>` is ≤120 characters† in the format:

```
<y exp> [VS <x exp>] [ON <time base>]
```

where:

`<y exp>, <x exp>`	::=	Expressions (see below)
`[VS <x exp>]`	::=	Indicates an XY waveform; if omitted, the waveform is YT.
`[ON <time base>]`	::=	Indicates time base—{MAIN | WIN1 | WIN2}; if omitted, defaults to MAIN

Terms and Operators Available to Form Expressions

<slot> <ui>	Channel designator, e.g. L1
STO<ui> or <filenames>	Stored waveform, the range is 1 to 420 (1 to 453 without the disk drive, or 1 to 918 with Option 4C installed)
<NRx>	Scalar number
<function>	Any of the following functions: ABS | AVG | CONVOLUTION | CORRELATION | DEJITTER | DELAY | DIFF | ENV | EXP | FFTIMAG | FFTMAG | FFTPHASE | FFTREAL | FILTER | IFFT | INTG | INTP | LN | LOG | PIADD† | PISUB† | PULSE | SIGNUM | SMOOTH | SQRT
<operators>	Any of the following operators: + | - | * | /

† <y exp> and <x exp> can have a maximum of 108 characters.

Note: You cannot use a waveform description that consists of only stored or scalar elements as the argument of an AVG or ENV function. You also cannot create a waveform with only stored or scalar elements on the WIN1 or WIN2 time base.

† *The PIADD and PISUB functions are not available from the front panel.*

XY Waveform Considerations. The DSA permits only one acquired XY waveform or two unacquired XY waveforms to be displayed via TRAce<ui> DEScription. (An acquired XY trace description must be high precision acquired waveforms; an unacquired XY trace description has only stored or scalar components.)

Components of XY Descriptions

Acquired XY Description	Unacquired XY Description
"L1 VS L2"	"STO50 VS STO12"
"L3 VS L1 + L4"	"STO90 VS 200"

In addition, the horizontal and vertical components (<x exp> and <y exp>) must have the same scaling mode; both must be integer mode or both floating-point mode waveforms.

PIADD and PISUB Functions. These functions, which are not available from the front panel, allow you to add or subtract the signals from any two channels in a plug-in unit and treat them as a single channel. This operation is an analog addition or subtraction performed in the plug-in unit. The syntax of these functions (using channels L1 and L2 as an example) is:

```
TRAce1 DEScription:'PIADD(L1,L2)'

TRAce2 DEScription:'PISUB(L1,L2)'
```

Because system calibration constants do not apply in this mode, there may be a DC offset. To check if there is a DC offset, turn off the two channels and acquire the baseline value. This value will be the DC offset.

Examples

`TRA2 DES:'ENV(L2)'; TRA3 DES:'STO9+C1'`
defines two new waveforms.

GRLocation

GRLocation moves the selected waveform to the upper or lower graticule pair.

Syntax: `TRAce<ui><sp>GRLocation:{LOWer|UPPer}`
`TRAce<ui>?<sp>GRLocation`

Examples

`TRA2 GRL:LOW`
moves waveform 2 to the lower graticule.

GRType

GRType sets the graticule type of the selected waveform to linear. (Linear is the only option currently available.)

Syntax: `TRAce<ui><sp>GRType:{LINear}`
`TRAce<ui>?<sp>GRType`

Examples

`TRA2 GRT:LIN`
selects a linear graticule type for waveform 2.

WFMCalc

Query Only. WFMCalc returns whether a waveform was created in integer mode (FASt) or floating-point mode (HIPrec). Once a waveform is created in one mode, you cannot change the waveform to the other mode. (Refer to WFMScaling command.)

Syntax: `TRAce<ui>?<sp>WFMCalc`

Query Responses

- **HIPrec**—floating point mode.
- **FASt**—integer mode.

Examples

TRA2? WFMC

returns the mode that the waveform was created in, such as:

TRACE2 WFMCALC:HIPREC

XUNit

Query Only. XUNit returns the horizontal units (X-axis) of the specified waveform.

Syntax: TRAce<ui>?<sp>XUNit

Query Responses

- AMPS, DIVS, DEGrees, HERtz, OHMs, SEConds, VOLts, or WATts.

Examples

TRA5? XUN

returns the horizontal units of trace 5, such as:

TRACE5 XUNIT:SECONDS

YUNit

Query Only. YUNit returns the vertical units (Y-axis) of the specified waveform.

Syntax: TRAce<ui>?<sp>YUNit

Query Responses

- AMPS, DIVS, DEGrees, HERtz, OHMs, SEConds, VOLts, or WATts.

Examples

TRA5? YUN

returns the vertical units of trace 5, such as:

TRACE5 YUNIT:VOLTS

TRANUm

Query Only. TRANUm returns the number of waveforms displayed on the front panel.

Syntax: `TRANUm?`

Returns: `<NR1>`

Examples

`TRANU?`

returns the number of waveforms displayed on the front panel, such as:

`TRANUM 4`

TRLevel

TRLEVEL sets the trigger DC level mode.

Syntax:
```
TRLevel<sp>{ABSOlute|SCReen}
TRLevel?
```

Arguments

- **ABSOLUTE** – absolute mode, the trigger level remains constant in input units (usually volts) when changes are made to vertical size or position. In this mode, the trigger level is constrained to remain on the screen. This is the default TRLEVEL mode.
- **SCREEN** – screen mode, the trigger level remains constant on screen when changes are made to the vertical sensitivity or offset of the input channel(s) (changes to the vertical size or position of a trace).

Examples

`TRL ABSO`

TRMain

ALEvel
ANBlevel
ANLevel
COUpling
MODe
SLOpe
SOUrce
STAtus
TIHoldoff
TIMER1
TIMER2

TRMain sets the parameters of the Main trigger.

Syntax:
```
TRMain<sp><link>:<arg>
TRMain?[<sp><link>]
```

Query Responses

- **TRMain?**—returns all links and their arguments, in the following order:

```
TRMAIN MODE:<arg>,ALEVEL:<NR3>,COUPLING:<arg>,
    SLOPE:<arg>,SOURCE:<qstring>,ANLEVEL:<NR3>,
    {DIVS|VOLTS},ANBLEVEL:<NR3>,{DIVS|VOLTS},
    STATUS:<arg>,TIHOLDOFF:<NR3>,TIMER1:<NR3>,
    TIMER2:<NR3>
```

ALEvel

When TRMain MODe is set to AUTOLevel, ALEvel sets the trigger level to a percentage of the peak-to-peak value of the trigger source signal.

When TRMain MODe is not set to AUTOLevel, the ALEvel value is saved and applied later when MODe is changed to AUTOLevel.

Syntax:
```
TRMain<sp>ALEvel:<NRx>
TRMain?<sp>ALEvel
```

Range: `<NRx>` = 20 to 80 percent.

Examples

```
TRM ALE:25
```
sets the main trigger level.

ANBlevel

When TRMain MODe is AUTO or NORmal and extended triggering mode is active (i.e., TRMain SOUrce: <*exp*> includes WHILE, AND, OR, TO, or XOR), ANBlevel sets the level of the B trigger source to the specified value.

Syntax:

```
TRMain<sp>ANBlevel:<NRx>,{DIVS|VOLts}
TRMain?<sp>ANBlevel
```

Range: The `ANBlevel:<NRx>,` DIVS range is -5 to +5 graticule divisions. VOLts range is calculated with the same formula as ANLEvel.

Examples

```
TRM ANB:150E-3,VOL
```

ANLEvel

When TRMain MODe is set to AUTO or NORmal, ANLEvel sets the trigger level to the specified value in the specified units (DIVS or VOLTS; see below for scaling information).

When TRMain MODe is set to AUTOLevel, you cannot set ANLEvel; the *set* value for ANLEvel is ignored. However, querying ANLEvel when MODe is set to AUTOLevel returns the current level value scaled in DIVS.

Note: Be sure to set the TRMain MODe, COUpling, and SOURCE links before setting ANLEvel.

Syntax:

```
TRMain<sp>ALEvel:<NRx>,{DIVS|VOLts}
TRMain?<sp>ALEvel
```

Range: The `ANLEvel:<NRx>,DIVS` range is –5 to +5 graticule divisions.

The range for `ANLEvel:<NRx>,VOLTS` is calculated with the following formulas using the sensitivity and offset of the trigger source channel (CH<slot><ui>? SEN,OFFS):

(-5 * SEN + OFFS) to (+5 * SEN + OFFS)

Trigger Level Scaling. If TRMain SOUrce is a single channel (e.g., L1) and TRMain COUpling is DC, DCHf, or DCNoise, the DSA scales the ANLEvel value in VOLts. For any other combination of TRMain SOUrce and COUpling, the DSA scales the ANLEvel value in DIVS.

When the DSA scales ANLEvel in VOLts, you can set ANLEvel in either VOLts or DIVS. DIVS are converted to VOLts using the formula:

<#_of_DIVS> * SEN + OFFS

where SEN and OFFS are the sensitivity and offset links of the trigger source channel.

When the DSA scales ANLEvel in DIVS, you can only set ANLEvel in DIVS. Attempting to set ANLEvel in VOLTS is an error.

Note: This formula also applies to the volts range for ANBlevel.

Trigger Level Usage Examples. The following are examples of trigger level usage. The first three columns contain the MODe, COUpling, and SOUrce arguments. The fourth column gives an ANLEvel value in either DIVS or VOLts, and the last column shows the effect.

Trigger Level Usage Examples

MODE:	COU:	SOU:	Level Setting	Result
AUTOL	DC	L1	ANL:3,DIVS	ignored
AUTOL	DC	L1	ANL:3,VOLts	ignored
AUTO	DC	L1+L2	ANL:3,DIVS	value OK
NOR	DC	L1	ANL:3,DIVS	converted
AUTO	DC	L1	ANL:3,VOLts	value OK
NOR	AC	L1	ANL:3,DIVS	value OK
AUTO	AC	L1	ANL:3,VOLts	error
NOR	AC	L1	ANL:3,VOLts	error

In the Result column of the previous table, "ignored" means the set value is not used; "value OK" means both the value and units are acceptable; "converted" means that the DIVS units were converted to VOLts; and "error" means that VOLts was an unacceptable unit.

Examples

`TRM ANL:150E-3,VOL`
sets the trigger level.

COUpling

COUpling selects the Main trigger coupling. LF and HF mean low frequency and high frequency, respectively.

Syntax:

```
TRMain<sp>COUpling:{AC|ACHf|ACLf|ACNoise
                    DC|DCHf|DCNoise}
TRMain?<sp>COUpling
```

Note: Be sure to set TRMain MODe, COUpling, and SOUrce before setting ANLEvel.

Examples

`TRM COU:DCH`
sets the main trigger coupling.

MODe

MODe selects Main triggering mode.

Syntax:

```
TRMain<sp>MODe:{AUTO|AUTOLevel|NORmal}
TRMain?<sp>MODe
```

Arguments

- **AUTO**—the trigger level is set with ANLEvel.
- **AUTOLevel**—the trigger level is set with ALEvel.
- **NORmal**—the trigger level is set with ANLEvel.

Note: Be sure to set TRMain MODe, COUpling, and SOUrce before setting ANLEvel.

Examples

`TRM MOD:AUTOL`
specifies that the trigger level is set by ALEvel.

SLOpe

SLOpe sets the Main trigger slope.

Syntax:
```
TRMain<sp>SLOpe:{MINUs|PLUs}
TRMain?<sp>SLOpe
```

Examples

`TRM SLO:MINU`
selects a minus slope for the main trigger.

SOUrce

SOUrce sets the trigger source to the specified expression <exp>.

Syntax: `TRMain SOUrce:<qstring>`

Range: `<qstring>` = <exp>

The following is the main trigger source expression syntax:

```
[NOT] {<lc>|LINE} [<binop> {<r>|LINE} [<time>]]
```

or:

```
[NOT] {<r>|LINE} [<binop> {<lc>|LINE} [<time>]]
```

Where:

`<lc>` ::= [+ |-] {L|C}<ui> [[+ |-] {L|C}<ui> ...]

`<r>` ::= [+ |-] R<ui> [[+ |-] R<ui> ...]

`<binop>` ::= AND | OR | TO | WHILE | XOR

`<time>` ::= >T1 | <T1 | >T1<T2 | <T1>T2

The following rules apply:

- LINE can appear only once in a trigger source expression.

- If <binop> is TO, then <time> must be >t1 or <t1.
- If <binop> is WHILE, then <time> is not allowed.
- You can combine L and C channels (add/subtract) with each other, but not with R channels.
- You can combine R channels with other R channels, but not with L or C channels.
- You can invert any channel, except the single input channel of an 11A71 Amplifier.
- You cannot reference a channel that is not installed.
- Triggers cannot be chopped between Main and Window time bases.

Chopped Triggers. Each plug-in unit has a single trigger output line. Trigger expressions define the use of this trigger line by specifying the number and polarity of each channel used from the plug-in unit. Once the trigger line is assigned, no other trigger access is available from that plug-in unit. Thus, two waveforms cannot use the trigger line from one plug-in unit in different ways.

In particular, when Window trigger mode (WTMode) is set to time holdoff or event holdoff (TIHoldoff or EVHoldoff), and both the Main and Window trigger source expressions reference the same plug-in unit, both expressions must reference the same channel(s) and no other channels from that plug-in compartment; otherwise, the triggers are chopped, which is not acceptable.

The following table contains examples of acceptable and unacceptable (chopped) trigger sources. (Assume WTMode is set to TIHoldoff and each plug-in compartment has a two-channel amplifier installed.)

Chopped Trigger Source Examples

TRMAIN SOURCE	Acceptable TRWIN SOU:	Chopped TRWIN SOU:
"L1"	"L1"	"L2"
"L1"	"L1+C1"	"L1+L1"
"L1"	"R2"	"L2+C2"
"L1"	"C1+C2"	"R1+C1"
"C1+C2"	"C1+C2"	"C1"
"C1+C2"	"C1+C2+L2"	"C1+C1"
"C1+C2"	"L1+L2"	"L2+C2"
"C1+C2"	"R1"	"R1+C1"

Note: When WTMode is set to MAIn, the Window trigger source has no effect on the Main trigger source and no checks are made for chopped triggers.

Examples

`TRM SOU:'L1-C1'`
specifies the trigger source as left plug-in channel 1 minus center plug-in channel 1.

`TRM SOU:'NOT L1-C1 or LINE >T1'`
specifies the trigger source as left plug-in channel 1 minus center plug-in channel 1 when below A trigger level and before timer1, or line above B trigger level before timer 1.

STAtus

Query Only. STAtus returns the trigger status of the Main time base.

Syntax: `TRMain?<sp>STAtus`

Query Responses

- **NOTrg**—the Main time base is not triggered.
- **TRG**—the Main time base is triggered.

Examples

```
TRM? STA
```

returns the trigger status of the main time base, such as:

```
TRMAIN STATUS:TRG
```

TIHoldoff

TIHoldoff sets the Main trigger time holdoff in seconds.

Syntax:

```
TRMain<sp>TIHoldoff:<NRx>
TRMain?<sp>TIHoldoff
```

Range: `<NRx>` = 2E–6 to 500 sec.

Examples

```
TRM TIH:24E-3
```

specifies the main trigger holdoff.

TIMER1

TIMER1 sets the first Main trigger timer in seconds.

Syntax:

```
TRMain<sp>TIMER1:<NRx>
TRMain?<sp>TIMER1
```

Range: `<NRx>` = 2E–9 to 1.048E–3 sec.

Examples

```
TRM TIMER1:5E-6
```

TIMER2

TIMER2 sets the second Main trigger timer in seconds.

Syntax:

```
TRMain<sp>TIMER2:<NRx>
TRMain?<sp>TIMER2
```

Range: `<NRx>` = 4E–9 to 2.096E–3 sec.
The TIMER2 range is:
(TIMER1 + 2E–9) to (TIMER1 + 1.048E–3)

Examples

```
TRM TIMER2:5E-6
```

TRWin

ALEvel
COUpling
EVHoldoff
MODe
NLEvel
SLOpe
SOUrce
STAtus
TIHoldoff
TIMER1
TIMER2

TRWin sets Window trigger parameters.

Syntax:
```
TRWin<sp><link>:<arg>
TRWin?[<sp><link>]
```

Query Responses

- **TRWin?**—returns all links and their arguments, in the following order:

```
TRWIN MODE:<arg>,ALEVEL:<NR3>,COUPLING:<arg>,
    EVHOLDOFF:<NR1>,NLEVEL:<NR3>,{DIVS|VOLTS},
    SLOPE:<arg>,SOURCE:<qstring>,STATUS:<arg>,
    TIHOLDOFF:<NR3>,TIMER1:<NRx>,TIMER2:<NR3>
```

ALEvel

When TRWin MODe is set to AUTOLevel, ALEvel sets the trigger level to a percentage of the peak-to-peak value of the trigger source signal.

When TRWin MODe is set to NORmal, the ALEveL value is saved and applied when MODe is changed to AUTOLevel.

Syntax:
```
TRWin<sp>ALEvel:<NRx>
TRWin?<sp>ALEvel
```

Range: <NRx> = 20 to 80 percent.

Examples

```
TRW ALE:25
```
sets the trigger level.

COUpling

COUpling selects Window trigger coupling. LF and HF mean low frequency and high frequency, respectively.

Syntax:
```
TRWin<sp>COUpling:{AC|ACHf|ACLf|ACNoise|
                   DC|DCHf|DCNoise}
TRWin?<sp>COUpling
```

Examples

`TRW COU:DCH`
sets the window trigger coupling.

EVHoldoff

EVHoldoff sets the Window trigger event holdoff to the specified number of events.

Syntax:
```
TRWin<sp>EVHoldoff:<NRx>
TRWin?<sp>EVHoldoff
```

Range: `<NRx>` = 1 to 1E9 events.

Examples

`TRW EVH:500`
specifies the window trigger event holdoff to 500 events.

MODe

MODe selects the Window triggering mode.

Syntax:
```
TRWin<sp>MODe:{AUTOLevel|NORmal}
TRWin?<sp>MODe
```

Arguments

- **AUTOLevel**—the trigger level is set with ALEvel.
- **NORmal**—the trigger level is set with NLEvel.

Examples

`TRW MOD:AUTOL`
sets the trigger level with ALEvel.

NLEvel

When TRWin MODe is set to NORmal, NLEvel sets the trigger level to the specified value in the specified units (DIVS or VOLts; see below for scaling information).

When TRWin MODe is set to AUTOLevel, the NLEvel set value is ignored; however, querying NLEvel returns the current level scaled in DIVS.

Note: Be sure to set TRWin MODe, COUpling, and SOUrce before setting NLEvel.

Syntax: `TRWin<sp>NLEvel:<NRx>,{DIVS|VOLts}`
`TRWin?<sp>NLEvel`

Range: The `NLEvel:<NRx>,DIVS` range is -5 to +5 graticule divisions.

The `NLEvel:<NRx>,VOLts` range is calculated with the following formulas using the sensitivity and offset of the trigger source channel (CH<slot><ui>? SEN,OFFS):

(-5 * SEN + OFFS) to (+5 * SEN + OFFS)

Trigger Level Scaling. If the TRWin SOUrce is a single channel and TRWin COUpling is DC, DCHf, or DCNoise, the DSA scales the NLEvel value in VOLTS. For any other combination of TRWin SOUrce, and COUpling, the DSA scales the NLEvel value in DIVS.

When the DSA scales NLEvel in VOLts, you can set NLEvel in either VOLts or DIVS. DIVS are converted to VOLts using this formula:

<#_of_DIVS> * SEN + OFFS

where SEN and OFFS are the sensitivity and offset links of the trigger source channel.

When the DSA scales NLEvel in DIVS, you can only set NLEvel in DIVS. Attempting to set NLEvel in VOLts will result in an error.

Usage. Window trigger NLEvel usage is the same as for Main trigger ANLEvel. Refer to page 301 for examples, substituting NLEvel:3, {DIVS|VOLts} in the Level Setting column.

Examples

`TRW NLE:-2.625E-3,VOL`
sets the trigger level.

SLOpe

SLOpe sets the Window trigger slope.

Syntax: `TRWin<sp>SLOpe:{MINUs|PLUs}`
`TRWin?<sp>SLOpe`

Examples

`TRW SLO:MINU`
selects a minus slope for the window trigger slope.

SOUrce

SOUrce sets the Window trigger source to the specified trigger expression, <exp>.

Syntax: `TRWin<sp>SOUrce:<qstring>`
`TRWin?<sp>SOUrce`

Range: `<qstring>` = <exp>

The following is the window trigger source expression syntax:

```
[NOT] {<lc>|<r>|LINE} [<time>]
```

Where:

<lc> ::= [+|-] {L|C}<ui> [[+|-] {L|C}<ui> ...]

<r> ::= [+|-] R<ui> [[+|-] R<ui> ...]

<time> ::= >T1 | <T1 | >T1<T2 | <T1>T2

TRWin SOUrce is a subset of TRMain SOUrce. Note that <binop> expressions are not allowed. Refer to TRMain SOUrce (page 303) for source restrictions and examples.

Examples

`TRW SOU:'L1-C1'`
specifies the window trigger source.

STAtus

Query Only. STAtus returns the trigger status of the Window time base.

Syntax: `TRWin?<sp>STAtus`

Query Responses

- **NOTrg** – the Window time base is not triggered.
- **TRG** – the Window time base is triggered.

Examples

```
TRW? STA
```
returns the trigger status of the window time base,such as:
```
TRWIN STATUS:NOTRG
```

TIHoldoff

TIHoldoff sets the Window trigger time holdoff in seconds. Maximum TRWin TIHoldoff ≤TRMain TIHoldoff.

Syntax:
```
TRWin<sp>TIHoldoff:<NRx>
TRWin?<sp>TIHoldoff
```

Range: `<NRx>` = 35E–9 to (TRMain TIHoldoff)

Examples

```
TRW TIH:24E-3
```
specifies the window trigger holdoff time.

TIMER1

TIMER1 sets the first Window trigger timer in seconds.

Syntax:
```
TRMain<sp>TIMER1:<NRx>
TRMain?<sp>TIMER1
```

Range: `<NRx>` = 2E–9 to 1.048E–3 sec.

Examples

```
TRM TIMER1:5E-6
```
specifies the main trigger holdoff.

TIMER2

TIMER2 sets the second Window trigger timer in seconds.

Syntax: `TRMain<sp>TIMER2:<NRx>`
`TRMain?<sp>TIMER2`

Range: `<NRx>` = 4E−9 to 2.096E−3 sec
The TIMER2 range is:
(TIMER1 + 2E−9) to (TIMER1 + 1.048E−3)

Examples

`TRM TIMER2:5E-6`
specifies the main trigger holdoff.

TSMain

Query Only. TSMain returns the elapsed time between the actual trigger point and the waveform sample identified as 0 seconds, for real-time single-shot acquisitions only.

Syntax: `TSMain?`

Examples

`TSM?`
returns the time between the actual trigger point and the 0 second point of the waveform sample, such as:

```
TSMAIN 2.228E-9
```

TTAverage

TTAverage sets the number of averages for the TTRig measurement. It applies to all waveforms.

Syntax:
```
TTAverage<sp><NRx>
TTAverage?
```

Range: `<NRx>` = 1, 10, 100, or 1000

Examples

```
TTA 100
```
specifies 100 averages for the TTRig measurement.

TTRig

Query Only. TTRig returns the time between the Main trigger point and the Window trigger point, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `TTRig?`

Returns: `<NR3> or <bblock>`

Examples

```
TTR?
```
returns the time between the main and window trigger points, such as:

```
TTRIG 9.7659E-7,EQ
```

UID

CENter
LEFt
MAIn
RIGht

UID queries or sets the serial numbers of the DSA and its plug-in units. Setting a serial number requires that an internal jumper be installed. Installing this jumper should only be done by a qualified service person. UID can be queried regardless of the jumper position.

Syntax:

```
UID<sp><link>:<arg>
UID?[<sp><link>]
```

Query Responses

- **UID?** – returns its links in the following order:

```
UID MAIN:<qstring>,LEFT:<qstring>,
   CENTER:<qstring>,RIGHT:<qstring>
```

CENter

CENter queries or sets the serial number of the center plug-in unit.

Syntax:

```
UID<sp>CENter:<qstring>
UID?<sp>CENter>
```

Range: `<qstring>` is $\leq$10 characters.

Examples

```
UID? CEN
```

returns the serial number of the center plug-in unit,such as:

```
UID  CENTER:"B010521"
```

LEFt

LEFt queries or sets the serial number of the left plug-in unit.

Syntax:
```
UID<sp>LEFt:<qstring>
UID?<sp>LEFt
```

Range: `<qstring>` is ≤10 characters.

Examples

```
UID? LEF
```
returns the serial number of the left plug-in unit, such as:
```
UID  LEFT:"B010562"
```

MAIn

MAIn queries or sets the serial number of the DSA.

Syntax:
```
UID<sp>MAIn:<qstring>
UID?<sp>MAIn
```

Range: <qstring> is ≤10 characters.

Examples

```
UID? MAI
```
returns the serial number of the DSA, such as:
```
UID  MAIN:"B010400"
```

RIGht

RIGht queries or sets the serial number of the right plug-in unit.

Syntax:
```
UID<sp>RIGht:<qstring>
UID?<sp>RIGht
```

Range: `<qstring>` is ≤10 characters.

Examples

```
UID? RIG
```
returns the serial number of the right plug-in unit, such as:
```
UID  RIGHT:"B010400"
```

UNDEF

Set Only. UNDEF removes from the list of logical names defined by DEF either the specified logical name or ALL defined logical names.

Syntax: `UNDEF<sp>{ALL|<qstring>}`

Examples

`UNDEF 'TB?'`
removes the specified logical name defined by DEF.

UNDershoot

Query Only. UNDershoot returns the difference between the BASeline value and the minimum signal amplitude, given as a percentage of the difference between the TOPline and BASeline values, and followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `UNDershoot?`

Returns: `<NR3> or <bblock>`

Examples

`UND?`
returns the difference between the baseline and the minimum amplitude, such as:

```
UNDERSHOOT 2.334E-9,EQ
```

UPTime

Query Only. UPTime returns the total number of hours the DSA has been powered on, in <NR3> form.

Syntax: `UPTime?`

Examples

```
UPT?
```

returns the number of hours the DSA has been on, such as:

```
UPTIME 1.243E+3
```

USERId

USERId stores the specified string in nonvolatile RAM.

Syntax:
```
USERId<sp><qstring>
USERId?
```

Range: `<qstring>` is ≤128 characters.

Examples

```
USERI '602A Test Station; BLDG 47'
```

V1Bar V2Bar

XCOord
XDIv

V1Bar and V2Bar set the absolute position of the vertical bar cursors.

Syntax:
```
V{1|2}Bar<sp><link>:<arg>
V{1|2}Bar?[<sp><link>]
```

XCOord

XCOord positions the first or second vertical bar cursor using the units of the selected waveform.

Syntax:
```
V{1|2}Bar<sp>XCOord:<NRx>
V{1|2}Bar?<sp>XCOord
```

Range: The XCOord range for a Main waveform is from:

MAINPos to (MAINPos + 10.22 * TBMain TIMe)

The XCOord range for a Window1 waveform is from:

WIN1Pos to (WIN1Pos + 10.22 * TBWin TIMe)

The XCOord range for a Window2 waveform is from:

WIN2Pos to (WIN2Pos + 10.22 * TBWin TIMe)

Examples

`V1B XCO:3.8E-4`
positions the first vertical bar cursor using selected waveform units.

XDIv

XDIv positions the first or second vertical bar cursor in graticule divisions. (-5.12 is the left edge of the display.)

Syntax:
```
V{1|2}Bar<sp>XDIv:<NRx>
V{1|2}Bar?<sp>XDIv
```

Range: <NRx> = -5.12 to +5.10

Examples

`V2B XDI:-4.1`
positions the second vertical bar cursor.

WAVfrm

Query Only. WAVfrm returns the waveform preamble and data points for the waveform specified by OUTput. WAVfrm? is equivalent to entering: WFMpre?;CURVe?.

Syntax: `WAVfrm?`

Refer to the WFMpre and CURVe commands for information on what is returned by WAVfrm.

Examples

`WAV?`
returns all waveform data.

WFMpre

ACState
BIT/nr
BN.fmt
BYT/nr
BYT.or
CRVchk
DATE
ENCdg
LABel
NR.pt
PT.fmt
TIMe
TSTime
WFId
XINcr
XMUlt
XUNit
XZEro
YMUlt
YUNit
YZEro

WFMpre transmits a Tek Codes and Formats preamble for each waveform sent to or from the controller. The preamble generated by the DSA provides scaling and other information for the waveform data transferred with the CURVe command. The preamble sent to the DSA by the controller scales the waveform data transfered to the DSA with the CURVe command. The waveform sent to the DSA with CURVe is specified with the INPut command. The waveform returned to the controller with CURVe? is specified with the OUTput command.

Syntax:
```
WFMpre<sp><link>:<arg>
WFMpre?[<sp><link>]
```

Note: Sending WFMpre implicitly deletes any existing waveform data at INPut STO<ui> and replaces it with null (unacquired) data points. If STO<ui> is the sole component of a displayed waveform (e.g., TRA3 DES:"STO22"), that waveform is removed from the display. If STO<ui> is one component of a complex waveform (e.g., TRA4 DES:"STO22+L1"), you cannot send a waveform preamble to that INPut STO<ui> location because you cannot delete a stored waveform that is part of a complex waveform.

XY Note: The DSA does not support stored XY waveforms. Therefore, although XY waveforms can be transferred to the controller, they cannot be sent back to the DSA.

Query Responses

- **WFMpre?**—returns its links in the following order:

```
WFMPRE ACSTATE:<arg>,BIT/NR:16,
    BN.FMT:RI,BYT/NR:2,BYT.OR:<arg>,
    CRVCHK:<arg>,ENCDG:<arg>,
    NR.PT:<NR1>,PT.FMT:<arg>,WFID:<arg>,
    XINCR:<NR3>,XMULT:<NR3>,
    XUNIT:<arg>,XZERO:<NR3>,
    YMULT:<NR3>,YUNIT:<arg>,
    YZERO:<NR3>,LABEL:<qstring>,
    TIME:<qstring>,DATE:<qstring>
    TSTIME:<NR3>
```

ACState

ACState indicates whether the waveform was created with Enhanced Accuracy or normal configuration calibration accuracy.

Syntax:
```
WFMprep<sp>ACState:{ENHanced|NENhanced}
WFMprep?<sp>ACState
```

Examples

```
WFM ACS:ENH
```
specifies which calibration accuracy was used when creating the waveform.

BIT/nr

Query Only. BIT/nr returns the number of bits per binary waveform point (which can be eight or sixteen). BIT/nr can be set by the BIT/nr command (page 70).

Syntax: `WFMpre?<sp>BIT/nr`

Examples

```
WFM? BIT
```
returns the number of bits per binary trace point, such as:
```
WFMPRE BIT/NR:8
```

BN.fmt

Query Only. BN.fmt returns the Tek Codes and Formats binary number format, which is always RI (right-justified, twos-complement integers).

Syntax: `WFMpre?<sp>BN.fmt`

Examples

```
WFM? BN.
```
returns the Tek Codes and Format binary number format which is always:
```
WFMPRE BN.FMT:RI
```

BYT/nr

Query Only. BYT/nr returns the binary data field width (which can be one or two bytes per binary waveform point). BYT/nr can be set by the BYT/nr command (page 70).

Syntax: `WFMpre?<sp>BYT/nr`

Examples

```
WFM? BYT/
```
returns the binary field width, which is:

```
WFMPRE BYT/NR:1
```

BYT.or

Query Only. BYT.or returns the transmission order of binary waveform data returned by CURVe?. The transmission order is set by the BYT.or command (LSB or MSB).

Syntax: `WFMpre?<sp>BYT.or`

Examples

```
WFM? BYT.
```
returns the order that bytes are transmitted, such as:

```
WFMPRE BYT.OR:LSB
```

CRVchk

Query Only. CRVchk returns the type of checksum appended to the waveform data after it is returned by a CURVe? query. The types are defined below.

Syntax: `WFMpre?<sp>CRVchk`

Query Responses

- **CHKsm0**—the Standard Tek Codes and Formats checksum. This is returned when ENCdg WAVfrm is set to BINary and OUTput is set to STO<ui>.
- **NONe**—no checksum is appended. This is returned when ENCdg WAVfrm is set to ASCII.

- **NULl** – a zero checksum value is appended. This is returned when ENCdg WAVfrm is set to BINary and OUTput is set to TRAce<ui>.

Examples

```
WFM? CRV
```
returns the checksum type, such as:
```
WFMPRE CRVCHK:CHKSM0
```

DATE

DATE is the date stamp for the waveform. The date stamp is recorded when a waveform is stored, or you can set it with this link. If WFMpre? DATE is queried when OUTput is TRAce<ui> (i.e., a displayed waveform), the current date is returned.

Syntax: `WFMpre<sp>DATE:<qstring>`
`WFMpre?<sp>DATE`

Range: `<qstring>` = <dd-mon-yy>

where dd is the day of the month,
mon is the first three letters of the month, and
yy is the last two digits of the year.

Examples

```
WFM DATE:'28-SEP-90'
```
sets the date for the trace.

ENCdg

Query Only. ENCdg returns the state of the data encoding set with the ENCdg command (ASCII or binary). This link is equivalent to an ENCdg? WAVfrm query.

Syntax: `WFMpre?<sp>ENCdg`

Examples

```
WFM? ENC
```
returns the type of data encoding, such as:
```
WFMPRE ENCDG:ASCII
```

LABel

LABel is the optional label associated with the waveform. If the waveform has no label, querying WFMpre? LABel returns a null string (LABel:"").

Syntax:
```
WFMpre<sp>LABel:<qstring>
WFMpre?<sp>LABel
```

Range: `<qstring>` is ≤10 characters.

Examples

```
WFM LAB:'SAMPLE3'
```
specifies a label for the trace.

NR.pt

NR.pt specifies the number of points in the transmitted waveform record. It is normally the same as {TBMain|TBWin} LENgth.

Syntax:
```
WFMpre<sp>NR.pt:{512|1024|2048|4096|5120|
                8192|10240|16384|20464|
                32768}
WFMpre?<sp>NR.pt
```

Note: If OUTput specifies a displayed waveform when Pan/Zoom mode is set to ON and HMAg is greater than 1 for that waveform, then the value returned by WFMpre? is defined as follows: if DISPlay mode is DOTs, then NR.pt equals the number of points displayed on the front panel, rather than the value of {TBMain|TBWin} LENgth. If DISPlay mode is VECtors, then NR.pt equals 512 (this is interpolated data for PAN/ZOOM).

For example, under the following conditions the WFMpre? NR.pt query returns 256:

```
TRACE1 DESCRIPTION:"L1 ON MAIN"
   DISPLAY MODE:DOTS
   TBMAIN LENGTH:2048
   ADJTRACE1 PANZOOM:ON,HMAG:8
   OUTPUT TRACE1
```

Examples

```
WFM NR.:1024
```
specifies the points in the transmitted trace record.

PT.fmt

PT.fmt indicates the point format of the waveform data.

Syntax: `WFMpre<sp>PT.fmt:{ENV|XY|Y}`
`WFMpre?<sp>PT.fmt`

Arguments

- **ENV** – applies to YT waveforms transmitted as maximum-minimum point-pairs, with the maximum point transmitted first.
- **XY** – is an XY waveform which returns an X, Y point-pair for each point in the waveform record.
- Y – indicates a YT waveform, which returns one ASCII or binary data point for each point in the waveform record.

Note: You cannot send XY waveforms to the DSA.

Examples

```
WFM PT.:Y
```
indicates a YT trace format.

TIMe

TIMe is the time stamp for the waveform. The time stamp is recorded when a waveform is stored, or you can set it with this link. If WFMpre? TIMe is queried when OUTput is TRAce<ui>, the system clocktime is returned. If the digitizer is stopped, the time of the last acquisition for that trace is returned.

Syntax: `WFMpre<sp>TIMe:<qstring>`
`WFMpre?<sp>TIMe`

Range: `<qstring>` = <hh:mm:ss[.nn]>

where hh is the hour in 24-hour format,
mm is the minute,
ss is the second, and
nn is hundredths of a second (optional).

Examples

```
WFM TIM:'17:15:13'
```
specifies the time stamp for the trace.

TSTime

Query only.TSTime returns the time between the actual trigger point and the waveform sample identified as 0 seconds.

Syntax: `WFMpre?<sp>TSTime`

Examples

```
WFM? TST
```
returns the time between the trigger time and point 0, such as:

```
WFMPRE TSTIME:1.48001E-7
```

WFId

Query Only. WFId identifies the source waveform for this preamble. The information returned by this link is the same as that returned by an OUTput? query, unless OUTput specifies a label. If a label is specified, then it returns either STO<ui> or TRA<ui> matching the label. See the OUTput command.

Syntax: `WFMpre?<sp>WFId`

Examples

```
WFM? WFI
```
returns the source trace for this preamble, such as:

```
WFMPRE WFID:TRACE7
```

XINcr

XINcr specifies the horizontal sample interval of a YT waveform. The range begins at 1 ps per point.

Syntax:
```
WFMpre<sp>XINcr:<NRx>
WFMpre?<sp>XINcr
```

Range: `<NRx>` ≥1E–12 sec/pt

Examples

```
WFM XIN:1.0E-9
```
sets the horizontal sampling interval of a YT waveform.

XMUlt

Query Only. XMUlt returns the vertical scale factor, in XUNit per unscaled data point value, of the horizontal component of an XY waveform.

Syntax: `WFMpre?<sp>XMUlt`

Returns: `<NR3>`

Note: For XMUlt usage, refer to the waveform scaling formulas in the CURVe entry.

Examples

```
WFM? XMU
```
returns the vertical scale factor of an XY trace, such as:
```
WFMPRE XMULT:1.0E-1
```

XUNit

XUNit specifies the horizontal units (X-axis) of the waveform data at the time of waveform creation. For YT waveforms, XUNit specifies the units of the horizontal axis in seconds or hertz. For XY waveforms, XUNit is the vertical units of the horizontal component. XUNit returns DIVS when the units of the waveform are indeterminate or undefined.

Syntax:
```
WFMpre<sp>XUNit:{AMPS|DB|DEGrees|DIVS|
                 HERtz|OHMs|SEConds|
                 VOLts|WATts}
WFMpre?<sp>XUNit
```

Examples

```
WFM XUN:SEC
```

XZEro

XZEro specifies the number of seconds of pre-trigger or post-trigger of a YT waveform; or it specifies the vertical offset of the horizontal component of an XY waveform.

Syntax:
```
WFMpre<sp>XZEro:<NRx>
WFMpre?<sp>XZEro
```

Range: `<NRx>` = −1E+15 to 1E+15

Examples

```
WFM XZE:2.5E-2
```

YMUlt

YMUlt specifies the vertical scale factor, in YUNit per unscaled data point value, of a YT waveform, or specifies the vertical scale factor, in YUNit per unscaled data point value, of the vertical component of an XY waveform. (YMUlt is equal to the vertical units-per-division, such as volts, divided by 6400.)

Syntax:
```
WFMpre<sp>YMUlt:<NRx>
WFMpre?<sp>YMUlt
```

Range: <NRx> = 1E−15 to 1E+15

Examples

```
WFM YMU:1.5625E-4
```
sets the vertical scale factor.

YUNit

YUNit specifies the vertical units (Y-axis) of the waveform data (YT or XY) to be transferred via the remote interfaces. Querying YUNit returns DIVS when the units of the waveform are indeterminate or undefined.

Syntax:
```
WFMpre<sp>YUNit:{AMPS|DB|DEGrees|DIVS|
                 HERtz|OHMs|SEConds|
                 VOLts|WATts}
WFMpre?<sp>YUNit
```

Examples

```
WFM YUN:VOL
```
sets the vertical units of the trace data.

YZEro

YZEro specifies the vertical offset of a YT waveform, or specifies the vertical offset of the vertical component of an XY waveform.

Syntax:
```
WFMpre<sp>YZEro:<NRx>
WFMpre?<sp>YZEro
```

Range: <NRx> = −1E+15 to 1E+15

Examples

```
WFM YZE:6.25E+1
```
sets the vertical offset.

WFMScaling

WFMScaling determines whether a new waveform is created in floating-point mode (FORce) or integer mode when possible (OPTional). Certain waveform types require floating-point mode regardless of the WFMScaling setting. (For example, stored waveforms are stored in floating-point mode.)

Syntax:

```
WFMScaling<sp>{FORce|OPTional}
WFMScaling?
```

Arguments

- **FORce** – all waveforms including single channel acquisitions (e.g., L1, R2), are created in floating-point mode.
- **OPTional** – integer mode, implies that no floating-point operations are used to display or position waveforms when possible.

Note: Waveforms created in integer mode have faster display update rates.

You can display the following waveform description types in integer mode:

Waveform Types Displayable in Integer Mode

Description	Example
A channel (<slot> <ui>)	C1
Average of a channel	AVG(C1)
Envelope of a channel	ENV(C1)
Inversion of a channel	-C1
Addition of channels	C1 + L2
Subtraction of channels	C1-L2
Combinations of the above	AVG(C1 + L2)

The following are some of the waveform types that you cannot display in integer mode:

Waveforms Not Displayable in Integer Mode

Waveform Type	Example
Stored waveform	STO11
Scalar value	2.23
Stored waveform plus scalar value	STO11 + 2.23
Any waveform using division	L1/L2
Any waveform using multiplication	R1 * R2
Any waveform using a floating point function	ABS(L1)

Examples

```
WFMS OPT
```
creates a new waveform in integer mode if possible.

WIDth

Query Only. WIDth returns the time a signal takes to go from one MESial voltage level crossing to the next MESial crossing of the opposite slope, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `WIDth?`

Returns: `<NR3> or <bblock>`

Examples

```
WID?
```

returns the time it takes to go between voltage level crossings of the opposite slope, such as:

```
WIDTH 5.009E-7,EQ
```

WIN1Pos WIN2Pos

WIN1Pos and WIN2Pos sets the position of the Window 1 or Window 2 acquisition records, respectively, relative to the Window trigger.

Syntax:
```
WIN{1|2}Pos<sp><NRx>
WIN{1|2}Pos?
```

Range: WIN1Pos or WIN2Pos range when WTMode is MAIn or EVHoldoff:

MAINPos - *win duration* to MAINPos + *main duration*

WIN1Pos or WIN2Pos range when WTMode is TIHoldoff:

- (TRWin TIHoldoff - MAINPos + *win duration*) to *(main duration* + MAINPos - TRWin TIHoldoff)

Refer to page 274 for the *duration* calculation.

Refer to the WTMode command.

Examples

```
WIN1P 0;WIN2P -1.35
```

positions the acquisition record relative to the window trigger.

WTMode

WTMode sets window triggering mode.

Syntax:
```
WTMode<sp>{EVHoldoff|MAIn|TIHoldoff}
WTMode?
```

Arguments

- **EVHoldoff** – the Window trigger is held off for the number of events specified by TRWin EVHoldoff.
- **MAIn** – the Window trigger coincides with the Main trigger; the Window trigger is not held off.
- **TIHoldoff** – the Window trigger is held off for the time specified by the trigger holdoff (TRWin TIHoldoff).

Note: When WTMode is set to MAIn, the DSA does not check whether the Main and Window triggers are chopped. When WTMode is changed to EVHoldoff or TIHoldoff, the DSA checks if the triggers are chopped. Refer to page 303 for more information on trigger chopping.

Examples

`WTM EVH`
holds off window triggering for a specific number of events.

YTEnergy

Query Only. YTEnergy returns the energy (in squared volts) under the curve of a YT waveform, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `YTEnergy?`

Returns: `<NR3> or <bblock>`

Examples

`YTE?`

returns the energy under the curve, such as:

```
YTENERGY 8.442E-7,EQ
```

YTMns_area

Query Only. YTMns_area returns the difference between the area under a YT curve above a specified reference level, and the area under the curve below that level, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) The reference level is set with the REFLevel command. Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `YTMns_area?`

Returns: `<NR3> or <bblock>`

Examples

`YTM?`

returns the difference between the energy under a curve above and below a specified reference level, such as:

```
YTMNS_AREA 3.332E-7,EQ
```

YTPls_area

Query Only. YTPls_area returns the total, absolute value of all areas between a YT waveform and a reference level set with REFLevel, followed by an accuracy qualifier. (Refer to page 192 for qualifier definitions.) Output encoding is determined by the ENCDG MEAS command. See MEAS? for <bblock> format.

Syntax: `YTPls_area?`

Returns: `<NR3> or <bblock>`

Examples

`YTP?`

returns total value of all areas between a YT trace and a reference level, such as:

```
YTPLS_AREA 1.052E-9,EQ
```

Status and Events

The DSA 601A and DSA 602A Digitizing Signal Analyzers provide a status and event reporting system for the GPIB and RS-232-C interfaces. The status and event system alerts you to significant conditions and events that occur within the DSA.

The status and event system has two principal subsystems:

- The status reporting subsystem is based on the service request (SRQ) function defined by IEEE STD 488 for the GPIB interface. It provides a single byte of general status information. For the RS-232-C interface, the STBYTE? query command provides essentially the same function.
- The event reporting subsystem is defined by the Tektronix Codes and Formats Standard using the EVENT? query command. This query provides more detailed information about the specific event that has occurred. The EVENT? response may be reported to either the GPIB or the RS-232-C interface.

A controller always has the option of reading or ignoring the event code(s) associated with a given status byte.

Status Reporting

The status reporting subsystem includes:

- Status Byte for conveying the type of event that has occurred.
- RQS command for GPIB asynchronous service requests and status messages.
- SRQMASK command for masking event conditions.
- STBYTE? query for RS-232-C polled status messages.
- RS232 VERBOSE command for RS-232-C asynchronous status messages.
- System Status Conditions for reporting categories of events, such as command errors and internal warnings.

Status Byte Definition

The table below describes the individual bits in the status byte. Bit 8 is the most significant bit of the status byte. DIO is an IEEE STD 488 abbreviation for Data Input Output.

Status Byte Definitions

DIO Bit #	Meaning
1 2 3 4	System status bits. The state of these four bits varies with the type of event that is reported.
5	Busy bit. Asserted only when DSA diagnostics are in progress.
6	Error bit. Asserted when an internal or external error condition generates an event.
7	RQS (request service) bit. Asserted when the DSA requests service from a GPIB controller.
8	Never asserted (bit DIO8 is always 0).

RQS Command

The IEEE STD 488 Service Request function (SRQ) permits a device to asynchronously request service from a GPIB controller whenever the device detects some noteworthy event. A GPIB controller services the request by serial polling each active device on the bus. A device responds to the serial poll by placing an 8-bit status byte on the bus. The controller determines the device-asserted SRQ by serially reading the status byte of each device and examining bit 7. If a particular device has requested service, bit 7 of its status byte is set. Otherwise, bit 7 is clear. (Refer to the Binary and Decimal Status Byte Codes table). The RQS command turns on the SRQ function in the DSA.

RQS only affects status and event reporting at the GPIB port. RQS has two major effects:

- It controls bit DIO7 of the status byte. The RQS ON command enables DIO7 assertion. The RQS OFF command disables assertion for all conditions, except power-on. At instrument power-on, RQS is on at the GPIB port and off at the RS-232-C port.
- The RQS command also controls whether or not the DSA is permitted to request service from a GPIB controller. The RQS OFF command disables service requests. The RQS ON command enables service requests.

RQS for GPIB service requests – causes the DSA to assert the SRQ signal line whenever a new event occurs and RQS is set to on. A GPIB controller may then interrogate the DSA with an IEEE STD 488 serial poll and obtain a status byte that describes the event that occurred.

When RQS is set to off, the only new event that will cause the DSA to assert SRQ is power-on. Thus, a GPIB controller will not be informed asynchronously (with SRQ) that an event has occurred. In this situation, a controller may still interrogate the DSA with an IEEE STD 488 serial poll to read the most recent status byte from the serial poll register of the DSA.

RQS for RS-232-C service requests – is always set to off at the RS-232-C port. There is no SRQ signal line for the RS-232-C interface. No asynchronous messages are sent to the controller. Thus, an RS-232-C controller is required to query (poll) the DSA to determine the latest status condition that has occurred in the DSA.

SRQMASK Command

Regardless of whether RQS is on or off, there may be occasions when you want to disable event reporting for a specific class of system conditions. Use the SRQMASK command to disable (mask off) a specific category of events. The event tables later in this section include the SRQMASK for each event type.

STBYTE? Query-only Command

The STBYTE? query allows RS-232-C controllers to read the status byte of the most recent event reported to the RS-232-C port.

The response to the STBYTE? query is:

STBYTE <*NR1*>

where <*NR1*> is a decimal number representing a status condition. (Status byte conditions are defined on page 338.)

RS232 Verbose Mode

RQS is always off for the RS-232-C interface. Therefore, no new instrument event will cause the DSA to request service.

However, in addition to polling the DSA using the STBYTE? query, the RS-232-C interface includes another means to synchronously report status messages, RS232 VERBOSE mode. This mode is turned on or off by using either the **RS232** pop-up menu in the Utility 2 major menu, or the RS232 VERBOSE command.

When VERBOSE is set to ON, each command sent to the DSA always returns an appropriate status message. (For more information on verbose mode, see the discussion on page 25.)

System Status Conditions

The status byte indicates nine system status conditions. System status conditions are divided into two categories: normal (DIO6 clear) and abnormal (DIO6 set).

Five normal conditions are defined:

- **No Status To Report** reports when there is no event or device-dependent status to report.
- **Power On** reports when the DSA has finished its power-on sequence.
- **Operation Complete** tells the controller that a time-consuming task has been completed.
- **User Request** reports when the RQS icon is selected at the front panel.
- **Calibration Due** reports when self-calibration is due.

Five abnormal conditions are defined:

- **Command Error** reports when a message cannot be parsed or lexically analyzed.
- **Execution Error** reports when a message is parsed but cannot be executed.
- **Internal Error** reports if the DSA malfunctions.
- **Execution Warning** reports when the DSA is operating, but these results may be inaccurate.
- **Internal Warning** reports when the DSA detects a problem. The instrument remains operational, but the problem should be corrected.

A list of the binary and decimal codes that correspond to the previously described system status conditions is provided in the following table.

Binary and Decimal Status Byte Codes

	BINARY		**DECIMAL**	
	Status Bits		**RQS**	
Condition	**8765**	**4321**	**ON**	**OFF**
Normal:				
No Status to Report	0000	0000	0	0
Power On	0R00	0001	65	1
Operation Complete	0R00	0010	66	2
User Request	0R00	0011	67	3
Calibration Due	0R00	0110	70	6
Abnormal:				
Command Error	0R10	0001	97	33
Execution Error	0R10	0010	98	34
Internal Error	0R10	0011	99	35
Execution Warning	0R10	0101	101	37
Internal Warning	0R10	0110	102	38

DIO7, shown as "R," is asserted when specifically enabled with the RQS command (GPIB only). Otherwise, the "R" bit is 0 (zero).

Event Reporting

The second subsystem is event code reporting. Event messages more clearly specify the event that has occurred by expanding the description of the status condition reported by the status byte.

GPIB and RS-232-C controllers may read DSA-generated event codes by using the EVENT? query-only command.

The response to an EVENT? is:

EVENT <*NR1*>[,<*qstring*>]

where <*NR1*> represents the numeric value of an event code, and <*qstring*> is a quoted string that describes the returned event code.

The response that includes <*qstring*> is returned only when the LONGFORM command is set to ON.

Event Code Descriptions

For all event classes the event codes and event code description strings are listed in the *Command Errors* table. The event code and event code description is in boldface. Commands that can generate the event code are listed immediately afterwards.

Formatting Symbols such as %A are combined in some of the description strings in the event code tables. When the event is queried, the formatting symbols are expanded, as described in the *Formatting Symbols* table on the next page.

Each formatting symbol begins with a percent sign (%). The symbols indicate that variable information will be substituted when LONGFORM is set to ON.

Formatting Symbols

Symbol	Expand With:
%a	Plug-in channel number or unsigned integer.
%A	Argument name.
%b	Plug-in compartment indicator: L, C, or R.
%B	Plug-in compartment indicator: LEFT, CENTER, or RIGHT.
%c	Probe calibration request string: "Probe gain/offset calibration error".
%C	Calibration request string: "Calibration due".
%d	Time base string: "Main" or "Window".
%D	Record length integer.
%I	Calibration request string: "Calibration due".
%M	A calibration fault string for the DSA. If no error occurred, %M is replaced by "Pass"; otherwise %M is replaced by a short descriptive string describing what caused the mainframe failure; for example, "Main Fine Holdoff."
%N	A stored waveform number or the number of stored waveforms.
%O	Option description string (for example, "Option 4C—Nonvolatile RAM").
%P	Plug-in compartment fault list. If there are no plug-in unit failures, %P is replaced with "NONE". Otherwise %P will be replaced with a comma-delimited list of plug-in compartments, "LEFT," "CENTER," or "RIGHT," according to which compartments reported failures.
%R	Error string for a syntax error when recalling an ASCII format stored waveform from disk.
%S	The erroneous data when recalling an ASCII format stored waveform from disk.

Formatting Symbols (Cont.)

Symbol	Expand With:
%T	Time, as "X minutes and Y seconds." If X is 0, then "X minutes" is omitted. If Y is 0, then "Y seconds" is omitted.
%W	Calibration request string: "Calibration due".
%?	Event code value.

Command Errors

Command errors are reported when a message cannot be parsed or lexically analyzed. Command errors have event codes from 100 to 199. The SRQMASK for command errors is SRQMASK CMDERR. The status byte for a command error returns **97** (decimal) with RQS set to ON, and **33** (decimal) with RQS set to OFF. All command errors are listed on the following pages.

Command Errors

Event Code	Description	Commands that Generate Code	Explanation
108	**Checksum error in binary block transfer**	SET <*bblock*>	Checksum comparison of binary settings failed. Settings are discarded.
109	**Illegal byte count value on a binary block transfer**	SET <*bblock*>	Binary block byte count of binary settings returned to the DSA exceeds maximum size of front-panel settings.
154	**Invalid number input**		Floating-point value too large or too long.
155	**Invalid string input**		String is too long, is not properly terminated, or contains a NULL character.
156	**Symbol not found**		DSA is unable to find the input symbol in its table.
157	**Syntax error**	Any command	Command was typed incorrectly.
		DELTA	Bad syntax with <*qstring*> argument.
		RQS	Attempted to turn RQS on at RS-232-C port.
		STBYTE?	Attempted to use STBYTE? query from GPIB port.
		TEST	Set or query command appended to TEST command. TEST command is ignored; all other commands are processed normally.
		TRACE<*ui*>	Syntax error found in TRACE expression (for example, "L1 + ", or attempted to create non-acquired trace component (for example, STO <*ui*>, <*NRx*>, or combinations) on WIN1 or WIN2 time base.

Command Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
157 (cont)	**Syntax error**	Trigger Source Expressions	Improper syntax in a trigger source expression; for example, any input sequence that cannot be parsed due to missing arguments, links, or delimiters, or incorrect links with commands. Note that this syntax does not permit RIGHT plug-in channels to be added to (or subtracted from) LEFT or CENTER plug-in channels.
160	**Expression too complex**	TRACE<*ui*>	Trace description exceeds 54 characters for either the vertical or horizontal description or there is insufficient stack space.
		DELTA	DELTA description cannot be parsed due to insufficient stack space.
161	**Excessive number of points in binary CURVE data input**	Waveform Retrieval and Scaling (Data Transfer)	More binary data points were sent than were specified with the WFMPRE NR.PT link.
162	**Excessive number of points in ASCII CURVE data input**	Waveform Retrieval and Scaling (Data Transfer)	More ASCII data points were sent than were specified with the WFMPRE NR.PT link.
163	**No input terminator seen**		RS-232-C input type-ahead buffer has overflowed. All input is discarded.
164	**Binary block input not allowed with ECHO ON**	CURVE <*bblock*> SET <*bblock*>	Attempted to send binary block data through RS-232-C port with echo on. The data are discarded.
167	**Insufficient data to satisfy binary block byte count**	SET?	Binary settings returned to GPIB port were prematurely terminated (for example, binary block byte count is not satisfied when EOI line is asserted).
168	**Unsupported constant**		
169	**Unsupported function**	TRACE<*ui*>	TRACE expression includes unsupported function.

Execution Errors

Execution errors are reported when a message is parsed but cannot be executed. Execution errors have event codes from 200 to 299. The SRQMASK for execution errors is SRQMASK EXERR. The status byte for an execution error returns **98** (decimal) with RQS set to ON, and **34** (decimal) with RQS set to OFF. All execution errors are listed on the following pages.

Execution Errors

Event Code	Description	Commands that Generate Code	Explanation
203	**I/O buffers full**		Both input and output buffers are full. Output buffer is cleared.
205	**%A out of range - value ignored**	ABSTOUCH	Out-of-range ABSTOUCH argument.
		MCALCONSTANTS	Out-of-range <*ui*> values.
211	**Can't change AUTO-ACQ trace selection**	AUTOACQ	Attempted to turn off last trace, select more than eight traces, or select XY trace for repetitive single trigger acquisition.
212	**Record length of delta description reference wfm must be an even multiple of the test wfm record length**	DELTA, TBM, TBW	The delta reference waveform record length must be an even multiple of the test trace record length.
213	**Window trigger select requires window trigger holdoff and window trace.**	CONDACQ	The window trigger mode must not be trigger from main when setting the conditional acquire window source.
214	**That function is incompatible with %O**	DELTA, FFT, HISTOGRAM	Attempted DELTA, HISTOGRAM, or FFT command with Option 3C-Acquisition Memory External Power Supply installed.
215	**Can't undo autoset**	AUTOSET	Attempted AUTOSET UNDO with no previous AUTOSET performed.
216	**Can't spool hardcopy**	COPY	Attempted COPY START when printer spooler is full.
217	**Can't keep scan waveform**	SCANSTOWFM	No current scan waveform exists to keep, or scanning was never started, or the template waveform has changed, or too many traces already exist to create another.
218	**Can't start scanning**	SCANSTOWFM	There are no stored waveforms, or eight traces are already defined, or Repetitive Trigger or Delta is in effect.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
219	**Record length of delta description test wfm cannot be greater than record length of test wfm**	DELTA	Attempted to enter new delta description or attempted to increase record length which conflicts with current delta description.
220	**Connect probe to calibrator and restart operation**	CALPROBE	Attempted to run probe calibration on a channel with no input signal.
221	**Illegal delta description**	DELTA	<*qstring*> argument not specified correctly (such as a non-envelope waveform).
222	**%O needed to support that function**	HSBATT?	Attempted to use an non-existent option.
223	**Illegal base label**	LABEL	Numerals specified as part of base label.
224	**Function not available in selected plug-in range**	CALPROBE	Attempted probe calibration on an 11A33 plug-in amplifier unit with CH<*slot*> <*ui*> PROTECT: ON.
225	**Cannot change label while current acquisition mode is running**	DELTA	Attempted to change stored waveform labels or base label with Act-on-Delta running.
		LABEL	Attempted to change stored waveform labels or base label with Repetitive Single Trigger, REPCURVE, (MS)REP<meas>, or (MS)REPMEAS running.
226	**Trigger timer not available**	Trigger Source Expressions	Trigger description specifies use of a timer more than once.
227	**Not available with Extended Triggering**	TRMAIN, TRWIN, Trigger Source Expressions	Attempted to set MODE to AUTOLEVEL, SLOPE to MINUS, COUPLING to AC, ACHF, or ACNOISE, or WTMODE to EVHOLDOFF or TIHOLDOFF with extended triggering mode active (i.e., a WHILE, AND, OR, or XOR operator appearing in the Main trigger expression).

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
228	**Label not found**	CLEAR, DELETE, INPUT, OUTPUT, RECALL, REMOVE, SELECT, STORE, SCANSTOWFM	No matching label found with <*qstring*> syntax used.
229	**No stored waveforms**	LABEL? STO, SCANSTOWFM	No waveforms were stored when LABEL was queried or SCANSTOWFM FROM or SCANSTOWFM TO link was issued.
230	**Can't set front panel calibrator amplitude**	CALIBRATOR	Attempted to set AMPLITUDE when frequency (FREQ) is 1 kHz or 1 MHz.
231	**Autoset - not functional with this waveform type**	AUTOSET	Attempted to autoset window waveform, which has no "parent" main waveform, when main waveform time base not triggered.
232	**That XY waveform has incompatible components**	TRACE<*ui*>	Attempted to create XY trace with horizontal and vertical components with incompatible scaling modes.
233	**Delayed trace must not be the selected trace**	DLYTRACE	Attempted to specify currently selected trace as PDELAY delayed trace.
234	**Unsupported printer function**	COPY	Format unsupported for currently selected printer.
236	**Illegal color number**	COLOR, HPGL, TEK4692, TEK4696	Out-of-range color index.
237	**No labels defined**	LABEL?	No labels defined for specified links.
238	**Label not defined**	LABEL?	No label is defined for FPS<*ui*>, STO<*ui*>, or TRACE<*ui*> links.
239	**Improper version number**	SET <*bblock*>	Version number of received binary settings block not the same as current firmware version number.
240	**Can't enable persistence for nonacquired waveform**	TRACE<*ui*>	Attempted to enable point accumulate with non-acquired trace.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
241	**Too many acquisitions**	TRACE<*ui*>	Trace definition would cause the DSA to acquire more than 14 total acquisitions.
242	**ENHANCED ACCURACY available after %T**	SELFCAL	Attempted SELFCAL FORCE before 20-minute warmup elapsed.
243	**That function is disabled by a hardware strap**	CCALCONSTANTS, LCALCONSTANTS, RCALCONSTANTS	Attempted to set plug-in unit calibration constants with hardware strap disabled.
		MCALCONSTANTS	Attempted to set DSA calibration constants with hardware strap disabled.
		UID	Attempted to modify serial number with hardware strap disabled.
244	**%B plug-in channel(s) used differently in main and window sources**	Trigger Source Expressions	A channel was "chopped" between the main and window trigger sources, when WTMODE is set to EVHOLDOFF or TIHOLDOFF.
245	**Autoset - only functional with 11K plug-ins**	AUTOSET	Attempted to autoset waveform containing a 7000-series plug-in channel.
246	**Can't sequence settings**	RECALL	Attempted to sequence settings with SETSEQ OFF.
247	**No settings defined**	LABEL?	Attempted to query LABEL? FPS when no settings exist.
		PROBE	Attempted to assign probes to SETSEQ with no settings defined.
		RECALL	Attempted to sequence settings with no settings defined.
		SETSEQ	Attempted to turn SETSEQ on with no settings defined.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
248	**Misuse of AVG/ENV function**	AVG, ENV	Attempted to turn AVG or ENV on when selected trace is XY, or when selected trace is composed only of stored and scalar components. Or attempted to turn AVG or ENV off when selected waveform's vertical description not enclosed by the AVG or ENV function.
		TRACE<*ui*>	Trace description includes AVG or ENV function with a non-acquired argument, such as AVG (ST01) or ENV(1000).
249	**Illegal use of trace positioning function**	ADJTRACE <*ui*>	Attempted to modify HMAG, HPOSITION, HVPOSITION, HVSIZE, TRSEP, VPOSITION, or VSIZE values when modification is not permitted (for example, when PANZOOM is off).
250	**No traces defined**	ADJTRACE?, LABEL?, TRACE?	Query attempted with no traces displayed.
		LABEL	Attempted to label or delete a label on a trace when no traces are currently displayed.
		AVG, CURSOR, DOT1ABS, DOT2ABS, DOT1REL, DOT2REL, ENV, H1BAR, H2BAR, V1BAR, or V2BAR	Attempted to set or query one of these commands with no traces defined.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
250 (cont)	**No traces defined**	BASELINE, DAINT, DISTAL, DLYTRACE, LMZONE, MESIAL, MSLOPE, MTIME,PROXIMAL, REFLEVEL, REFTRACE, RMZONE, SNRATIO, TOPLINE	Attempted to set or query one of these measurement commands with no selected trace.
251	**Illegal trace number**	ADJTRACE, AUTO-ACQ, CLEAR, COL-OR MAP, CURSOR, DELTA, DLYTRACE, LABEL, OUTPUT, REFTRACE, RE-MOVE, REPMEAS, SELECT, STORE, TRACE	Out-of-range <*ui*> argument with one of these commands.
252	**Illegal stored settings number**	DELETE, LABEL, RECALL, STORE	Out-of-range <*ui*> argument with one of these commands.
253	**%B plugin chan-nel(s) used different-ly in first and second term of main source**	TRM	Trigger source uses channels from the same plug-in differently on either side of a boolean expression.
254	**Trigger timer not available. Window trigger source modi-fied**	WTMODE	This occurs when changing WTMODE from triggered by main, if the trigger timer is used more than once.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
255	**Out of memory**	DELTA	Insufficient memory to create DELTA DESCRIPTION:<*qstring*>.
		DIGITIZER	Insufficient memory to save repetitive trigger or delta waveform.
		STORE	Insufficient memory to store a trace or insufficient NVRAM to store settings.
		SCANSTOWFM	KEEP link or MODE:SCAN link sent; insufficient memory to create another displayed trace.
		TRACE	Insufficient memory to create a new trace.
		Waveform Retrieval and Scaling (Data Transfer)	INPUT command references nonexistent stored waveform, insufficient memory to create stored waveform record.
		WFMPRE	Insufficient memory to create stored waveform record for preamble.
257	**Illegal stored waveform number**	DELETE, DELTA, INPUT, LABEL, OUTPUT, STORE, TRACE	Out-of-range STO <*ui*> argument for one of these commands.
		SCANSTOWFM	Argument of FROM or TO link not valid stored waveform.
263	**Illegal channel number**	CH <*slot*> <*ui*>	Attempted to set query parameters of plug-in channel that does not exist.
		TRACE <*ui*>	TRACE expression references illegal plug-in channel.
		Trigger Source Expressions	Trigger Source Expression references nonexistent 11000-series channel number.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
264	**No further XY waveforms may be defined**	TRACE<*ui*>	Attempted to define more than the maximum permissible number of XY traces.
265	**Illegal DATE/TIME**	DATE	Illegal date value or syntax specified.
		TIME	Illegal time value or syntax specified.
		WFMPRE	Invalid date or time string entered. The date or time is set to the current clock value.
266	**DEF expansion overflow**	DEF	Expansion string overflowed internal expansion buffer.
267	**Illegal DEF string**	DEF	Illegal logical name specified.
268	**Illegal DEF recursion**	DEF	Unacceptable DEF recursion detected. Recursive logical names are acceptable only when recursion occurs to the right of an unquoted semicolon.
269	**No such trace**	ADJTRACE, AUTO-ACQ, CLEAR, CURVE, DELTA, LABEL, REMOVE, SELECT, STORE, TRACE, WAVFRM?, WFMPRE	Referenced, or attempted to set or query parameters of a nonexistent trace using one of these commands.
270	**No such stored waveform**	CURVE?	CURVE? query attempted, OUTPUT references nonexistent stored waveform.
		DELETE	Attempted to delete nonexistent stored waveform.
		DELTA	Attempted to reference a nonexistent stored waveform.
		LABEL	Attempted to label or query for a label of a nonexistent stored waveform.
		TRACE <*ui*>	TRACE expression referenced legal but undefined stored waveform.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
270 (cont)	**No such stored waveform**	WAVFRM?	WAVFRM? query attempted, OUTPUT referenced nonexistent stored waveform.
		WFMPRE	WFMPRE? query attempted, OUTPUT referenced nonexistent stored waveform.
271	**No such DEF**	UNDEF	UNDEF argument not defined in current list of logical names.
272	**That function is not supported by this plug-in**	CH<*slot*> <*ui*>	Attempted to set or query AMPOFFSET, MNSCOUPLING, MNSOFFSET, MNSPROBE, PLSCOUPLING, PLSOFFSET, PLSPROBE, PROTECT, or VCOFFSET links of nondifferential amplifier, or attempted to set or query COUPLING or PROBE linked of differential amplifier.
			Attempted to set or query BWHI or BWLO parameters of plug-in unit that does not support the high/low bandwidth limit function, or attempted to set or query BW parameter of plug-in unit that supports high/low bandwidth limits.
		Trigger Source Expressions	Attempted to invert the trigger channel for an 11A71 plug-in unit.
273	**No such FPS**	DELETE	Attempted to delete undefined stored settings number.
		LABEL	Attempted to label or query undefined stored settings number.
		RECALL	Attempted to recall undefined stored settings number. In this context, "undefined" refers to previously deleted settings or settings that have never been initialized.
274	**No appropriate 11K plug-ins loaded**	CH?	CH? query attempted, DSA has no plug-in units that support 11000-series generic plug-in unit interface.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
275	**%B slot not loaded with appropriate 11K plug-in**	CCALCONSTANTS, LCALCONSTANTS, RCALCONSTANTS	Attempted to set or query calibration constants of plug-in compartment not loaded with 11000-series plug-in unit.
		CH <*slot*> <*ui*>	Attempted to set or query parameters of plug-in compartment not loaded with 11000-series generic plug-in unit.
		TRACE <*ui*>	TRACE expression references 7000-series or missing plug-in unit as 11000-series signal.
		Trigger Source Expressions	Trigger Source Expression references plug-in compartment not loaded with 11000-series generic plug-in unit as 11000-series trigger channel.
276	**%B slot not loaded with 7K plug-in amplifier**	TRACE <*ui*>	TRACE expression references 11000-series plug-in unit as 7000-series signal.
		Trigger Source Expressions	Attempted to reference 11000-series plug-in unit as 7000-series channel; for example, if the left compartment is loaded with an 11000-series plug-in unit, any attempted to refer to the left compartment as an "L" trigger channel returns this event code.
277	**Misuse of 7K plug-in**	Trigger Source Expressions	Same 7000-series channel used more than once in trigger expression, or attempted to invert 7000-series trigger channel in expression (for example, TRM SOU:"-C" or TRW SOU:"L-C").
278	**Plug-in channel used more than once in trigger source**	Trigger Source Expressions	Same 11000-series channel used more than once in trigger expression.
279	**Line trigger not available for window trigger source**	Trigger Source Expressions	Attempted to set source of window time base to line.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
281	**Can't delete active stored waveform**	DELETE	Attempted to delete stored waveform that is a component of a combined active trace.
		WFMPRE	Returning WFMPRE data would cause deletion of a stored waveform that is not the sole component of a waveform description of a displayed trace. The WFMPRE data are discarded.
282	**Can't store trace**	STORE	Attempted to store XY trace, or attempted to copy a trace over an existing stored waveform when the two waveforms do not have equal record lengths.
283	**Can't clear nonacquired waveform**	CLEAR	Attempted to clear trace that has only stored waveform components (for example, TRACE1 DESCRIPTION:"STO3").
284	**Requested coupling for channel %a not available on %B plug-in**	CH <*slot*> <*ui*>	Attempted to set coupling to value not supported by plug-in unit, or attempted to set plug-in channel's coupling to value not allowed because a Level 2 TekProbe is connected to that channel, or attempted to set coupling to value that would increase overload on input channel.
285	**Requested input impedance for channel %a not available on %B plug-in**	CH <*slot*> <*ui*>	Attempted to set a plug-in channel to impedance value not allowed because Level 2 TekProbe is connected to that channel, or attempted to set impedance to value that would increase overload on input channel.
286	**Too many measurements specified**	MSLIST	More than six measurements specified.
287	**Hardcopy absent or off line**	COPY	CENTRONICS port specified as COPY output port, printer not connected to port or currently connected printer is offline.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
288	**Inappropriate trigger level units**	TRMAIN	Improper ANLEVEL units specified.
		TRWIN	Improper NLEVEL units specified.
289	**Split cursors not permitted on XY trace**	CURSOR	Attempted to SPLIT cursors across XY trace.
290	**Current reference measurement failed**	REFSET	CURRENT reference cannot be computed due to one of the following conditions: ▪ No waveforms are defined (regardless of measurement). ▪ Selected waveform is XY or point accumulate trace (regardless of measurement). ▪ Reference measurement specified as DUTY, FREQ, or PERIOD; no period can be found within specified measurement zone. ▪ Reference measurement specified is MEAN, RMS, YTENERGY, YTMNS_AREA, or YTPLS_AREA while DAINT is set to SINGLE; no period can be found within specified measurement zone. ▪ Reference measurement specified is CROSS and REFLEVEL does not fall between computed maximum and minimum of specified measurement zone. ▪ Reference measurement specified is RISETIME and measurement system cannot compute valid proximal and distal time within specified measurement zone.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
290 (cont)	**Current reference measurement failed**	REFSET	▪ Reference measurement specified is FALLTIME and measurement system cannot compute valid distal and proximal time within specified measurement zone. ▪ Reference measurement specified is WIDTH and two mesial crossings of opposite slope cannot be found within specified measurement zone. ▪ Reference measurement specified is GAIN, PHASE, or PDELAY and only one trace is defined.
292	**%B slot not loaded with 11K plug-in**	UID	Attempted to set or query serial number of plug-in compartment not loaded with 11000-series plug-in unit.
295	**Record length too long for Persistence waveform**	SELECT	Attempted to select point accumulate waveform whose time base length is greater than 2048 points. The point accumulate waveform becomes the selected trace but is displayed in normal Yt format.
		TBMAIN/TBWIN	Attempted is made to increase time base length of point accumulate waveform to more than 2048 points.
		TRACE <*ui*>	Attempted to enable point accumulate with time base record length greater than 2048 points.
297	**Panzoom may not be enabled**	ADJTRACE <*ui*>	Attempted to enable PANZOOM for XY trace.
298	**Panzoom may not be disabled**	ADJTRACE <*ui*>	Attempted to disable PANZOOM for stored or scalar trace, or for FFT magnitude phase traces.

Execution Errors *(Cont.)*

Event Code	Description	Commands that Generate Code	Explanation
299	**CONDACQ function not available**	CONDACQ	AVG or ENV conditional acquisition specified, but no traces include AVG or ENV function in trace descriptions.
			CONDACQ set to BOTH, but the following condition does not exist: ▪ At least one waveform description includes the AVG function and at least one other waveform description includes the ENV function. ▪ One waveform includes both AVG and ENV in its description.
			Conditional acquisition of any type except CONTINUOUS specified, with no traces defined.
			DELTA conditional acquisition specified, but no valid delta description exists.
2001	**No delta waveform exists**	OUTPUT	Attempted to set OUTPUT to DELTA when there is no valid delta trace.
2002	**That trace is not a live frequency domain trace**	ADJTRACE<ui>	Attempted to set FSPAN or FRES links of a trace that is not a frequency domain trace.
2003	**Waveforms cannot depend on themselves**	TRACE<ui>	Circular dependencies in trace descriptions are not allowed.
2004	**Can't start REPMEAS**	REPMEAS	Attempted to start repetitive measurements when it is not possible to do so.
2005	**Can't start MSREPMEAS**	MSREPMEAS	Attempted to start repetitive statistical measurements when it is not possible to do so.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
2006	**Illegal text number**	TEXT<ui>	Out of range <ui> argument for text.
2010	**Unrecognized file name extension**	TRACE<ui>	Filename specified in DESCRIPTION <qstring> has an invalid extension.
		RECALL DCOPY	Filename specified in <qstring> has an invalid extension.
2100	**File not found**	DCOPY	Attempted to copy a non-existent file from disk.
		DELETE	Attempred to delete a non-existent file from disk.
		DELTA	File specified as part of DESCRIPTION <qstring> does not exist.
		RECALL	Attempted to recall a non-existent file from disk.
		TRACE	File specified as part of DESCRIPTION <qstring> does not exist.
2101	**Invalid path**	DCOPY DELETE	Path specified in <qstring> argument is invalid.
		DELTA	Path specified in DESCRIPTION <qstring> argument is invalid.
		MKDIR RECALL RMDIR STORE	Path specified in <qstring> argument is invalid.
		TRACE	Path specified in DESCRIPTION <qstring> argument is invalid.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
2102	**Invalid device name**	CHDIR CHKDSK DCOPY DELETE	Device specified in <qstring> argument is invalid.
		DELTA	Device specified in DESCRIPTION <qstring> argument is invalid.
		FORMAT MKDIR RECALL RMDIR STORE	Device specified in <qstring> argument is invalid.
		TRACE	Device specified in DESCRIPTION <qstring> argument is invalid.
2103	**Disk is write protected**	CHKDSK	The write protect tab on the disk is set to the read-only postion.
		DCOPY	Attempted to copy to the disk when the write protect tab is set to the read-only postion.
		DELETE	Attempted to delete a file when the write protect tab is set to the read-only postion.
		FORMAT	The write protect tab on the disk is set to the read-only position.
		LABEL	Attempted to label the disk when the write protect tab is set to the read-only position.
		MKDIR	Attempted to create a new directory when the write protect tab is set to the read-only position.
		RMDIR	Attempted to remove a directory when the write protect tab is set to the read-only position.
		STORE	Attempted to store to the disk when the write protect tab is set to the read-only position.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
2104	**Bad media**	CHKDSK DCOPY DELETE DELTA DIR?	The currently installed disk is unreadable.
		FPSLIST? FPSNUM?	When SETDEV FPS is DISK and the currently installed disk is unreadable.
		FORMAT	The currently installed disk is unreadable.
		LABEL	Attempted to label the disk when the currently installed disk is unreadable.
		MKDIR	The currently installed disk is unreadable.
		RECALL	Attempted to recall a setting from the disk when the currently installed disk is unreadable.
		RMDIR	The currently installed disk is unreadable.
		STOLIST? STONUM?	When SETDEV STO is DISK and the currently installed disk is unreadable.
		STORE	Attempted to store to the disk when the currently installed disk is unreadable
		TRACE	DESCRIPTION <qstring> contains a file name and the currently installed disk is unreadable.
2105	**Device error**	CHKDSK DCOPY DELETE DELTA DIR?	No disk is installed.
		FPSLIST? FPSNUM?	When SETDEV FPS is DISK and no disk is installed.
		FORMAT	No disk is installed.
		LABEL	Attempted to label the disk when no disk is installed.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
2105 (cont.)		MKDIR	No disk is installed.
		RECALL	Attempted to recall a setting from the disk when no disk is installed.
		RMDIR	No disk is installed.
		STOLIST? STONUM?	When SETDEV STO is DISK and no disk is installed.
		STORE	Attempted to store to the disk when no disk is installed.
		TRACE	DESCRIPTION <qstring> contains a file name and no disk is installed.
2106	**Directory full**	DCOPY	Attempted to copy to the disk when the root directory is full.
		MKDIR	Attempted to create a new directory when the root directory is full.
		STORE	Attmpted to store to the disk when the root directory is full.
2107	**Directory not empty**	RMDIR	Attempted to remove a directory that was not empty.
2108	**Invalid volume label**	FORMAT	Specified an invalid volume label.
		LABEL	Attempted to label the disk with an invalid label.
2109	**Disk Full**	DCOPY	Attempted to copy to the disk when it is full.
		MKDIR	Attempted to create a new directory when the disk is full.
		STORE	Attempted to store to the disk when it is full.
2110	**Duplicate file exists**	DCOPY	Attempted to copy to a file that already exists.
		STORE	Attempted to store to a file that already exists.

Execution Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
2112	**Device timed out**	DIR?	No disk is installed.
2114	**Disk changed**	COPY	Disk is removed and a different disk installed while hardcopy is writing to disk.
		DCOPY	Disk is removed and a different disk installed while writing to disk.
		STORE	Disk is removed and a different disk installed while writing to disk.
2115	**Current path**	RMDIR	Attempted to remove a directory that is part of the current path.
2116	**Duplicate directory name**	MKDIR	Attempted to create a new directory with the same name as an existing directory.
2117	**Different device**	RENAME	Attempted to rename a file with different devices.
		RENDIR	Attempted to rename a directory with different devices.
2118	**Error in reading data**	DCOPY RECALL TRACE	Read error while reading from the disk.
2119	**Access denied**	DELETE	Attempted to delete a read-only file.

Internal Errors

Internal errors are reported if the DSA malfunctions. Internal errors have event codes from 300 to 399. The SRQMASK for internal errors is SRQMASK INERR. The status byte for internal errors returns **99** (decimal) with RQS set to ON, and **35** (decimal) with RQS set to OFF. All internal errors are described on the following pages.

Internal Errors

Event Code	Description	Commands that Generate Code	Explanation
308	**Bad level 2 probe checksum on channel %b%a**	CH<*slot*> <*ui*>	Level 2 TekProbe improperly connected to input channel, or Level 2 TekProbe properly connected, but malfunctioned and needs repair.
328	**DIG plug-in ENHANCED ACCURACY failed**	SELFCAL	Digitizer plug-in calibration failed. Internal digitizer error.
329	**Deskew failed: %c**	SELFCAL	Digitizer was unable to deskew a channel.
330	**ENHANCED ACCURACY failed. Mainframe: %M Plug-in: %P**	SELFCAL	ENHANCED ACCURACY initiated, but failed.
331	**Probe calibration failed: %c**	CALPROBE	Probe calibration initiated, but subsequently failed.
332	**Partial ENHANCED ACCURACY failed. Plug-in: %P**	Power on	Automatic calibration of new plug-in configuration failed.
394	**Test completed and failed**	TEST	Self-tests diagnostics or extended diagnostics completed and failed.
395	**General DIG failure detected (code = %a)**		Digitizer detected an internal error.

Internal Errors (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
396	**%B plug-in communication failure**	CH <*slot*> <*ui*>	DSA detects that communication is no longer possible with a particular plug-in unit. The DSA may continue to operate, depending on the type of message that was in progress when communication failure occurred. Any of the following problems may exist: failed hardware, a software bug, or a plug-in unit was removed after the DSA was powered up, and communication was then attempted with the empty plug-in compartment.
397	**Internal DAC overflow on channel %a of %B plug-in**	CH <*slot*> <*ui*>	A plug-in unit detects that the requested setting overflowed an internal DAC. The plug-in unit sets the DAC to the limit nearest the requested setting. This event code usually indicates failed hardware.
398	**Invalid DIG table ID detected**		Digitizer detected an invalid table ID.
399	**Invalid DIG field ID detected**		Digitizer detected an invalid field ID.

System Events

System events are normal conditions of the system and are listed on the following pages. System events have event codes from 400 to 499. The SRQMASK for each event is included in the table.

Note: Event 400 (system function normal) and event 401 (power on) cannot be masked with SRQMASK.

System Events

Event Code	Description	SRQMASK	Status Bytes		Commands that Generate Code
400	**System function normal**	-none-	0	0	
401	**Power on**	-none-	65	1	
403	**Front panel RQS icon selected**	USER	67	3	
450	**Conditional acquire complete**	OPCMPL	66	2	DIGITIZER
451	**Abstouch**	ABSTOUCH	67	3	ABSTOUCH
457	**Probe %a ID button pressed on %B plug-in**	IDPROBE	67	3	
458	**Hardcopy aborted**	OPCMPL	66	2	COPY
460	**Test completed and passed**	OPCMPL	66	2	TEST
461	**ENHANCED ACCURACY completed and passed**	OPCMPL	66	2	SELFCAL
462	**Hardcopy complete**	OPCMPL	66	2	COPY
463	**Measurements complete**	OPCMPL	66	2	
464	**Autoset complete**	OPCMPL	66	2	AUTOSET
465	**Warmup complete - %C**	CALDUE	70	6	
466	**New configuration - partial ENHANCED ACCURACY occurring**	CALDUE	70	6	
467	**Warmup complete with new configuration - %C**	CALDUE	70	6	
468	**Warmup complete with new configuration - automatic ENHANCED ACCURACY occurring**	CALDUE	70	6	
469	**Temperature change - automatic ENHANCED ACCURACY occurring**	CALDUE	70	6	
470	**Temperature change - %C**	CALDUE	70	6	
471	**Warmup complete - ENHANCED ACCURACY in effect**	CALDUE	70	6	
472	**Warmup complete - automatic ENHANCED ACCURACY occurring**	CALDUE	70	6	

System Events (Cont.)

Event Code	Description	SRQMASK	Status Bytes		Commands that Generate Code
473	Front panel recall complete	OPCMPL	66	2	RECALL
474	INIT complete	OPCMPL	66	2	INIT
475	Probe calibration completed and passed	OPCMPL	66	2	CALPROBE
476	Temperature change - %I	OPCMPL	66	2	
477	Warmup complete with new configuration - %W	OPCMPL	66	2	
478	Warmup complete - ENHANCED ACCURACY in effect. Compensate probe to use max Real Time sample rate	OPCMPL	66	2	
479	Partial ENHANCED ACCURACY completed and passed	OPCMPL	66	2	
480	Single acquistion complete	OPCMPL	66	2	DIGITIZER, REP-CURVE
481	Number of stored waveforms re-coved:%N	OPCMPL	66	2	RECOVER
482	Format complete	OPCMPL	66	2	FORMAT
483	Chkdsk complete	OPCMPL	66	2	CHKDSK

Execution Warnings

Execution warnings are reported when the DSA is operating, but may produce inaccurate results. Execution warnings have event codes from 500 to 599. The SRQMASK for execution warnings is SRQMASK EXWARN. The status byte returns **101** (decimal) with RQS set to ON, and **37** (decimal) with RQS set to OFF. All execution warnings are listed on the following pages.

Execution Warnings

Event Code	Description	Commands that Generate Code	Explanation
550	**%A out of range - limit set**	ADJTRACE <*ui*>	HMAG, HPOSITION, HVSIZE, HVPOSITION, TRSEP, VPOSITION, or VSIZE link argument out of range.
		BASELINE	Out-of-range BASELINE argument.
		BIT/NR	Out-of-range argument.
		BYT/NR	Out-of-range argument.
		CALIBRATOR	Out-of-range AMPLITUDE or FREQ link argument
		CH <*slot*> <*ui*>	AMPOFFSET, MNSOFFSET, OFFSET, PLSOFFSET, SENSITIVITY, or VCOFFSET link argument out of range.
		COLOR <*ui*>	HUE, LIGHTNESS, or SATURATION link argument out of range.
		CONDACQ	FILL link argument out of range.
		DELTA	CONSECPTS or TOTALPTS link argument out of range.
		DISPLAY	INTENSITY or PERSIST link argument out of range.
		DISTAL	Out-of-range DISTAL argument.
		DLYTRACE	Out-of-range DLYTRACE argument.
		DOT1ABS, DOT2ABS	PCTG, XDIV, or XCOORD link argument would position dot cursors off waveform record of selected trace.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
550 (cont)	**%A out of range - limit set**	DOT1REL, DOT2REL	PCTG, XDIV, or XCOORD link argument would position dot cursors outside limits specified for corresponding DOT1ABS/ DOT2ABS links.
		H1BAR, H2BAR	Out-of-range YCOORD or YDIV link argument.
		HISTOGRAM	Out-of-range REFRESH, {C|D}.WINTOP, {C|D}.WINBOTTOM, {C|D}.WINLEFT, {C|D}.WINRIGHT assignment.
		HNUMBER	Out-of-range argument.
		HPGL	Out-of-range plotter pen assignment.
		LABABS	Out-of-range <*NRx*> value.
		LABREL	Link argument would position label outside limits specified for LABABS command.
		LMZONE	Out-of-range LMZONE argument.
		MAINPOS, WIN1POS, WIN2POS	Out-of-range MAINPOS, WIN1POS, or WIN2POS argument. The valid range of WIN1POS and WIN2POS depend upon the current value of MAINPOS. Thus, changing the value of MAINPOS can cause the current value of WIN1POS or WIN2POS to be out of range. In this case, WIN1POS or WIN2POS are set to the closest legal value.
		MESIAL	Out-of-range MESIAL argument.
		MSCOUNT	Out-of-range argument.
		MSTO	Out-of-range FROM - TO argument.
		NAVG	Out-of-range NAVG value.
		NENV	Out-of-range NENV value.
		NEXTSTO	Out-of-range value.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
550 (cont)	**%A out of range - limit set**	NEXTFPS	Out-of-range argument.
		NHIST.PT	Out-of-range argument.
		NWAVFRM	Out-of-range argument.
		NREPTRIG	Out-of-range NREPTRIG argument.
		PINDEX	Out-of-range argument.
		PROXIMAL	Out-of-range PROXIMAL argument.
		REFLEVEL	Out-of-range REFLEVEL argument.
		REFSET	Out-of-range argument.
		REPCURVE	Out-of-range NREPCURVE argument.
		REPMEAS	Out-of-range NREPMEAS argument.
		RMZONE	Out-of-range RMZONE argument
		RS232	Out-of-range BAUD, DELAY, or STOPBITS link argument.
		SCANSTOWFM	Out-of-range FROM, RATE, or TO link argument.
		SNRATIO	Out-of-range SNRATIO argument.
		TBMAIN, TBWIN	Out-of-range LENGTH or TIME link argument.
		TEK4692	Out-of-range RGB value.
		TEK4696	Out-of-range color inkjet selection.
		TEXT	X or Y link argument out of range.
		TOPLINE	Out-of-range TOPLINE argument.
		TRMAIN	ALEVEL, ANLEVEL, ANBLEVEL, TIMER1, TIMER2, TIHOLDOFF or TSTIME link argument out of range.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
550 (cont)	**%A out of range – limit set**	TRWIN	ALEVEL, EVHOLDOFF, NLEVEL, TIMER1, TIMER2, or TIHOLDOFF link argument out of range.
		TTAVERAGE	TTAVERAGE argument out of range.
		V1BAR, V2BAR	XCOORD or XDIV link argument out of range.
		WFMPRE	NR.PT, TSTIME, XINCR, XZERO, YMULT, or YZERO link argument out of range.
551	**Insufficient data to satisfy binary block byte count**	Waveform Retrieval and Scaling (Data Transfer)	Binary waveform data sent to GPIB port prematurely terminated (for example, binary block byte count not satisfied when EOI line asserted). The waveform is filled out with NULL points.
552	**Checksum error in binary block transfer**	Waveform Retrieval and Scaling (Data Transfer)	Checksum of received binary waveform data does not match checksum of original binary block. The waveform data is retained, regardless of the outcome of the test. **Note:** *If the binary data was created with a NULL checksum, the checksum test is almost certain to fail. Since the returned data is not discarded, this failure is not important.*
553	**Window trigger source set equal to main trigger source**	Trigger Source Expressions	WTMODE changed from MAIN to TIHOLDOFF or EVHOLDOFF, and window source is incompatible with main source. The window trigger source is set equal to main source.
554	**Autoset – no signal detected**	AUTOSET	AUTOSET initiated with no traces defined and no signal source can be found (for example, no plug-in units are loaded in the plug-in compartments), or the signal being autoset is DC (it has no AC component).

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
555	**Binary curve odd data byte discarded**	Waveform Retrieval and Scaling (Data Transfer)	An odd number of data bytes was sent to the DSA.
556	**No acquisitions active - digitizer remains stopped**	DIGITIZER	Attempted to start digitizer when no traces are defined, or when no defined traces contain "active" components (as opposed to scalar and stored components).
557	**Hardcopy aborted**	COPY	COPY operation aborted.
558	**Nothing to abort**	COPY	COPY ABORT attempted, nothing to abort.
559	**XY PT.FMT not permitted, PT.FMT not changed**	WFMPRE	WFMPRE PT.FMT set to XY. The data format is not changed.
560	**AUTOSET - vertical search failed**	AUTOSET	AUTOSET failure for any reason other than those that generate event code 554.
561	**Base label index greater than 999, waveform not stored**	DIGITIZER	Digitizer reached maximum base label index during repetitive single trigger; digitizer stopped.
562	**AUTOSET - trigger search failed**	AUTOSET	AUTOSET could not find a valid trigger signal.
563	**AUTOSET - horizontal search failed**	AUTOSET	Horizontal autoset algorithm cannot correctly calculate period of selected trace.
564	**AUTOSET - ac signal too large**	AUTOSET	Vertical AUTOSET algorithm detects signal whose AC component is too large for least-sensitive gain setting of plug-in channel.
565	**AUTOSET - dc signal too large**	AUTOSET	Vertical AUTOSET algorithm detects signal whose DC component is larger than offset range of least-sensitive gain setting of plug-in channel.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
566	**Interleave Enabled - Press ENHANCED ACCURACY then compensate probe to use the max Real Time sample rate**	INTERLEAVE	ENHANCED ACCURACY must be in effect and probes must be calibrated to get 1 GSample/sec in DSA 601 or 2 GSample/sec in DSA 602.
567	**Trigger timer2 value modified due to change to timer1**	TRMAIN, TRWIN	Change in trigger timer1 caused change in trigger timer2.
568	**Trigger mode changed to Normal**	CONDACQ REPMEAS REP<meas>	The trigger mode is set to Normal when CONDACQ TYPE is set to SEQUENCE, SINGLE, or DELTA, and when REPMEAS or REP<meas> is started.
569	**Argument out of range. Limit set. Valid smoothing range is: 3 - 999**	TRACE<*ui*>	Trace description contains a SMOOTH function with an out of range argument.
570	**Argument out of range. Limit set. Valid dejitter range is: 0 - 9**	TRACE<*ui*>	Trace description contains a DEJITTER function with an out of range argument.
571	**Interleave Enabled - Compensate probe to use the maximum Real Time sample rate**	INTERLEAVE	When INTERLEAVE is enabled, probe calibration must be run to achieve maximum accuracy.
572	**%d record length changed to %D**	CONDACQ	When CONDACQ TYPE is DELTA, attempted to increase record length which conflicts with current delta description.
573	**FFT record length must be a power of 2**	TBMAIN, TBWIN	The record length of an FFT function must be a power of 2.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
574	**Delta description no longer valid**	COMPARE, DELETE, REMOVE	If you delete an element of a delta description (for example, TRACE or STOWFM), the delta description is invalid.
575	**Argument out of range. Limit set. Valid filter range is: 4ps to 100s**	TRACE<ui>	Trace description contains a filter function with out of range argument.
576	**Warning: Adjustable constants can be used only as standalone parameters in this case**	TRACE<ui>	Trace description contains an adjustable constant combined with other operators as an argument to a function.
577	**Only the first 1024 delta points are displayed**	DIGITIZER	When CONDACQ TYPE is DELTA, and there are more than 1024 points of the test waveform, this error is issued when the digitizer stops.
578	**Stored waveform %N skipped because of incompatible record length**	SCANSTOWFM	A stored waveform was not scanned because its record length was not the same as the first scanned stored waveform.
579	**Cannot access stored waveforms during autostore**	TRACE<ui>	Description contains STO <ui> while REPTRIG is running.
580	**File syntax warning: '%S'%R**	TRACE<ui> DCOPY	Description contains disk stored file in format that has syntax error.
581	**File read warning: insufficient data**	TRACE<ui> DCOPY	Description contains disk stored file in format that has insufficient data.

Execution Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
582	**File read warning: syntax error in binary block**	TRACE<ui> DCOPY	Description contains disk stored file in <bblock> format that has syntax error.
583	**File read warning: extra file data ignored**	TRACE<ui> DCOPY	Description contains disk stored file that has too much data.
584	**Automatically stored waveforms will be created in RAM. Use Disk Copy menu to transfer them to disk**	CONDACQ	When type set is to DELTA or REPTRIG and current stored waveform device is disk.

Internal Warnings

Internal warnings are reported when a problem has been detected. The DSA remains operational, but the problem should be corrected. Internal warnings have event codes from 600 to 699. The SRQMASK for internal warnings is SRQMASK INWARN. The status byte for internal warnings returns **102** (decimal) with RQS set to ON, and **38** (decimal) with RQS set to OFF. All internal warnings are listed on the following pages.

Internal Warnings

Event Code	Description	Commands that Generate Code	Explanation
651	**Input channel %a overload on %B plug-in**	CH <*slot*> <*ui*>	Input signal overloads the low-impedance termination resistor of a plug-in unit. The plug-in unit changes impedance to protect against this condition and returns the event code.
652	**Input channel %a overdrive on %B plug-in**	CH <*slot*> <*ui*>	Input signal of plug-in unit is overdriven in a way that might distort the displayed signal.
653	**RS-232 input parity error**		RS232 PARITY is ON, and a byte is recieved over the RS232 with an incorrect parity bit.
654	**RS-232 input framing error**		RS232 stop bit not detected.
655	**RS-232 input buffer overrun**		RS232 input buffer was overwritten. New data was recieved before the DSA could remove the old data from the input buffer.
656	**Internal table search failed**		
657	**Probable nonvolatile RAM battery failure. Nonvolatile RAM completely reset**		

Internal Warnings (Cont.)

Event Code	Description	Commands that Generate Code	Explanation
658	**Teksecure Status: failed; refer instrument to qualified personnel**		
665	**Teksecure Erase Memory Status: Erased; Instrument ID, on-time, and number of power-ups retained**	TEKSECURE	Indicates all memory was successfully erased.

Programming Examples

The four examples in this section demonstrate how to program typical DSA operations. These examples are based on Examples 2, 4, 6, and 9 given in the *DSA 601A and DSA 602A Tutorial*. The programs for this section are contained in the Learning by Example software, located on a single IBM-formatted $5^1/_4$-inch floppy disk in the disk sleeve in the front of this manual.

To run the examples, you need a basic knowledge of how to use the DSA from the front panel, how to use an IBM PC/XT/AT or compatible computer, and how to write programs in BASIC.

Each example begins with a brief explanation of its purpose and a listing of new DSA commands in the example. A listing of each program line (GPIB version) follows.

Required Hardware and Software

The programs included in this section are written for a Tektronix PEP301 Instrument Controller, or an IBM PC/XT/AT or other PC-compatible computer configured with a GPIB interface and running the GURU II GPIB controller software from Tektronix.

If you are using a different controller, or different software to control the GPIB, or prefer to use the RS-232-C interface, the examples should still prove useful. Since most of the calls to the GURU II software are invocations of IBWRT (send a message over the GPIB) or IBRD (receive a message over the GPIB), it is a simple matter to translate the programs to work with different hardware or software. To translate the examples for use over the RS-232-C interface, for example, most of the calls to IBWRT and IBRD can simply be converted to PRINT and INPUT statements referencing an appropriate device number.

There are nine BASIC programs on the floppy disk.The programs parallel those given in the *DSA 601A and DSA 602A Tutorial*. The programs run under most common BASIC language implementations, including:

- IBM BASIC.COM or BASICA.COM
- IBM Compiled BASIC, versions 1.0 and 2.0
- Microsoft QuickBASIC, version 1.0–4.0 or Compiled 6.0

The examples disk contains two directories, GPIB and RS232. The GPIB directory contains example programs that control the DSA using the GPIB interface. The RS232 directory contains programs that control the DSA using the RS-232-C interface.

Other hardware you will need includes the Pocket Signal Generator, which generates square waves used in the *DSA 601A and DSA 602A Tutorial*, and a GPIB or RS-232-C cable, as appropriate.

Installing the Learning by Example Software

Before running the examples, you should copy them onto your hard disk or to another floppy.

Hard Disk Installation

To install the example programs onto a hard disk:

☐ Step 1: Create a directory on the hard disk to store the example programs (600EXMPL might be a good choice for a name), using the MKDIR command from MS-DOS:

```
MKDIR 600EXMPL
```

☐ Step 2: Make that directory the current directory:

```
CD 600EXMPL
```

☐ Step 3: Insert the examples disk into drive A in your computer. If you want to use the GPIB programs, execute the following command:

```
COPY A:\GPIB\*.*
```

If you want to use the RS-232-C programs instead, execute the following command:

```
COPY A:\RS232\*.*
```

Once installation is complete, put the examples disk into the disk jacket in the manual for safekeeping.

Floppy Disk Installation

To install the examples onto a floppy disk in a dual-floppy-drive system:

☐ Step 1: Insert the examples disk into drive A, and a formatted target disk into drive B. Create a directory on the target floppy disk to store the examples (600EXMPL might be a good choice for a name), using the MKDIR command from MS-DOS:

```
MKDIR 600EXMPL
```

☐ Step 2: Make that directory the current directory:

```
CD 600EXMPL
```

☐ Step 3: If you want to use the GPIB programs, execute the following command:

```
COPY A:\GPIB\*.* B: 600EXMPL
```

If you want to use the RS-232-C programs instead, execute the following command:

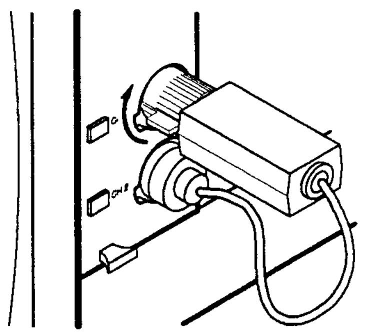
```
COPY A:\RS232\*.* B: 600EXMPL
```

☐ Step 4: If you intend to use the target floppy disk as a start-up disk (it must be formatted with the /S option in order to do this), copy the following additional files from your original start-up disk onto the target disk:AUTOEXEC.BAT, CONFIG.SYS, and the name of your BASIC program file, for example, BASIC.COM, or BASICA.COM.

Once installation is complete, put the examples disk into the disk jacket in the manual for safekeeping.

Attaching the Pocket Signal Generator

To run the examples, you must connect the pocket signal generator to channels 1 and 2 of the left plug-in amplifier. Connect the large end of the pocket signal generator to channel 1; connect the other end to channel 2. See the following illustration.



How the Pocket Signal Generator Should Be Connected

Running the Learning by Example Software

You can run the programs in either of two ways: from the MENU program or individually.

To run the programs from the menu:

☐ Step 1: Check that the current directory is the directory where the MENU program resides, for example, 600EXMPL. Make sure the location of your BASIC application is in the directory search path.

☐ Step 2: Enter the name of your BASIC application followed by the program name MENU, for example:

```
BASIC MENU
```

or BASICA MENU, or whatever it happens to be.

The MENU program displays the following list of programs for you to choose from:

1) Displaying a Single Waveform
2) Managing Multiple Waveforms
3) Defining Complex Waveforms
4) Using Signal Processing
5) Taking Automated Measurements
6) Comparing to a Reference Measurement
7) Taking Delay Measurements Using Cursors
8) Comparing Waveforms to Stored Waveforms
9) Using the Disk Drive

☐ Step 3: Type the number of the program you want to run, and press <Enter>.

To run an individual program:

☐ Step 1: Check that the current directory is the directory where the program resides, for example, 600EXMPL\GPIB.

☐ Step 2: Type the name of your BASIC application and the name of the program that you want to run (the DIR command displays the available program names), followed by <Enter>, for example,

```
BASIC SINGLE.BAS
```

or BASICA SINGLE.BAS.

Exiting the Learning by Example Software

When a program completes, you can type:

- <Enter> (which returns you to the MENU program).
- **Q** (which exits the program and leaves you in BASIC).
- **S** (which exits the program and returns control to MS-DOS.

To exit a program without completing it, press CTRL-C. This will leave you in BASIC (most likely with a disabled front panel; see below). After re-enabling front-panel operation, execute a SYSTEM command to return to MS-DOS.

If a program is terminated prematurely, front panel operation will most likely be disabled. To re-enable front panel operation, do one of the following:

- Cycle power on the DSA.
- If your computer is communicating with the DSA over GPIB, execute the following BASIC commands:

  ```
  WRT$="FPANEL ON"
  CALL IBWRT(TEKDEV1%,WRT$)
  ```

- If your computer is communicating with the DSA over RS-232-C, execute the following command:

  ```
  PRINT #1,"FPANEL ON"
  ```

Setting GPIB Device 0 to "TEKDEV1"

In using the examples with a PEP controller or GURU II software, the name of GPIB device 0 must be set to "TEKDEV1." Use the IBCONF program that came with your PEP controller or GURU II software to check for this name, and change GPIB device 0 to be "TEKDEV1" if necessary.

Managing Multiple Waveforms

Example 2 in the *DSA 601A and DSA 602A Tutorial* demonstrates how to display, label, and control multiple waveforms, how to divide the display into two graticules, and how to return the display to a single graticule.

Commands and links introduced in this example include:

- CH<*slot*><*ui*> IMPEDANCE sets the impedance of a specified channel.
- CH<*slot*><*ui*> OFFSET sets the vertical offset of a specified channel (moves a waveform up or down).
- CH<*slot*><*ui*> SENSITIVITY sets the "sensitivity," or vertical volts/division, of a specified channel. Use this command to adjust the size of a waveform on the display.
- DEBUG GPIB, when set to ON, causes commands input to the GPIB port to be displayed at the top of the screen.
- FPANEL, when set to OFF, disables front-panel operation; when set to ON, enables it again.
- INIT returns the DSA to default settings.
- LONGFORM when set to ON, causes query responses to contain full header and link spellings; when set to OFF, query responses are in abbreviated form.
- TBMAIN TIME sets the horizontal scale (time/division).
- TRACE<*ui*> DESCRIPTION defines the source description of a specified waveform.
- DISPLAY GRATICULE selects single or dual display graticules.
- LABABS PCTG sets the horizontal position of a label on the selected waveform as a percentage of the waveform record.
- LABEL TRACE<*ui*> defines a label for a specified waveform.

- REMOVE TRACE<*ui*> removes a specified waveform from the display.
- SELECT TRACE<*ui*> selects a specified waveform.
- TRACE<*ui*> GRLOCATION positions the selected waveform to the upper or lower graticule.

Example 2 Program Listing

```
100  CLS
110  PRINT "DSA 600 Series Digitizing Signal Analyzer"
120  PRINT "Example 2: Managing Multiple Waveforms"
130  PRINT "(GPIB Version)"
140  PRINT
150  REM
160  REM decl.bas
170  REM
180  REM GURU initialization code; declarations
190  REM
200  CLEAR ,58900!              'IBM BASICA Declarations; =BYTES
     FREE -size(bib.m);
210  IBINIT1 = 58900!           'a smaller-than-calculated # is OK
220  IBINIT2 = IBINIT1 + 3      'these lines (thru CALL statements
     below)
230  BLOAD "bib.m",IBINIT1 'MUST be included in your program
240  CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IBSIC,IBLOC,IBPPC,
     IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IBIST,
     IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
250  CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWRTA,
     IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,
     IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IBERR%,
     IBCNT%)
260  BDNAME$ = "TEKDEV1"
270  CALL IBFIND (BDNAME$,TEKDEV1%)
280  IF TEKDEV1% < 0 THEN PRINT "IBFIND ERROR":END
290  INPUT "Press ENTER to set up the DSA.",A$
300  WRT$= "init;longform on;fpanel off;debug gpib:on"
310  REM
320  REM Line 300 initializes the DSA, specifies the long form of the
     query responses, turns the front panel off, and turns the front panel
     GPIB ASCII display on.
330  REM
```

```
340  CALL IBWRT(TEKDEV1%,WRT$)
350  INPUT "Press ENTER to display a waveform from CH 1 of the left
     plug-in amplifier.",A$
360  WRT$= "trace1 description:'L1' "
370  REM
380  REM Line 360 defines the source for trace one: left plug-in channel
     one.
390  REM
400  CALL IBWRT(TEKDEV1%,WRT$)
410  INPUT "Press ENTER to position the waveform.",A$
420  WRT$= "tbmain time:10e-6;mainpos -8e-6;chl1 imped-
     ance:50"
430  REM
440  REM Line 420 sets the main time base to ten microseconds per
     division, the horizontal position of the main waveform to minus
     eight microseconds (the first point on the graticule is minus eight
     microseconds from the trigger point), and channel one of the left
     plug-in module to an impedance of fifty ohms.
450  REM
460  WRT$= "chl1 sensitivity:200e-3;chl1 offset:400e-3;trmain
     anlevel:500e-3,vol"
470  REM
480  REM Line 460 sets channel one of the left plug-in module to have
     a sensitivity of 200 millivolts per division and an offset of 400
     millivolts. The main trigger is set to 500 millivolts.
490  REM
500  CALL IBWRT(TEKDEV1%,WRT$)
510  INPUT "Press ENTER to display a waveform from CH 2 of the left
     plug-in amplifier.",A$
520  WRT$= "trace2 description:'L2' "
530  REM
540  REM Line 520 defines the source description of the waveform (left
     plug-in module, channel two).
550  REM
560  CALL IBWRT(TEKDEV1%,WRT$)
```

```
570  INPUT "Press ENTER to position the waveform.",A$
580  WRT$ = "chl2 impedance:50;chl2 sensitivity:200e-3;chl2
     offset:600e-3"
590  REM
600  REM Line 580 sets channel two of the left plug-in module to have
     an impedance of fifty ohms, a sensitivity of 200 millivolts per
     division, and an offset of 600 millivolts.
610  REM
620  CALL IBWRT(TEKDEV1%,WRT$)
630  INPUT "Press ENTER to select trace 1.",A$
640  WRT$ = "sel trace1"
650  REM
660  REM Line 640 selects trace (waveform) one.
670  REM
680  CALL IBWRT(TEKDEV1%,WRT$)
690  INPUT "Press ENTER to label trace 1 and position its label.",A$
700  WRT$ = "label display:on;label trace1:'Trace 1';lababs
     pctg:10"
710  REM
720  REM Line 700 turns the waveform label function on, labels trace
     one as Trace 1, and sets the horizontal position of the label at
     10 percent of the waveform record (if the entire record is dis-
     played, the label will be indented to the right approximately 10
     percent of the screen's width).
730  REM
740  CALL IBWRT(TEKDEV1%,WRT$)
750  INPUT "Press ENTER to select trace 2.",A$
760  WRT$ = "sel tra2"
770  REM
780  REM Line 760 selects trace two.
790  REM
800  CALL IBWRT(TEKDEV1%,WRT$)
810  INPUT "Press ENTER to label trace 2 and position its label.",A$
820  WRT$ = "label tra2:'Trace 2';lababs pctg:25"
```

```
830  REM
840  REM Line 820 labels trace two as Trace 2 and sets the horizontal
     position of the label to 25 percent of the waveform record.
850  REM
860  CALL IBWRT(TEKDEV1%,WRT$)
870  INPUT "Press ENTER to display two graticules.",A$
880  WRT$ = "disp gra:dua"
890  REM
900  REM Line 880 displays two graticules.
910  REM
920  CALL IBWRT(TEKDEV1%,WRT$)
930  INPUT "Press ENTER to move trace 2 to the upper graticule.",A$
940  WRT$ = "tra2 grl:upper"
950  REM
960  REM Line 940 moves trace two to the upper graticule.
970  REM
980  CALL IBWRT(TEKDEV1%,WRT$)
990  INPUT "Press ENTER to move trace 1 to the lower graticule.",A$
1000 WRT$ = "trace1 grl:lower"
1010 REM
1020 REM Line 1000 moves trace one to the lower graticule.
1030 REM
1040 CALL IBWRT(TEKDEV1%,WRT$)
1050 INPUT "Press ENTER to return the display to a single
     graticule.",A$
1060 WRT$ = "disp gra:sin"
1070 REM
1080 REM Line 1060 displays one graticule.
1090 REM
1100 CALL IBWRT(TEKDEV1%,WRT$)
1110 INPUT "Press ENTER to remove trace 2.",A$
1120 WRT$ = "rem tra2"
```

```
1130 REM
1140 REM Line 1120 removes trace two.
1150 REM
1160 CALL IBWRT(TEKDEV1%,WRT$)
1170 INPUT "Press ENTER to re-enable the front panel.",A$
1180 WRT$ = "fpanel on"
1190 REM
1200 REM Line 1180 turns the front panel on.
1210 REM
1220 CALL IBWRT(TEKDEV1%,WRT$)
1230 PRINT "End of example 2."
1240 PRINT "Press ENTER to return to the Examples menu,"
1250 PRINT "press 'Q' to quit the program without exiting BASIC,"
1260 INPUT "or press 'S' to quit the program and exit BASIC.",A$
1270 IF LEFT$(A$,1) = "Q" OR LEFT$(A$,1) = "q" THEN END
1280 IF LEFT$(A$,1) = "S" OR LEFT$(A$,1) = "s" THEN SYSTEM
     ELSE LOAD "menu.bas",R
```

Using Signal Processing

Example 4 in the *DSA 601A and DSA 602A Tutorial* demonstrates how to use the signal-processing features of the DSA to provide more information about a waveform than is available from the "normal" display.

Commands and links introduced in this example include:

- CONDACQ TYPE sets the acquisition type, such as averaging, continuous, enveloping, repetitive trigger, etc.
- NAVG sets the number of waveform samples to be used for averaging.
- TBMAIN LENGTH sets the record length of the main time base in points per waveform.
- TRACE<*ui*> ACCUMULATE sets point accumulate mode to INFINITE or OFF for the specified waveform.
- TRMAIN COUPLING sets the trigger coupling.

Example 4 Program Listing

```
100  CLS
110  PRINT "DSA 600 Series Digitizing Signal Analyzer"
120  PRINT "Example 4: Using Signal Processing"
130  PRINT "(GPIB version)"
140  PRINT
150  REM
160  REM decl.bas
170  REM
180  REM GURU initialization code; declarations
190  REM
200  CLEAR ,58900!              'IBM BASICA Declarations; = BYTES
     FREE -size(bib.m)
210  IBINIT1 = 58900!           'a smaller than calculated # is OK
220  IBINIT2 = IBINIT1 + 3      'these lines (thru CALL statements
     below)
230  BLOAD "bib.m",IBINIT1 'MUST be included in your program
240  CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IBSIC,IBLOC,
     IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IBIST,
     IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
250  CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWRTA,
     IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,
     IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IB-
     ERR%,IBCNT%)
260  BDNAME$ = "TEKDEV1"
270  CALL IBFIND (BDNAME$,TEKDEV1%)
280  IF TEKDEV1% < 0 THEN PRINT "IBFIND ERROR":PRINT
290  INPUT "Press ENTER to set up the DSA.",A$
300  WRT$= "init;longform on;fpanel off;debug gpib:on"
310  REM
320  REM Line 300 initializes the DSA, specifies the long form of the
     query responses, turns the front panel off, and turns the front-
     panel GPIB ASCII display on.
330  REM
```

```
340 CALL IBWRT(TEKDEV1%,WRT$)
350 INPUT "Press ENTER to display a waveform from CH 2 of the left
    plug-in amplifier.",A$
360 WRT$="trace1 description:'L2'"
370 REM
380 REM Line 360 defines the source description of the waveform (left
    plug-in module, channel two).
390 REM
400 CALL IBWRT(TEKDEV1%,WRT$)
410 INPUT "Press ENTER to set up the waveform.",A$
420 WRT$="tbmain time:10e-6;mainpos -1.2e-6;chl2 imped-
    ance:1e6;chl2 sens:1;chl2 offset:4;trmain anlevel:2.7,vol"
430 REM
440 REM Line 420 sets the main time base to ten microseconds per
    division, the horizontal position of the main waveform to minus
    1.2 microseconds, and channel two of the left plug-in module to
    an impedance of 1 M ohm, a sensitivity of 1 volt, an offset of 4
    volts, and a main trigger level of 2.7 volts.
450 REM
460 CALL IBWRT(TEKDEV1%,WRT$)
470 INPUT "Press ENTER to re-enable the front panel.",A$
480 WRT$="fpanel on"
490 REM
500 REM Line 480 turns the front panel on.
510 REM
520 CALL IBWRT(TEKDEV1%,WRT$)
530 PRINT "The front panel is now re-enabled."
540 PRINT "Set the Vertical Offset resolution to FINE"
550 PRINT "(touch the vertical arrows selector,"
560 PRINT "touch the Vertical Offset selector, then press FINE)"
570 PRINT "and adjust the vertical offset until you find a point"
580 PRINT "near the vertical center of the waveform"
590 PRINT "where triggering is unstable"
```

```
600  PRINT "(i.e., where the waveform moves around on the screen)."
610  INPUT "Then press ENTER to continue the program.",A$
620  WRT$ = "fpanel off"
630  REM
640  REM Line 620 turns the front panel off.
650  REM
660  CALL IBWRT(TEKDEV1%,WRT$)
670  INPUT "Press ENTER to set the main trigger's coupling to DC
     High-Frequency Reject.",A$
680  WRT$ = "trmain coupling:dchf"
690  REM
700  REM Line 680 sets main triggering coupling to reject DC high
     frequency.
710  REM
720  CALL IBWRT(TEKDEV1%,WRT$)
730  INPUT "Press ENTER to set the main trigger's coupling back to
     DC.",A$
740  WRT$ = "trmain coupling:dc"
750  REM
760  REM Line 740 sets main triggering coupling to DC.
770  REM
780  CALL IBWRT(TEKDEV1%,WRT$)
790  INPUT "Press ENTER to set main size to 50ns/div, main position
     to -105ns.",A$
800  WRT$ = "tbmain time:50e-9;mainpos -105e-9"
810  REM
820  REM Line 800 sets the main time base to 50 nanoseconds per
     division and the horizontal position of the main waveform to minus
     105 nanoseconds.
830  REM
840  CALL IBWRT(TEKDEV1%,WRT$)
850  INPUT "Press ENTER to turn infinite persistence mode on.",A$
860  WRT$ = "trace1 accumulate:infp"
```

```
870  REM
880  REM Line 860 turns on the infinite persistence mode for trace 1.
890  REM
900  CALL IBWRT(TEKDEV1%,WRT$)
910  INPUT "Press ENTER to turn infinite persistence mode off.",A$
920  WRT$ = "trace1 accumulate:off"
930  REM
940  REM Line 920 turn off the infinite persistence mode for trace 1.
950  REM
960  CALL IBWRT(TEKDEV1%,WRT$)
970  INPUT "Press ENTER to set the trigger level voltage one divisision
     higher.",A$
980  REM
990  REM get the original sensitivity value.
1000 REM
1010 WRT$ = "chl2? sensitivity"
1020 REM
1030 REM Line 1010 queries the sensitivity of channel two of the left
     plug-in module.
1040 REM
1050 CALL IBWRT(TEKDEV1%,WRT$)
1060 RD$ = SPACE$(63)
1070 CALL IBRD(TEKDEV1%,RD$)
1080 SENS$ = RD$
1090 REM
1100 REM process the returned string.
1110 REM
1120 WHILE RIGHT$(SENS$,1) = " "
1130 SENS$ = LEFT$(SENS$,LEN(SENS$)-1)
1140 WEND
1150 REM
1160 REM find the position of the ":" character.
1170 REM
```

```
1180 COLPOS=INSTR(1,SENS$,":")
1190 REM
1200 REM extract the numeric value.
1210 REM
1220 SENS=VAL(MID$(SENS$,COLPOS+1,LEN(SENS$)))
1230 REM
1240 REM get the original main trigger level.
1250 REM
1260 WRT$= "trmain? anlevel"
1270 REM
1280 REM Line 1260 queries the main trigger's current level.
1290 REM
1300 CALL IBWRT(TEKDEV1%,WRT$)
1310 RD$=SPACE$(63)
1320 CALL IBRD(TEKDEV1%,RD$)
1330 TRM$=RD$
1340 REM
1350 REM process the returned string.
1360 REM
1370 WHILE RIGHT$(TRM$,1)= " "
1380 TRM$=LEFT$(TRM$,LEN(TRM$)-1)
1390 WEND
1400 REM
1410 REM find the position of the "," character.
1420 REM
1430 COMPOS=INSTR(1,TRM$,",")
1440 REM
1450 REM extract the units string.
1460 REM
1470 UNIT$=MID$(TRM$,COMPOS+1,LEN(TRM$))
1480 TRM$=MID$(TRM$,1,COMPOS-1)
1490 REM
```

```
1500 REM find the position of the ":" character.
1510 REM
1520 COLPOS=INSTR(1,TRM$,":")
1530 REM
1540 REM extract the numeric value.
1550 REM
1560 TRM=VAL(MID$(TRM$,COLPOS+1,LEN(TRM$)))
1570 IF UNIT$< > "VOLTS" THEN TRM=TRM*SENS
1580 REM
1590 REM calculate new trigger level.
1600 REM
1610 TRM=TRM+SENS
1620 REM
1630 REM put a new command string together and send it.
1640 REM
1650 TRM$=STR$(TRM)
1660 WRT$= "trmain anlevel:"+TRM$+",volts"
1670 REM
1680 REM Line 1660 sets the main trigger level to the value of TRM$ (a
     BASIC variable) volts.
1690 REM
1700 CALL IBWRT(TEKDEV1%,WRT$)
1710 INPUT "Press ENTER to display the average value of CH 2.",A$
1720 WRT$= "tbmain time:10e-6;trace1 description:'avg(L2)' "
1730 REM
1740 REM Line 1720 sets the main time base to 6 microseconds per
     division and describes trace one as the average value of channel
     two of the left plug-in module.
1750 REM
1760 CALL IBWRT(TEKDEV1%,WRT$)
1770 INPUT "Press ENTER to set the number of acquisitions to be
     averaged to 128.",A$
1780 WRT$= "navg 128"
```

```
1790 REM
1800 REM Line 1780 sets the number of waveform acquisitions to be
     averaged to 128.
1810 REM
1820 CALL IBWRT(TEKDEV1%,WRT$)
1830 INPUT "Press ENTER to stop acquisition after 128 acquisi-
     tions.",A$
1840 WRT$= "condacq type:avg"
1850 REM
1860 REM Line 1840 stops acquisition on the condition that the number
     of waveform acquisitions (specified by NAVG) has been met.
1870 REM
1880 CALL IBWRT(TEKDEV1%,WRT$)
1890 INPUT "Press ENTER to re-start continuous acquisition.",A$
1900 WRT$= "condacq type:continuous"
1910 REM
1920 REM Line 1900 starts continuous acquisition.
1930 REM
1940 CALL IBWRT(TEKDEV1%,WRT$)
1950 INPUT "Press ENTER to display the original waveform from CH
     2.",A$
1960 WRT$= "trace1 description:'L2' "
1970 REM
1980 REM Line 1960 defines the source description of the waveform (left
     plug-in module, channel two).
1990 REM
2000 CALL IBWRT(TEKDEV1%,WRT$)
2010 INPUT "Press ENTER to set the main record length to 512.",A$
2020 REM
2030 REM save the original record length.
2040 REM
2050 WRT$= "tbmain? length"
2060 REM
```

```
2070 REM Line 2050 queries the length of the main time base.
2080 REM
2090 CALL IBWRT(TEKDEV1%,WRT$)
2100 RD$=SPACE$(64)
2110 CALL IBRD(TEKDEV1%,RD$)
2120 TBMLEN$=RD$
2130 WHILE RIGHT$(TBMLEN$,1)=" "
2140 TBMLEN$=LEFT$(TBMLEN$,LEN(TBMLEN$)-1)
2150 WEND
2160 WRT$="tbmain length:512"
2170 REM
2180 REM Line 2160 sets the length of the main time base to 512 points
     per waveform.
2190 REM
2200 CALL IBWRT(TEKDEV1%,WRT$)
2210 INPUT "Press ENTER to set the main record length to its upper
     limit.",A$
2220 WRT$="tbmain length:32768"
2230 REM
2240 REM Line 2220 sets the length of the main time base to 32768
     points per waveform.
2250 REM
2260 CALL IBWRT(TEKDEV1%,WRT$)
2270 INPUT "Press ENTER to set the main record length back to its
     original value.",A$
2280 WRT$=TBMLEN$
2290 CALL IBWRT(TEKDEV1%,WRT$)
2300 INPUT "Press ENTER to re-enable the front panel.",A$
2310 WRT$="fpanel on"
2320 REM
2330 REM Line 2310 turns the front panel on.
2340 REM
2350 CALL IBWRT(TEKDEV1%,WRT$)
```

```
2360 PRINT "End of example 4."
2370 PRINT "Press ENTER to return to the Examples menu,"
2380 PRINT "press 'Q' to quit the program without exiting BASIC,"
2390 INPUT "or press 'S' to quit the program and exit BASIC.",A$
2400 IF LEFT$(A$,1) = "Q" OR LEFT$(A$,1) = "q" THEN END
2410 IF LEFT$(A$,1) = "S" OR LEFT$(A$,1) = "s" THEN SYSTEM
     ELSE LOAD "menu.bas",R
```

Comparing to a Reference Measurement

Example 6 in the *DSA 601A and DSA 602A Tutorial* demonstrates how to set up a reference waveform and compare it to other waveforms.

Commands and links introduced in this example include:

- COMPARE, when set to ON, returns the difference between the measurement (made on the selected waveform) and the measurement′s reference value (made on a reference waveform); when set to OFF, measurements return the value of the measurement.
- MEAS? returns the current values of the measurements in the current measurements list.
- MSLIST selects the measurements to be included in the current measurements list.
- MSYS ON, when set to ON, displays the Measurements Major Menu at the bottom of the front-panel display.
- REFSET CURRENT executes a specified measurement and stores the result as a reference value.

Example 6 Program Listing

```
100  CLS
110  PRINT "DSA 600 Series Digitizing Signal Analyzer"
120  PRINT "Example 6: Comparing to a Reference Measurement"
130  PRINT "(GPIB version)"
140  PRINT
150  REM
160  REM decl.bas
170  REM
180  REM GURU initialization code; declarations
190  REM
200  CLEAR ,58900!              'IBM BASICA Declarations; =BYTES
     FREE -size(bib.m);
210  IBINIT1 = 58900!           'a smaller-than-calculated # is OK
220  IBINIT2 = IBINIT1 + 3      'these lines (thru CALL statements
     below)
230  BLOAD "bib.m",IBINIT1 'MUST be included in your program
240  CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IBSIC,IBLOC,
     IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IBIST,
     IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
250  CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWRTA,
     IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,
     IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IB-
     ERR%,IBCNT%)
260  BDNAME$ = "TEKDEV1"
270  CALL IBFIND (BDNAME$,TEKDEV1%)
280  IF TEKDEV1% < 0 THEN PRINT "IBFIND ERROR":END
290  INPUT "Press ENTER to set up the DSA.",A$
300  WRT$= "init;longform on;fpanel off;debug gpib:on"
310  REM
320  REM Line 300 initializes the DSA, specifies the long form of the
     query responses, turns the front panel off, and turns the
     front-panel GPIB ASCII display on.
330  REM
```

```
340  CALL IBWRT(TEKDEV1%,WRT$)
350  INPUT "Press ENTER to display a waveform from CH 1 of the left
     plug-in amplifier.",A$
360  WRT$= "trace1 description:'L1' "
370  REM
380  REM Line 360 defines the source description of the waveform (left
     plug-in module, channel one).
390  REM
400  CALL IBWRT(TEKDEV1%,WRT$)
410  INPUT "Press ENTER to set up the waveform.",A$
420  WRT$= "tbmain time:10e-6;mainpos -1.2e-6;chl1 imped-
     ance:50,sens:0.2,offset:0.8;trmain anlevel:0.4,vol"
430  REM
440  REM Line 420 sets the main time base to ten microseconds per
     division, the horizontal position of the main waveform to minus
     1.2 microseconds, and channel one of the left plug-in module to
     an impedance of fifty ohms, a sensitivity of 0.2 volts, and an offset
     of 0.8 volts. The main trigger level is set to 0.4 volts.
450  CALL IBWRT(TEKDEV1%,WRT$)
460  INPUT "Press ENTER to display a waveform from CH 2 of the left
     plug-in amplifier.",A$
470  WRT$= "trace2 description:'L2' "
480  REM
490  REM Line 470 defines the source description of the waveform (left
     plug-in module, channel two).
500  REM
510  CALL IBWRT(TEKDEV1%,WRT$)
520  INPUT "Press ENTER to set up the waveform.",A$
530  WRT$= "chl2 impedance:50,sens:0.2,offset:0.2"
540  REM
550  REM Line 530 sets channel two of the left plug-in module to an
     impedance of fifty ohms, a sensitivity of 0.2 volts, and an offset of
     0.2 volts.
560  REM
570  CALL IBWRT(TEKDEV1%,WRT$)
```

```
580  INPUT "Press ENTER to add Peak-to-Peak to the measurements
     list.",A$
590  WRT$ = "mslist?"
600  REM
610  REM Line 590 queries the current measurement list.
620  REM
630  CALL IBWRT(TEKDEV1%,WRT$)
640  RD$ = SPACE$(64)
650  CALL IBRD(TEKDEV1%,RD$)
660  MSLIST$ = RD$
670  WHILE RIGHT$(MSLIST$,1) = " "
680  MSLIST$ = LEFT$(MSLIST$,LEN(MSLIST$)-1)
690  WEND
700  PRINT "The current measurements list is:"
710  PRINT MSLIST$
720  WRT$ = "mslist pp"
730  REM
740  REM Line 720 sets the measurement list to peak to peak.
750  REM
760  CALL IBWRT(TEKDEV1%,WRT$)
770  WRT$ = "mslist?"
780  REM
790  REM Line 770 queries the current measurement list.
800  REM
810  CALL IBWRT(TEKDEV1%,WRT$)
820  RD$ = SPACE$(64)
830  CALL IBRD(TEKDEV1%,RD$)
840  MSLIST$ = RD$
850  WHILE RIGHT$(MSLIST$,1) = " "
860  MSLIST$ = LEFT$(MSLIST$,LEN(MSLIST$)-1)
870  WEND
880  PRINT "The new measurements list is:"
```

```
890  PRINT MSLIST$
900  INPUT "Press ENTER to display the measurements menu.",A$
910  WRT$ = "msys on"
920  REM
930  REM Line 910 turns the measurement system on.
940  REM
950  CALL IBWRT(TEKDEV1%,WRT$)
960  INPUT "Press ENTER to take a measurement.",A$
970  GOSUB 1360
980  INPUT "Press ENTER to use Peak-to-Peak as a reference mea-
     surement.",A$
990  WRT$ = "refset current:pp"
1000 REM
1010 REM Line 990 sets the current value of the peak-to-peak measure-
     ment as a reference value.
1020 REM
1030 CALL IBWRT(TEKDEV1%,WRT$)
1040 INPUT "Press ENTER to turn COMPARE on.",A$
1050 WRT$ = "compare on"
1060 REM
1070 REM Line 1050 turns the compare function on, so any deviation
     from the set reference values will be reported when the measure-
     ment is queried.
1080 REM
1090 CALL IBWRT(TEKDEV1%,WRT$)
1100 INPUT "Press ENTER to take a comparison measurement.",A$
1110 WRT$ = "select trace1"
1120 REM
1130 REM Line 1110 selects trace one. Line 1160 is a subroutine call to
     line 1360, line 1380 does the actual measurement query.
1140 REM
1150 CALL IBWRT(TEKDEV1%,WRT$)
1160 GOSUB 1360
```

```
1170 INPUT "Press ENTER to turn COMPARE off.",A$
1180 WRT$= "compare off"
1190 REM
1200 REM Line 1180 turns the compare function off.
1210 REM
1220 CALL IBWRT(TEKDEV1%,WRT$)
1230 INPUT "Press ENTER to re-enable the front panel.",A$
1240 WRT$= "fpanel on"
1250 REM
1260 REM Line 1240 turns the front panel on.
1270 REM
1280 CALL IBWRT(TEKDEV1%,WRT$)
1290 PRINT "End of example 6."
1300 PRINT "Press ENTER to return to the Examples menu,"
1310 PRINT "press 'Q' to quit the program without exiting BASIC,"
1320 INPUT "or press 'S' to quit the program and exit BASIC.",A$
1330 IF LEFT$(A$,1) = "Q" OR LEFT$(A$,1) = "q" THEN END
1340 IF LEFT$(A$,1) = "S" OR LEFT$(A$,1) = "s" THEN SYSTEM
     ELSE LOAD "menu.bas",R
1350 REM
1360 REM subroutine to take measurements
1370 REM
1380 WRT$= "meas?"
1390 REM
1400 REM Line 1380 queries all of the measurements in the measure-
     ment list.
1410 REM
1420 CALL IBWRT(TEKDEV1%,WRT$)
1430 RD$=SPACE$(64)
1440 CALL IBRD(TEKDEV1%,RD$)
1450 MEAS$=RD$
1460 WHILE RIGHT$(MEAS$,1) = " "
```

```
1470 MEAS$=LEFT$(MEAS$,LEN(MEAS$)-1)
1480 WEND
1490 PRINT "The measurement results are:"
1500 PRINT MEAS$
1510 RETURN
```

Using the Disk Drive

Example 9 in the *DSA 601A and DSA 602A Tutorial* demonstrates how to store and recall waveforms on the floppy disk.

The commands and links introduced in this example are:

- CD changes the current working directory.
- DCOPY copies a file from one location to another.
- DELETE deletes a file on the disk.
- DIR? lists the files in the current working directory.
- FORMAT formats a floppy disk.
- MKDIR creates a directory on the disk.
- NREPTRIG sets the number of repetitive trigger acquisitions.
- RENAME renames a file on the disk.
- RMDIR removes a directory.
- SETDEV sets the storage device to RAM or disk.
- STOLIST lists the stored waveforms in RAM or disk.
- STORE TRA<*ui*> stores a specified trace.

Example 9 Program Listing

```
100 CLS
110  PRINT "DSA 600 Series Digitizing Signal Analyzer"
120  PRINT "Example 9: Using the Disk Drive"
130  PRINT "(GPIB Version)"
140  PRINT
150  REM
160  REM decl.bas
170  REM
180  REM GURU initialization code;declarations
190  REM
200  CLEAR ,58900!        ´IBM BASICA Declarations; = BYTES FREE
     -size(bib.m);
210  IBINIT1 = 58900!     ´a smaller-than-calculated # is OK
220  IBINIT2 = IBINIT1 + 3  ´these lines (thru CALL statements below)
230  BLOAD "bib.m",IBINIT1  ´MUST be included in your program
240  CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IB-
     SIC,IBLOC,IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBS
     AD,IBIST,IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
250  CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWR-
     TA,IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,IB
     XTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IB-
     ERR%,IBCNT%)
260  BDNAME$ = "TEKDEV1"
270  CALL IBFIND (BDNAME$,TEKDEV1%)
280  IF TEKDEV1% < 0 THEN PRINT "IBFIND ERROR":END
290  INPUT "Press ENTER to set up the DSA.",A$
300  WRT$ = "init;longform on;fpanel off;debug gpib:on"
310  REM
320  REM line 300 initializes the DSA, specifies the long form of the
     query responses, turns off the front panel, and turns on the GPIB
     ASCII display.
330  REM
340  CALL IBWRT(TEKDEV1%,WRT$)
```

```
350  INPUT "Press ENTER to display a waveform from CH 1 of the left
     plug-in amplifier.",A$
360  WRT$ = "trace1 description:'L1' "
370  REM
380  REM line 360 defines the source description of the waveform (Left
     plug-in module, channel 1).
390  REM
400  CALL IBWRT(TEKDEV1%,WRT$)
410  INPUT "Press ENTER to position the waveform.",A$
420  WRT$ = "chl1 imp:50;autoset start"
430  REM
440  REM line 420 sets the impedance of channel 1 to 50 ohms, and
     autosets the DSA.
450  REM
460  CALL IBWRT(TEKDEV1%,WRT$)
470  PRINT "Insert a blank disk in the DSA's disk drive. Press ENTER
     to format the disk."
480  INPUT "NOTE: Formatting erases all information on the disk.",A$
490  WRT$ = "format 'A:' "
500  REM
510  REM line 490 formats the disk.
520  REM
530  CALL IBWRT(TEKDEV1%,WRT$)
540  PRINT "Press ENTER to specify the disk as the stored waveform
     device"
550  INPUT "and make a directory called 'TEST1'. Wait until format is
     finished.",A$
560  WRT$ = "setdev sto:disk;mkdir 'A:\TEST1' "
570  REM
580  REM line 560 sets the storage device (for stored waveforms) to
     disk and makes a directory on the disk called TEST1.
590  REM
600  CALL IBWRT(TEKDEV1%,WRT$)
```

```
610  INPUT "Press ENTER to make a directory called ´TEST2´.",A$
620  WRT$= "mkdir ´A:\TEST2´ "
630  REM
640  REM line 620 makes a directory on the disk called TEST2.
650  REM
660  CALL IBWRT(TEKDEV1%,WRT$)
670  INPUT "Press ENTER to see a list of directories.",A$
680  WRT$= "cd ´A:\´;dir?"
690  REM
700  REM Line 680 change directories and lists its contents.
710  REM
720  CALL IBWRT(TEKDEV1%,WRT$)
730  RD$=SPACE$(128)
740  CALL IBRD(TEKDEV1%,RD$)
750  DIR$=RD$
760  WHILE RIGHT$(DIR$,1)= " "
770  DIR$=LEFT$(DIR$,LEN(DIR$)-1)
780  WEND
790  PRINT "Current directories on the disk:"
800  PRINT DIR$
810  INPUT "Press ENTER to store the displayed waveform to disk as
     ´STO001.WFA´ ",A$
820  WRT$= "store trace1:´A:\TEST1\STO001.WFA´ "
830  REM
840  REM line 820 stores a waveform in TEST1 as STO001.WFA.
850  REM
860  CALL IBWRT(TEKDEV1%,WRT$)
870  INPUT "Press ENTER to remove the displayed waveform.",A$
880  WRT$= "remove trace1"
890  REM
900  REM line 880 removes the waveform from the display.
910  REM
```

```
920  CALL IBWRT(TEKDEV1%,WRT$)
930  INPUT "Press ENTER to display the disk-stored waveform.",A$
940  WRT$= "trace1 description:´A:\TEST1\STO001.WFA´ "
950  REM
960  REM line 940 defines trace 1 as the stored waveform
     STO001.WFA.
970  REM
980  CALL IBWRT(TEKDEV1%,WRT$)
990  INPUT "Press ENTER to display channel two for comparison with
     the stored waveform.",A$
1000 WRT$= "trace2 description:´L2´ "
1010 REM
1020 REM Line 1000 defines trace 2 as amplifier channel two.
1030 REM
1040 CALL IBWRT(TEKDEV1%,WRT$)
1050 INPUT "Press ENTER to change directories to A:\TEST1 and
     display its contents",A$
1060 WRT$= "cd ´a:\test1´;dir?"
1070 REM
1080 REM Line 1060 changes directories to TEST1 and lists its con-
     tents.
1090 REM
1100 CALL IBWRT(TEKDEV1%,WRT$)
1110 RD$=SPACE$(128)
1120 CALL IBRD(TEKDEV1%,RD$)
1130 DIR$=RD$
1140 WHILE RIGHT$(DIR$,1)=" "
1150 DIR$=LEFT$(DIR$,LEN(DIR$)-1)
1160 WEND
1170 PRINT "Current files in A:TEST1:"
1180 PRINT DIR$
1190 INPUT "Press ENTER to rename the file STO001.WFA to WAV-
     NO1.WFA",A$
```

```
1200 WRT$ = "rename 'A:\TEST1\STO001.WFA','A:\TEST1\WAV-
     NO1.WFA' "
1210 REM
1220 REM line 1200 renames the stored waveform to WAVNO1.WFA.
1230 REM
1240 CALL IBWRT(TEKDEV1%,WRT$)
1250 INPUT "Press ENTER to display a list of the files in the directo-
     ry.",A$
1260 WRT$ = "dir?"
1270 REM
1280 REM Line 1260 lists the contents of TEST1.
1290 REM
1300 CALL IBWRT(TEKDEV1%,WRT$)
1310 RD$ = SPACE$(128)
1320 CALL IBRD(TEKDEV1%,RD$)
1330 DIR$ = RD$
1340 WHILE RIGHT$(DIR$,1) = " "
1350 DIR$ = LEFT$(DIR$,LEN(DIR$)-1)
1360 WEND
1370 PRINT "Current files in A:TEST1:"
1380 PRINT DIR$
1390 INPUT "Press ENTER to copy the file WAVNO1.WFA to the
     directory called TEST2.",A$
1400 WRT$ = "dcopy 'A:\TEST1\WAVNO1.WFA','A:\TEST2\WAV-
     NO1.WFA' "
1410 REM
1420 REM Line 1400 copies the file WAVNO1.WFA in TEST1 to TEST2.
1430 REM
1440 CALL IBWRT(TEKDEV1%,WRT$)
1450 PRINT "Press ENTER to change directories to TEST2"
1460 INPUT "and display a list of the files in the directory.",A$
1470 WRT$ = "cd 'A:\TEST2';dir?"
1480 REM
```

```
1490 REM Line 1470 changes the current directory to TEST2 and lists
     its contents.
1500 REM
1510 CALL IBWRT(TEKDEV1%,WRT$)
1520 RD$ = SPACE$(128)
1530 CALL IBRD(TEKDEV1%,RD$)
1540 DIR$ = RD$
1550 WHILE RIGHT$(DIR$,1) = " "
1560 DIR$ = LEFT$(DIR$,LEN(DIR$)-1)
1570 WEND
1580 PRINT "Current files in A:TEST2:"
1590 PRINT DIR$
1600 INPUT "Press ENTER to delete the file WAVNO1.WFA from the
     directory called TEST2.",A$
1610 WRT$ = "delete 'A:\TEST2\WAVNO1.WFA' "
1620 REM
1630 REM Line 1610 deletes WAVNO1.WFA from TEST2.
1640 REM
1650 CALL IBWRT(TEKDEV1%,WRT$)
1660 INPUT "Press ENTER to verify the file has been deleted.",A$
1670 WRT$ = "dir?"
1680 REM
1690 REM Line 1670 lists the contents of the TEST2 directory.
1700 REM
1710 CALL IBWRT(TEKDEV1%,WRT$)
1720 RD$ = SPACE$(128)
1730 CALL IBRD(TEKDEV1%,RD$)
1740 DIR$ = RD$
1750 WHILE RIGHT$(DIR$,1) = " "
1760 DIR$ = LEFT$(DIR$,LEN(DIR$)-1)
1770 WEND
1780 PRINT "Current files in A:TEST2:"
1790 PRINT DIR$
```

```
1800 PRINT "Press ENTER to remove the directory called TEST2."
1810 INPUT "(Directories must be empty, before they can be re-
     moved.)",A$
1820 WRT$= "rmdir ´A:\TEST2´ "
1830 REM
1840 REM Line 1820 removes the directory TEST2.
1850 REM
1860 CALL IBWRT(TEKDEV1%,WRT$)
1870 PRINT "Press ENTER to initialize the DSA and set the storage
     device"
1880 INPUT "(for stored waveforms) to RAM.",A$
1890 WRT$= "init;setdev sto:ram"
1900 REM
1910 REM Line 1890 initializes the DSA and sets the stored waveform
     device to RAM.
1920 REM
1930 CALL IBWRT(TEKDEV1%,WRT$)
1940 INPUT "Press ENTER to define a waveform as channel 1 of the left
     plug-in module.",A$
1950 WRT$= "trace1 description:´L1´ "
1960 REM
1970 REM line 1950 defines the source description of the waveform (Left
     plug-in module, channel 1).
1980 REM
1990 CALL IBWRT(TEKDEV1%,WRT$)
2000 INPUT "Press ENTER to autoset the DSA.",A$
2010 WRT$= "autoset start"
2020 REM
2030 REM Line 2010 starts the autoset function for the selected trace.
2040 REM
2050 CALL IBWRT(TEKDEV1%,WRT$)
2060 INPUT "Press ENTER to change directories to TEST1.",A$
2070 WRT$= "cd ´A:\TEST1´ "
```

```
2080 REM
2090 REM Line 2070 changes the current directory to TEST1.
2100 REM
2110 CALL IBWRT(TEKDEV1%,WRT$)
2120 INPUT "Press ENTER to set the number of repetitive trigger
     acquisitions to 10.",A$
2130 WRT$= "nreptrig 10"
2140 REM
2150 REM Line 2130 sets the number of repetitive trigger acquisitions to
     10.
2160 REM
2170 CALL IBWRT(TEKDEV1%,WRT$)
2180 PRINT "Press ENTER to set the type of acquisition"
2190 INPUT "to repetitive trigger, and start the waveform acquisi-
     tion.",A$
2200 WRT$= "condacq type:reptrig;digitizer run"
2210 REM
2220 REM Line 2200 sets the conditional acquisition type to repetitive
     trigger and starts the digitizer.
2230 REM
2240 CALL IBWRT(TEKDEV1%,WRT$)
2250 INPUT "Press ENTER to display a list of waveforms stored in
     RAM.",A$
2260 WRT$= "stolist?"
2270 REM
2280 REM Line 2280 queries the RAM for stored waveforms.
2290 REM
2300 CALL IBWRT(TEKDEV1%,WRT$)
2310 RD$=SPACE$(255)
2320 CALL IBRD(TEKDEV1%,RD$)
2330 STO$=RD$
2340 WHILE RIGHT$(STO$,1)=" "
2350 STO$=LEFT$(STO$,LEN(STO$)-1)
```

```
2360 WEND
2370 PRINT "Current files in RAM:"
2380 PRINT STO$
2390 INPUT "Press ENTER to copy the RAM stored waveforms to
     disk.",A$
2400 WRT$= "dcopy sto1:'A:STO1';dcopy sto2:'A:STO2';dcopy
     sto3:'A:STO3';dcopy sto4:'A:STO4';dcopy
     sto5:'A:STO5';dcopy sto6:'A:STO6';dcopy
     sto7:'A:STO7';dcopy sto8:'A:STO8';dcopy
     sto9:'A:STO9';dcopy sto10:'A:STO10' "
2410 REM
2420 REM Line 2400 copies the ten files in RAM to disk.
2430 REM
2440 CALL IBWRT(TEKDEV1%,WRT$)
2450 INPUT "Press ENTER to verify that the disk contains the files.",A$
2460 WRT$= "dir?"
2470 REM
2480 REM Line 2460 displays the contents of the current directory.
2490 REM
2500 CALL IBWRT(TEKDEV1%,WRT$)
2510 RD$=SPACE$(251)
2520 CALL IBRD(TEKDEV1%,RD$)
2530 DIR$=RD$
2540 WHILE RIGHT$(DIR$,1) = " "
2550 DIR$=LEFT$(DIR$,LEN(DIR$)-1)
2560 WEND
2570 RD$=SPACE$(255)
2580 CALL IBRD(TEKDEV1%,RD$)
2590 DIR2$=RD$
2600 WHILE RIGHT$(DIR2$,1) = " "
2610 DIR2$=LEFT$(DIR2$,LEN(DIR2$)-1)
2620 WEND
2630 PRINT "Current files in A:TEST1:"
```

```
2640 PRINT DIR$
2650 PRINT DIR2$
2660 INPUT "Press ENTER to re-enable the front panel.",A$
2670 WRT$= "fpanel on"
2680 REM
2690 REM Line 2670 turns the front panel on.
2700 REM
2710 CALL IBWRT(TEKDEV1%,WRT$)
2720 PRINT "End of example 9."
2730 PRINT "Press ENTER to return to the Examples menu,"
2740 PRINT "press ´Q´ to quit the program without exiting BASIC,"
2750 INPUT "or press ´S´ to quit the program and exit BASIC.",A$
2760 IF LEFT$(A$,1) = "Q" OR LEFT$(A$,1) = "q" THEN END
2770 IF LEFT$(A$,1) = "S" OR LEFT$(A$,1) = "s" THEN SYSTEM
     ELSE LOAD "menu.bas",R
```

Appendix A: Improving System Performance

Optimum system performance means acquiring accurate data with the fastest system throughput. This appendix discusses the components of system performance and suggests techniques to improve them.

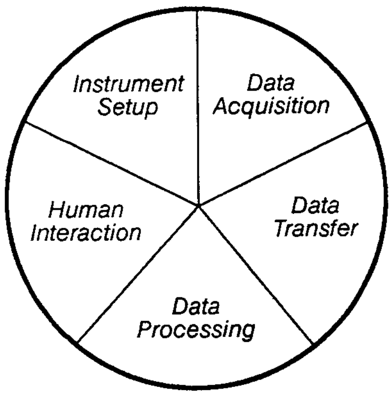
First, you must be familiar with your instrument controller, measurement instruments, data recorders, and with your chosen software (operating system, programming language, device drivers, etc.). When you know the capabilities of your system, you are better prepared to write efficient application programs.

Then you must decide which interface (GPIB or RS-232-C) best suits your application needs.

A good way to develop a thorough understanding of your system instruments is to study their manuals. In particular, learn about the command vocabulary and data formats (for example, ASCII or binary) for each instrument and learn how each device buffers data and executes commands. This gives you information about which hardware configurations and program algorithms will be most efficient for your application.

Components of System Performance

Five major components affect the overall system performance, as summarized in the following illustration. The sum of these components is the total time required to execute your application.



System Performance Components

The contribution of each component to the total execution time varies, based on your specific system configuration.

For example, a data logging system generally requires little time to set up and doesn't require operator intervention. However, significant time is spent acquiring and transferring data. In contrast, a production test system may spend less time acquiring data, but more time processing data and interacting with the operator. Each situation requires a different focus for optimizing system performance.

The best way to determine the time that each component contributes to system performance is to measure it. You can use a real time clock in your controller to do this.

For example, to measure the time it takes to execute a PP? (peak-to-peak amplitude) measurement query, turn on your controller real-time clock before the command, then read the elapsed time immediately after reading the PP? response. Repeating this measurement a few times under varying system configurations will produce typical values you can use to judge the impact of each component on system performance.

Instrument Setup Time

Instrument setup time can be divided into two parts: the time required to decode and execute a setting command, and the time required for new settings to stabilize.

The time it takes to decode and execute a single DSA command is usually short, but if a command initiates a complex or lengthy operation, it can increase the setup time.

For instance, some commands require the DSA to check whether any settings associated with the command function have changed prior to the command. If any associated settings have changed, the DSA must load the new settings into its hardware.

The second part of the setup time is the time it takes the DSA to settle to the specified setting. For example, when vertical size is set automatically, the DSA takes a reading of the input voltage, tests for under- or over-voltage conditions, steps the vertical scale range up or down, and takes another reading. Several readings might have to be made until the correct range is determined. The process stops once the reading is within the new vertical scale range. Thus, a single change in test conditions can cause a significant change in setup time.

Optimizing setup time – requires reducing the number of setting changes or reducing the time required for the DSA to execute the setting changes.

Here are some suggestions to optimize setup time:

- Group tests that use common settings.
- Set your ranges explicitly. Generally, autoset takes more time.
- First set up instruments that require more settling time. While they are settling, you can be setting up other devices.
- Use the store setting features. Reconstructing a setting takes more time.

- Use low byte-count and less complex commands. For example, use the LONGFORM OFF command for abbreviated responses to queries. This can significantly reduce the byte count for data transfers.

Data Acquisition Time

The second component of system performance is the time required to acquire a full record of the input source (the selected waveform). This is the data acquisition time.

The DSA has two acquisition modes: real time and equivalent time. In real time acquisition, all waveform samples are taken at one trigger event. Equivalent time acquisition takes several trigger events to fill a waveform record.

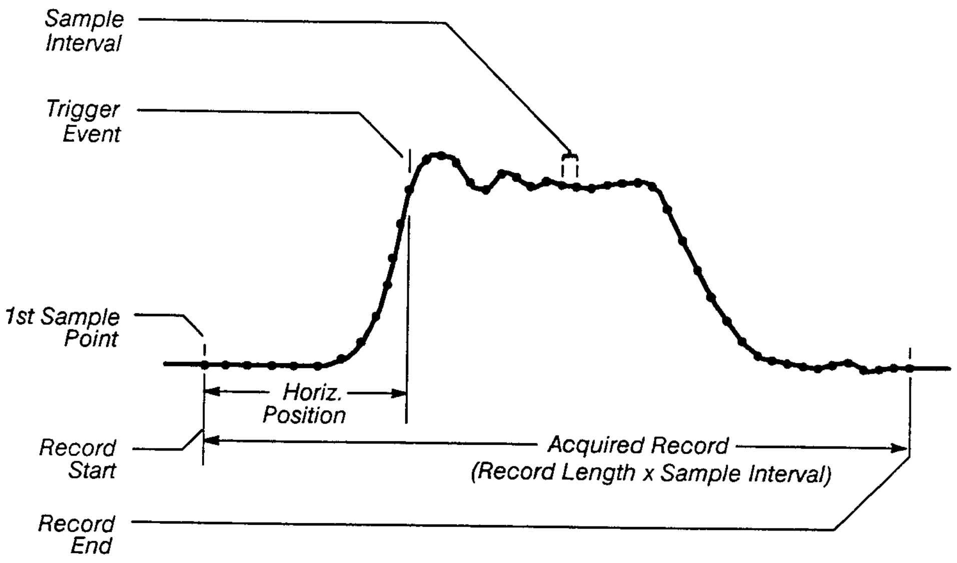
In real time mode, the data acquisition time corresponds to the duration of the record (the record length times the sample interval). In equivalent time mode, several factors affect the data acquisition time: frequency of trigger events, horizontal size, and the waveform record length.

Optimizing data acquisition time – requires careful attention in setting up the acquisition.

Here are some suggestions to optimize data acquisition time:

- Faster digitizing can be achieved by increasing the repetition rate of the input signal (if possible), or by changing the time base setting. The fastest digitizing rate occurs at the maximum real time sample rate. At a slower sample rate, the DSA takes longer to acquire a waveform record. Faster effective sample rates use equivalent time random sampling, which is slower than real time acquisition.

- Use an operation-complete SRQ interrupt instead of waiting for the acquisition to finish. You can continue processing while the acquisition completes.



Components of Data Acquisition Time in Real Time Mode

Data Transfer Time

The third component of system performance is the time it takes to transfer data from one instrument to another. The data transfer time depends on two factors: the number of bytes being transferred and the time it takes to transfer each byte.

The number of bytes transferred depends on the size of the message (number of characters) and the data format (for example, ASCII or binary). For GPIB transfers, the transfer rate depends on the speed of the slowest addressed device on the bus. For RS-232-C transfers, the data transfer rate depends on the baud rate setting of the DSA and controller.

Understanding the processing of GPIB and RS-232-C I/O statements is the key to estimating data transfer times.

GPIB I/O execution time – consists of five parts:

- Addressing sequence
- Unaddressing sequence
- Statement overhead
- Buffer overhead
- Data overhead

The addressing and unaddressing sequences are composed of GPIB interface messages that make the DSA talk or listen to the controller. The time required depends on the data handshake rate of the slowest device connected to the bus.

Statement overhead is the time required to examine the I/O statement for content and syntax (parsing). For the controller, this includes evaluating the statement's I/O function(s) and other expressions, and the statement clauses (DSA commands).

Buffer overhead is the time it takes to fill or empty an I/O memory register with the I/O statement. This depends on the the amount of data (how many characters), and the type of data (string or numeric, ASCII or binary).

Data overhead is the time it takes to transfer data over the interface bus. Again, the time depends on the data transfer rate of the slowest device involved in the transfer, and on the amount and type of data transferred (for example, numeric arrays are a little faster than an equivalent number of scalar variables). This includes the spaces and formatting characters for each message. The total data transmission time is the number of bytes being transferred divided by the data transfer rate (in bytes/second).

RS-232-C I/O execution time – consists of five parts, similar to the GPIB:

- Statement overhead
- Buffer overhead
- Start message
- Data overhead
- Stop message

The RS-232-C statement and buffer overheads consist of the same elements as in GPIB I/O.

The start and stop message time consists of the time required to send one or two bits (depending on the configuration of the RS-232-C interface) before and after each byte of the message in order to synchronize the transmission.

The RS-232-C data overhead time is determined by the baud rate setting of the RS-232-C port on each device.

Since data are sent serially over the RS-232-C interface, additional time is required to convert information from serial-to-parallel, for input data, and from parallel-to-serial for output data. Thus, throughput for an RS-232-C message tends to be slower than throughput for the same GPIB message.

Optimizing data transfer time – involves two major areas. The first is the system configuration, and the second is the program that controls the transfer.

These suggestions will help you optimize the system configuration:

- Choose instruments that have an optimum transfer rate as near as possible to the bus capacity.
- If your controller has more than one GPIB port, use frequently interacting devices on one bus, or put faster devices together on one bus.

- Use direct-memory access (DMA) transfers whenever possible and keep the faster instruments on this bus.
- Be sure to unaddress slow devices when they are not required in the transfer.
- If you have two ports, put a device under test (DUT) on one bus, and the test equipment on the other bus. Then, if the DUT has an error or malfunction, it won't affect the test equipment.

Follow these suggestions to optimize transfer program parameters:

- Choose the most efficient I/O statements that your controller provides. In most cases high-level commands are fastest, except where long strings are encountered. Then use low-level transfer commands (if provided).
- Minimize bus traffic by reducing the number of bytes being sent. You can do this by abbreviating command names, deleting unnecessary spaces, and omitting unnecessary zeros.
- Minimize buffer overhead. This can be done by defining buffer size (usually possible for most controllers) to accommodate the entire data transfer. You may also store the data within a string variable; string variables store data directly from the I/O buffer and reduce overhead time.
- Use binary block data transfers if possible. Binary data is a little more complicated to handle than ASCII data, but binary transfers tend to be much faster because they involve fewer bytes than an equivalent ASCII transfer.

Data Processing Time

The fourth component of system performance is the time required to manipulate the acquired data for a desired result.

The data processing time is composed of the time it takes the DSA to manipulate the data, plus the time required by the controller to further process the data. The DSA can deliver raw, semi-processed data, or completely processed data, depending on the requirements of the application. The processing speed of the DSA depends on the type or complexity of the operation performed.

Optimizing data processing time – involves using faster algorithms and distributed processing.

These suggestions will help you optimize data processing time:

- Evaluate your choice of algorithms to ensure that the most efficient operations are used for your application and system configuration.
- Use implied array operations instead of for/next loops in your controller programs. This allows numeric operations to be performed much faster. The implied array operation creates temporary arrays to perform the implicit operation (for example, add a scalar to the array) rather than an element-by-element operation.
- Carefully select the data type for your controller programs. Try to group integer, short floating-point, and long floating-point operations. It takes less time to process each as a group, rather than to do mixed data type operations that require conversion from one format to another.
- Evaluate your measurement needs to identify the most effective device for each data processing task. For example, would the DSA best perform a given function on a waveform, or would your controller perform that function more quickly?

Human Interaction Time

The fifth major component of system performance is determined by operator intervention required to enter test parameters or to make adjustments to a device under test (DUT).

This component can easily become the largest part of the total operating time for a system. Direct measurement of this component is the best way to determine its effect on system performance.

Optimizing human interaction time – can be difficult. The best advice is to avoid the need for human interaction with the system as much as possible.

Follow these suggestions to optimize human interaction time:

- Use programmable interfaces and switches to route signal connections wherever possible. These include programmable relay scanners, multi-function interfaces, and signal multiplexers.

- Keep the user interface simple. The DSA is designed especially for this purpose. User menus are quick and easy to use, so you can make changes quickly.

Appendix B: Reserved Words

Reserved words represent the entire set of predefined command words for the DSA, including headers, links, and arguments.

In this section, reserved words appear in mixed case, with the required minimum substring in uppercase.

A

ABBwfmpre
ABOrt
ABORTIng
ABSOlute
ABStouch
AC
ACCumulate
ACHf
ACLf
ACNoise
ACState
ACTions
ADJtrace
ALEvel
ALL
ALL_Wavfrm
ALTinkjet
ALWays
AMPLitude
AMPoffset
AMPS
ANBlevel
ANLevel
ARMed
ASCii
AUTO
AUTOAcq
AUTOLevel
AUTOSet
AVG
AVGType

B

BACKWeight
BASEDelta
BASELAbel
BASeline
BASEName
BAUd
BINary
BINHex
BIT/nr
BITMap
BLAckman
BLHarris
BN.fmt
BOTh
BW
BWHi
BWLo
BYPassed
BYT.or
BYT/nr

C

C.WINBottom
C.WINLeft
C.WINRight
C.WINTop
CALDue
CALIbrator
CALJumper
CALProbe
CALStatus
CALTempdelta
CCAlconstants
CD
CENter
CENTRonics
CH
CHDir
CHIme
CHKDsk
CHKsm0
CHSkew
CLEar
CMDerr
CMOde
COLor
COLORMap

COMpare
CONDacq
CONFig
CONSecpts
CONTInuous
COPy
COUpling
CPLugin
CR
CRLf
CROss
CROSSHair
CRVchk
CURMode
CURRent
CURSor
CURVe

D

D.WINBottom
D.WINLeft
D.WINRight
D.WINTop
DAInt
DATA
DATACompress
DATAFormat
DATE
DBFund
DBM
DBVPeak
DBVRms
DC
DCHf
DCNoise
DCOpy
DCSup
DEBug
DEF
DEFAult
DEGrees
DELAy
DELete
DELTa
DEScription
DIAg
DIGitizer
DIGJumpers
DIR
DIREction
DISAble
DISK
DISPlay
DISTal
DIThered
DIVS
DLYtrace
DOT1Abs
DOT1Rel
DOT2Abs
DOT2Rel
DOTs
DRAft
DSYmenu
DSYSTOFmt
DSYStotd
DUAl
DUTy

E

ECHo
ECL
EDGe
EKUtil
EMPty
ENAble
ENCdg
END
ENHanced
ENV
EOL
EQ
ERAsed
EVEN
EVENT
EVENTS
EVHoldoff
EXErr
EXIt
EXPDJumpers
EXPMJumpers
EXWarn

F

FAIled
FALltime
FASt
FEOi
FFT
FILl
FILTer
FINTerval

FLAgging
FORce
FORMat
FPAnel
FPNext
FPS
FPSList
FPSNum
FPUpdate
FREq
FRESolution
FROm
FSPan
FULl

G

GAIn
GPIb
GRAticule
GRLocation
GRType
GT

H

H1Bar
H2Bar
HAMming
HANning
HARd
HARMonic
HBArs
HCP
HERtz
HIPrec
HIRes
HIST.pt
HISTogram
HISTScaling
HMAg
HNUmber
HOLd
HORiz
HPGl
HPOsition
HSBatt
HSYs
HUE
HUNdredths
HVPosition
HVSize

I

ID
IDLe
IDProbe
IMPedance
INCAcq
INErr
INFPersist
INIt
INPut
INTensity
INTERleave
INTERPolation
INWarn

J

JMPR

K

KEEp

L

LABAbs
LABel
LABRel
LCAlconstants
LEFt
LENgth
LF
LFCr
LIGhtness
LINear
LMZone
LOG10
LONgform
LOTus
LOWer
LPLugin
LSB
LT

M

MAIn
MAINPos
MANual
MAX
MCAlconstants
MEAN
MEAS
MEMWrap
MESial
MID

MIN
MINUs
MKDir
MLEvel
MNSCoupling
MNSOffset
MNSProbe
MODe
MPEak
MSB
MSCount
MSCRoss
MSDElay
MSDUty
MSFAlltime
MSFRequency
MSGain
MSLIst
MSLOpe
MSMAx
MSMEan
MSMID
MSMIN
MSNum
MSOvershoot
MSPDelay
MSPEriod
MSPHase
MSPP
MSREPCRoss
MSREPDElay
MSREPDUty
MSREPFAlltime
MSREPFReq
MSREPGain
MSREPMAx
MSREPMEan
MSREPmeas
MSREPMID
MSREPMIN
MSREPOvershoot
MSREPPDelay
MSREPPEriod
MSREPPHase
MSREPPP
MSREPRisetime
MSREPRMs
MSREPSFrequency
MSREPSkew
MSREPSMagnitude
MSREPTHd
MSREPTTrig
MSREPUndershoot
MSREPWidth
MSREPYTEnergy
MSREPYTMns_area
MSREPYTPls_area
MSRisetime
MSRMs
MSSFRequency
MSSkew
MSSMAgnitude
MSTat
MSTHd
MSTO
MSTOCRoss
MSTODElay
MSTODUty
MSTOFAlltime
MSTOFReq
MSTOGain
MSTOMAx
MSTOMEAN
MSTOMEAS
MSTOMID
MSTOMIN
MSTOOvershoot
MSTOPDelay
MSTOPEriod
MSTOPHase
MSTOPP
MSTORisetime
MSTORMs
MSTOSFrequency
MSTOSkew
MSTOSMagnitude
MSTOTHd
MSTOTTrig
MSTOUndershoot
MSTOWidth
MSTOYTEnergy
MSTOYTMns_area
MSTOYTPls_area
MSTTrig
MSUndershoot
MSWidth
MSYs

MSYTEnergy
MSYTMns_area
MSYTPls_area
MTIme
MTRack
MULTitrace

N

NAVg
NENHanced
NENV
NEVer
NEWconfig
NEXt
NEXTFps
NEXTRep
NEXTSto
NHISt.pt
NLEvel
NONe
NORmal
NOTErased
NOTInstalled
NOTrg
NR.pt
NREPCurve
NREPMeas
NREptrig
NT
NTAuto
NULl
NUMPts
NVRam
NWAVfrm
NWFm

O

ODD
OFF
OFFSet
OHMs
ON
OPCmpl
OPTional
OPTIONS
ORIginal
OUTput
OVErshoot

P

PAIred
PANzoom
PARity
PASsed
PATh
PCTg
PDElay
PEAK
PERiod
PERSistence
PHAse
PIN24
PIN8
PINdex
PIVersion
PIVOt
PLOtter
PLSCoupling
PLSOffset
PLSProbe
PLUs
PMPeak
PORt
POWeron
PP
PPEak
PREvious
PRInter
PRINTIng
PROBe
PROTect
PROXimal
PSINx
PT.fmt
PULse
PZMode
RATe

R

RCAlconstants
READout
RECall
RECOver
RECTangular
REDuced
REFErence
REFLevel
REFREsh
REFset
REFTrace
RELative
REMAining
REMove
RENAme
RENDir
REPCRoss

REPCurve
REPDElay
REPDUty
REPeat
REPFAlltime
REPFReq
REPGain
REPMAx
REPMEan
REPMeas
REPMID
REPMIN
REPOvershoot
REPPDelay
REPPEriod
REPPHase
REPPP
REPRisetime
REPRMs
REPS
REPSFrequency
REPSKew
REPSMagnitude
REPTHd
REPTrig
REPTTrig
REPUndershoot
REPWidth
REPYTEnergy
REPYTMns_area
REPYTPls_area
RI
RIGht
RISetime
RMDir
RMS
RMSDev
RMZone
RPLugin
RQS
RS232
RUN

S

SATuration
SAVe
SAVEFactory
SCAn
SCANStowfm
SCLockd
SCReen
SEConds
SECUre
SELect
SELECTEd
SELFcal
SENsitivity
SEQuence
SET
SETDev
SETPIPE
SETSeq
SFRequency
SHOrt
SHOWPts
SIGMA1
SIGMA2
SIGMA3
SINgle
SINX
SKEw
SLOpe
SMAgnitude
SMOde
SNRatio
SOFt
SOUrce
SPEaker
SPLit
SPOoling
SRQ
SRQMask
STANdard
STARt
STATHist
STATIstics
STAtus
STByte
STO
STOFmt
STOList
STONum
STOP
STOPBits
STORe
STORE_Recall
STRing
SUMMation
SYSMON
SYStem

T

TBMain
TBWin
TEK4692
TEK4696
TEK4697
TEKSECURE
TESt
TEXt
THD
TIHoldoff
TIMe
TIMER1
TIMER2
TO
TOPDelta
TOPline
TOTalpts
TR
TRAce
TRANUm
TRG
TRIAngular
TRIgger
TRLevel
TRMain
TRSep
TRWin
TSMain
TSTime
TTAverage
TTL
TTRig
TYPe

U

UID
UN
UNDEF
UNDershoot
UNDO
UNIts
UNWrap
UPPer
UPTime
USEr
USERId
USIng
UTIlity
UTILITY1
UTILITY2
UTILITY3

V

V1Bar
V2Bar
VARPersist
VBArs
VC
VCOffset
VECtors
VERBose
VERt
VOLts
VPEak
VPOsition
VRMs
VSIze

W

WATts
WAVfrm
WFId
WFMCalc
WFMpre
WFMScaling
WFMSCAN
WHOle
WIDth
WIN1Pos
WIN2Pos
WINDow
WRAp
WTMode

X

X
XCOord
XDIv
XINcr
XMUlt
XQUal
XTNd
XUNit
XY
XZEro

Y

Y
YCOord
YDIv
YMUlt

YQUal
YTEnergy
YTMns_area
YTPls_area
YUNit
YZEro
YZEro

Appendix C: Character Sets

The character sets include standard ASCII characters and a special set of characters that include math, Greek, European, and graphic symbols.

The special "escape" characters are formed by putting an ASCII escape character (octal 33) in front of another ASCII character. For example, to place an integral math symbol ($\int$) on the DSA display, enter an escape character (represented by *<ESC>*) followed by the letter **d**.

```
TEXT STRING:"<ESC>d"
```

For more information on placing characters on the display, see the TEXT command.

The character-set tables begin on the following page.

ASCII Character Set

	0	1	2	3	4	5	6	7
0	0 NU	16 DL	32	48 0	64 @	80 P	96 `	112 p
1	1 SH	17 D1	33 !	49 1	65 A	81 Q	97 a	113 q
2	2 SX	18 D2	34 "	50 2	66 B	82 R	98 b	114 r
3	3 EX	19 D3	35 #	51 3	67 C	83 S	99 c	115 s
4	4 ET	20 D4	36 $	52 4	68 D	84 T	100 d	116 t
5	5 EQ	21 NK	37 %	53 5	69 E	85 U	101 e	117 u
6	6 AK	22 SY	38 &	54 6	70 F	86 V	102 f	118 v
7	7 BL	23 EB	39 '	55 7	71 G	87 W	103 g	119 w
8	8 BS	24 CN	40 (	56 8	72 H	88 X	104 h	120 x
9	9 HT	25 EM	41)	57 9	73 I	89 Y	105 i	121 y
A	10 LF	26 SB	42 *	58 :	74 J	90 Z	106 j	122 z
B	11 VT	27 EC	43 +	59 ;	75 K	91 [	107 k	123 {
C	12 FF	28 FS	44 ,	60 <	76 L	92 \	108 l	124 |
D	13 CR	29 GS	45 -	61 =	77 M	93]	109 m	125 }
E	14 SO	30 RS	46 .	62 >	78 N	94 ^	110 n	126 ~
F	15 SI	31 US	47 /	63 ?	79 O	95 _	111 o	127 ▒

Escape Character Set

	0	1	2	3	4	5	6	7
0	0 Ä	16 Ñ	32 color 1	48	64 Π	80 π	96 ↓	112 …
1	1 ä	17 ñ	33 color 2	49	65 α	81 θ	97 ↑	113 Ä
2	2 Ö	18 ¿	34 color 3	50	66 γ	82 ρ	98 →	114 🄴
3	3 ö	19 ¡	35 color 4	51	67 δ	83 Σ	99 ←	115 🅁
4	4 Ü	20 Ã	36 color 5	52	68 Δ	84 τ	100 ∫	116 🅃
5	5 ü	21 ã	37 color 6	53	69 ϵ	85 υ	101 ÷	117 ┘
6	6 à	22 À	38 color 7	54	70 φ	86 ν	102 °	118 ┐
7	7 è	23 Õ	39	55	71 Γ	87 ω	103 √	119 ┌
8	8 á	24 õ	40	56	72 Θ	88 χ	104 ¬	120 └
9	9 é	25 É	41	57	73 ι	89 ξ	105 ±	121 ┼
A	10 Å	26 Ø	42	58	74 ψ	90 ζ	106 ≠	122 ─
B	11 å	27 ø	43	59	75 κ	91 Φ	107 ≤	123 ├
C	12 Æ	28 Œ	44	60	76 λ	92 Λ	108 ≥	124 ┤
D	13 æ	29 œ	45	61	77 μ	93 Ψ	109 ©	125 ┴
E	14 ç	30 Ç	46	62	78 ℏ	94 σ	110 ®	126 ┬
F	15 β	31 ∞	47	63	79 Ω	95 Ξ	111 ≈	127 │

Appendix D: Utility Programs

Common external interface operations include:

- Taking measurements
- Binary waveform transfer into an array
- Storing and recalling front panel settings
- Handling SRQs (DSA service requests)
- String transfer to the DSA display

The following programs demonstrate these operations on popular instrument controllers.

Setup

These applications are for use with the Tektronix PEP series of controllers, or IBM PC-compatible computers configured with a National Instruments GPIB-PC Interface Card. A compatible computer with a similar GPIB interface card can also be used. These programs are written in Microsoft QuickBASIC, Version 4.0.

We also show Hewlett-Packard 200 and 300 Series controller versions of these programs. These programs are written in HP BASIC, Versions 2.1 through 4.0.

Interface Configuration

Set up the GPIB parameters of the DSA as follows:

GPIB Interface Configuration

GPIB Function	Selection
Mode	TalkListen
Address	1
Terminator	EOI/LF
Debug	Off

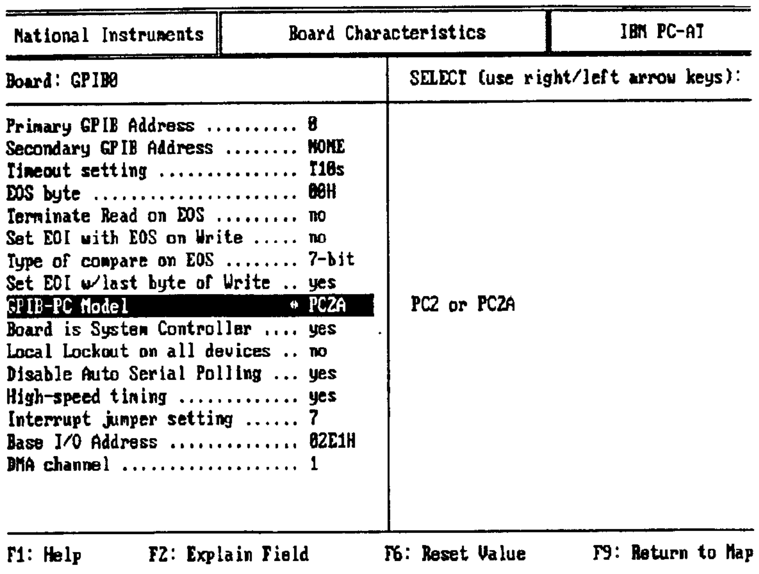
Computer Interface Configurations

The following information describes how to set up your GPIB driver system for using these programs.

Tektronix PEP Series or IBM PC-Compatible Computers – require you to invoke the configuration program for your GPIB interface. For example, for the National Instruments GPIB-PC Interface Card, invoke the **ibconfig.exe** file and follow the instructions.

The following illustrations show the appropriate configuration for using these utility programs.

The first illustration shows how your GPIB driver board characteristics should be set, and the second illustration shows how your device (DSA) characteristics should be set.



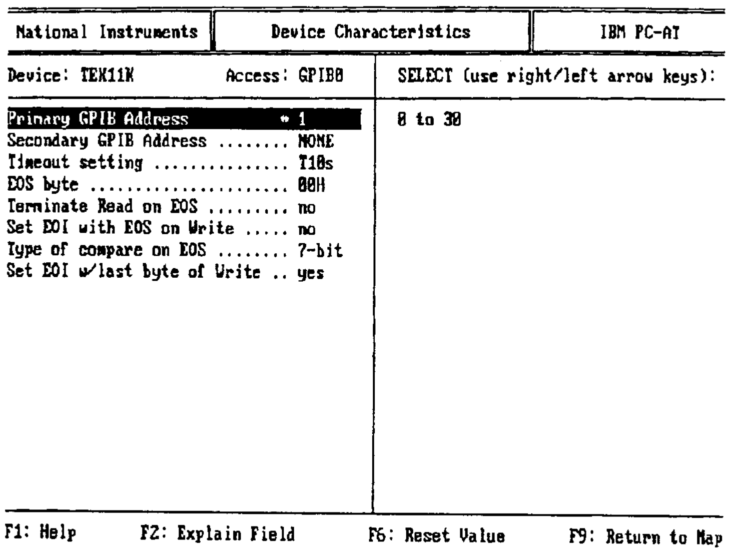
```
National Instruments      Board Characteristics        IBM PC-AT

Board: GPIB0                           SELECT (use right/left arrow keys):

Primary GPIB Address .......... 0
Secondary GPIB Address ........ NONE
Timeout setting ............... T10s
EOS byte ...................... 00H
Terminate Read on EOS ......... no
Set EOI with EOS on Write ..... no
Type of compare on EOS ........ 7-bit
Set EOI w/last byte of Write .. yes
GPIB-PC Model                 ◆ PC2A   PC2 or PC2A
Board is System Controller .... yes
Local Lockout on all devices .. no
Disable Auto Serial Polling ... yes
High-speed timing ............. yes
Interrupt jumper setting ...... 7
Base I/O Address .............. 02E1H
DMA channel ................... 1

F1: Help      F2: Explain Field      F6: Reset Value      F9: Return to Map
```

GPIB Driver-Board (Controller) Settings



```
National Instruments    Device Characteristics         IBM PC-AT

Device: TEK11K          Access: GPIB0   SELECT (use right/left arrow keys):

Primary GPIB Address           * 1      0 to 30
Secondary GPIB Address ........ NONE
Timeout setting ............... T10s
EOS byte ...................... 00H
Terminate Read on EOS ......... no
Set EOI with EOS on Write ..... no
Type of compare on EOS ........ 7-bit
Set EOI w/last byte of Write .. yes

F1: Help      F2: Explain Field      F6: Reset Value      F9: Return to Map
```

GPIB Driver-Device (Oscilloscope) Settings

Refer to your HP 200 or 300 Series controller programming manual for configuration details.

HP 200/300 Series Controllers—These programs require you to load the accompanying "I/O" file for your controller.

Note: *In these examples, it is assumed that the "@Br" and "BR%" variables identify the DSA assigned to the GPIB port of your controller.*

Tektronix PEP or IBM PC/XT/AT Controller

The following five program examples are for IBM controllers.

Taking Measurements

```
CALL IBFIND("tek11k", bd%)
CALL IBWRT(bd%, "MSLIST PER,FREQ,MAX,PP,RISE,
 FALL;MEAS?")
msg$ = SPACE$(200)
CALL IBRD(bd%, msg$)
PRINT msg$
END
```

Transferring a Binary Waveform into an Array

```
REM WFM I/O for the 11k scope using Microsoft
   QuickBASIC 4.0 & BC 6.0
CALL ibfind("tek11k", bd%)
CALL ibwrt(bd%, "LONGFORM ON;SELECT?")
msg$ = SPACE$(80): CALL ibrd(bd%, msg$)
CALL ibwrt(bd%, "ENCDG WAV:BIN;BYT.OR LSB;
 OUTPUT   " + MID$(msg$, 8, 6))

CALL ibwrt(bd%, "CURVE?")
CALL ilrd(bd%, msg$, 20)
 hbyte$ = " ": lbyte$ = " "
CALL ilrd(bd%, hbyte$, 1): CALL ilrd(bd%,
   lbyte$, 1)
bytes = ASC(hbyte$) * 256 + ASC(lbyte$)
nr.pt = (bytes - 1) / 2
DIM wfm%(nr.pt)
CALL ibrdi(bd%, wfm%(), bytes)
CALL ilrd(bd%, msg$, 1)

SCREEN 2:   WINDOW (0, -32767)-(nr.pt, 32767)
PSET (0, wfm%(0))
FOR i = 0 TO nr.pt - 1: PSET (i, wfm%(i)):
NEXT
END
```

Storing and Recalling Front Panel Settings

```
CALL ibfind("tek11k", bd%)
CALL ibwrt(bd%, "ENCDG SET:BINARY;SET?")
msg$ = SPACE$(5000)
CALL ibrd(bd%, msg$)
INPUT "Press Enter to send the setup back to
the   scope", A$
CALL ibwrt(bd%, msg$)
END
```

Handling SRQs

```
CALL ibfind("tek11k", bd%)
CALL ibwrt(bd%, "SRQMASK USER:ON;RQS ON")
PRINT "Press the RQS icon on the DSA (Esc to
exit)"
WHILE INKEY$<>CHR$(27)
    GOSUB POLL
WEND
END

POLL:
 msg$ = SPACE$(80)
 stbyte%=0
 call ibrsp(bd%, stbyte%)
 IF stbyte%<>0 THEN
   CALL ibwrt(bd%, "EVENT?")
   CALL ibrd(bd%, msg$)
   PRINT "Status byte:";stbyte%
   PRINT msg$ : PRINT
 END IF
RETURN
```

Transferring a String to the DSA Display

```
CALL ibfind("tek11k", bd%)
x = 5: REM x: {0 to 49}
y = 5: REM y: {0 to 31}
text$ = "'hello there world'"
msg$ = "text x:" + STR$(x) + ",y:" + STR$(y) +
  ",string:" + text$
CALL ibwrt(bd%, msg$)
END
```

HP 200 & 300 Series Controllers

The following five program examples are for HP controllers.

Taking Measurements

```
10 DIM Meas$[200]
20 ASSIGN @Br TO 701;EOL CHR$(10) END
30 OUTPUT @Br;"MSLIST PER,FRE,MAX,PP,RISE,FALL"
40 OUTPUT @Br;"MEAS?"
50 ENTER @Br;Meas$
60 PRINT Meas$
70 END
```

Transferring a Binary Waveform into an Array

```
 10 ASSIGN @Br TO 701;EOL CHR$(10) END
 20 ASSIGN @Brbin TO 701;FORMAT OFF
 30 OUTPUT @Br;"LONGFORM ON"
 40 OUTPUT @Br "SELECT?"
 50 ENTER @Br;Trace$
 60 OUTPUT @Br;"ENCDG WAVFRM:BIN;BYT.OR MSB;
     OUTPUT "&Trace$[8]
 70 OUTPUT @Br;"CURVE?"
 80 ENTER @Br USING "#,20A,W";Header$,Bytcnt
 90 Nr_pt = (Bytcnt-1)/2
100 ALLOCATE INTEGER Curve(1:Nr_pt)
110 ENTER @Brbin;Curve(*)
120 ENTER @Br USING "B";Cksum
130 PRINT Curve(*)
140 DEALLOCATE Curve(*)
150 END
```

Storing and Recalling Front Panel Settings

```
10 DIM Setting$[5000]
20 ASSIGN @Br TO 701;EOL CHR$(10) END
30 OUTPUT @Br;"ENCDG SET:BINARY;SET?"
40 ENTER @Br USING "-K";Setting$
50 DISP "press CONTINUE to reset the front panel"
60 PAUSE
70 OUTPUT @Br;Setting$
80 END
```

Handling SRQs

```
 10 DIM Event$[100]
 20 ASSIGN @Br TO 701;EOL CHR$(10) END
 30 ON INTR 7 GOSUB Poll
 40 ENABLE INTR 7;2
 50 OUTPUT @Br;"SRQMASK USER:ON;RQS ON"
 60 DISP "press RQS icon on the DSA"
 70 GOTO 70
 80 POLL: Stat = SPOLL(701)
 90  OUTPUT @Br;"EVENT?"
100  ENTER @Br;Event$
110  PRINT Stat,Event$
120  ENABLE INTR 7
130 RETURN
140 END
```

Transferring a String to the DSA Display

```
10 DIM Text$[100]
20 ASSIGN @Br TO 701;EOL CHR$(10) END
30 INPUT "TEXT: ",Text$,"LOCATION: ",X,Y
40 OUTPUT @Br;"TEXT X:";X;",Y:";Y;",
     STRING:'"&Text$&"'"
50 END
```

Appendix E: GPIB Interface Functions

GPIB Interface Functions Implemented

The following table lists the GPIB interface function and electrical function subsets supported by the DSA 601A and DSA 602A Digitizing Signal Analyzers, with a brief description of each.

GPIB Functions

Interface Function	Subset	Meaning
Acceptor Handshake	AH1	The DSA can receive multi-line messages across the GPIB from other devices.
Controller	C0	No Controller capability; the DSA cannot control other devices.
Device Clear	DC1	The DSA can respond both to the DCL (Device Clear) interface message, and to the Selected Device Clear (SDC) interface message when the DSA is listen-addressed.
Device Trigger	DT0	No Device Trigger capability; the DSA does not respond to the GET (Group Execute Trigger) interface message.
Electrical	E2	The DSA uses tri-state buffers, which are optimal for high-speed data transfer.
Listener	L4	The DSA becomes a listener when it detects its listen address being sent over the bus with the ATN line asserted. The DSA ceases to be a listener and becomes a talker when it detects its talk address being sent over the bus with the ATN line asserted.
Parallel Poll	PP0	No Parallel Poll capability; the DSA does not respond to PPC (Parallel Poll Configure), PPD (Parallel Poll Disable), PPE (Parallel Poll Enable), or PPU (Parallel Poll Unconfigure) interface messages, nor does it send out a status message when the ATN and EOI lines are asserted simultaneously.

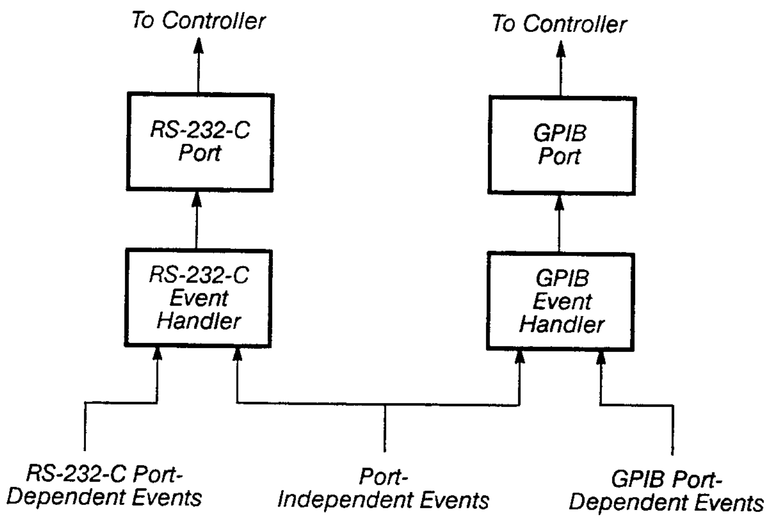
GPIB Functions (Cont.)

Interface Function	Subset	Meaning
Remote/ Local	RL1	The DSA can respond to both the GTL (Go To Local) and LLO (Local Lock Out) interface messages.
Service Request	SR1	The DSA can assert the SRQ line to notify the controller-in-charge that it requires service.
Source Handshake	SH1	The DSA can initiate multi-line messages to send across the GPIB to other devices.
Talker	T5	The DSA becomes a talker when it detects its talk address being sent over the bus with the ATN line asserted. The DSA ceases to be a talker and becomes a listener when it detects its listen address being sent over the bus with the ATN line asserted. The DSA also ceases to be a talker when it detects another device's talk address being sent over the data lines with ATN asserted.

Appendix F: System Event Handling

Status and Event Reporting System

Status bytes and event codes combine to represent common instrument system events. The following illustration shows the remote interface status and event reporting system of the DSA and summarizes its major elements. These elements will be discussed in the following pages.



Remote Interface Status Reporting System Block Diagram

The system events that are generated by the DSA are handled as either port-dependent or port-independent events.

Port-dependent Events

A port-dependent event is generated when one of the following system status conditions occurs:

- Command error
- Execution error
- Execution warning

Port-dependent events are returned only to the port responsible for the event. For example, if the DSA detects a command error in an RS-232-C-only command, the event associated with the error will be returned only to the RS-232-C port.

Port-independent Events

Port-independent events are always returned to both the GPIB and RS-232-C ports. A port-independent event is generated when one of the following system status conditions occurs:

- Internal error
- Internal warning
- Operation complete
- Power-on
- User request

System Event Handling Priorities

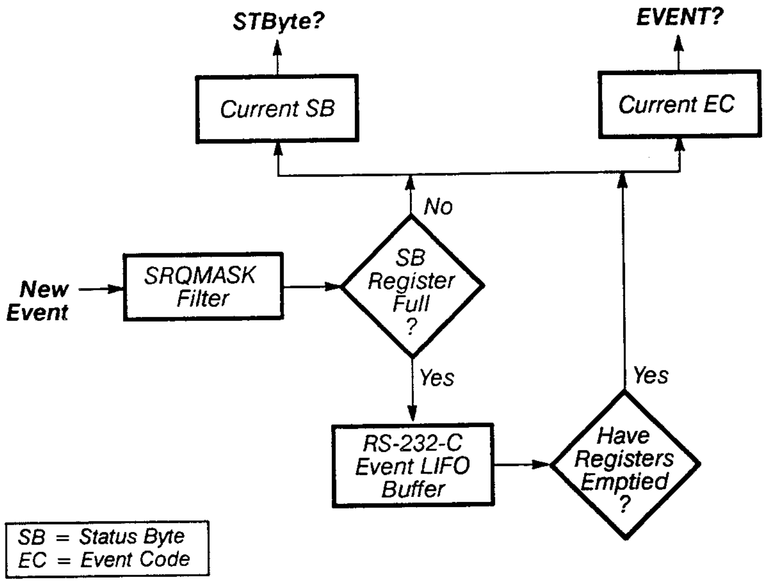
Since more than one event may occur before a GPIB or RS-232-C controller can respond to a service request, the DSA uses the following priorities to report events.

Event Priorities

Priority	Event Class
1	Power on
2	Command error
	Execution error
	Execution warning
	Internal error
	Internal warning
	Operation complete
	User request
3	No status to report

RS-232-C Event Handling

The following illustration is a block diagram of the RS-232-C event handler. The event handler consists of two software registers (SB and EC in the illustration) for the current status byte and current event code, and a LIFO (last-in first-out) buffer.



RS-232-C Event Handling

When a new event is passed to the event handler, the DSA checks the SRQMASK for that event. If the SRQMASK is off, the event is discarded. If the SRQMASK is on, the DSA checks to see if the current status byte register is empty (has "no status to report"). If it is empty, the event handler latches the new status byte and event code into the current status byte and event code registers. Once these registers contain data, all subsequent events are stacked in a 40-event LIFO buffer. Should a new event cause the LIFO buffer to overflow, the oldest event in the buffer is discarded.

Reading the RS-232-C Current Event Registers

An RS232 STBYTE? query returns the contents of the current status byte register. This is a nondestructive read.

An RS232 EVENT? query returns the contents of the current event code register and, assuming the LIFO buffer contains event(s), moves the top LIFO event into the current status byte and event code registers. If the buffer is empty, the current status byte is changed to "No Status To Report" and event code 400 is written to the current event-code register. In effect, EVENT? causes the RS-232-C event handler to update its software registers and make the next event (if any) available for subsequent STBYTE? or EVENT? queries.

GPIB Event Handling

The illustration on the following page is a block diagram of the GPIB event handler. This event handler consists of two software registers (Polled EC and Current EC in the illustration), a LIFO buffer, and the IEEE STD 488 serial poll register (a hardware register).

The RS-232-C current-status-byte software register is functionally equivalent to the serial poll hardware register diagram shown on the following page.

Operation of the GPIB event handler depends upon whether GPIB RQS is set to ON or OFF.

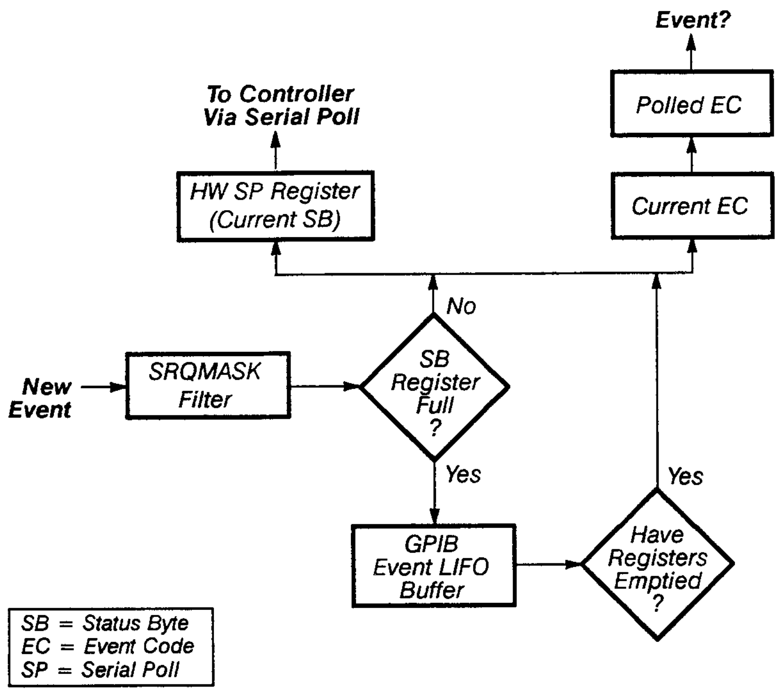
Event Reporting When GPIB RQS is Off

When GPIB RQS is off, the polled event code register is not used when a new event is passed to the event handler. If the SRQMASK for an event is off, then the event is discarded. However, if the SRQMASK for the event is on, the DSA checks to see if the current status byte register is empty or has "no status to report." If it is empty, the event handler latches the new status byte and event code into the hardware serial poll register and current event code register. Once this latched state is entered, all subsequent events are stacked in a 41-event LIFO buffer. Should a new event cause the LIFO buffer to overflow, the oldest event in the buffer is discarded.

Notice that when GPIB RQS is off, the GPIB event handler behaves virtually the same as the RS-232-C event handler, with the exception that the current status byte is stored in a hardware register and not in a software register.

Reading the GPIB Current Event Registers (RQS Off)

A GPIB controller uses an IEEE-STD-488 serial poll to read the contents of the hardware serial poll register, which is identical to the current status byte register. This is a nondestructive read. There is no DSA command provided to read the GPIB hardware serial poll register.



GPIB Event Handling

When RQS is off, only the EVENT? query updates the event handler's software and hardware registers. Repeated serial polls simply return the same status-byte value.

A GPIB EVENT? query command returns the contents of the current event code register and, assuming the LIFO buffer contains event(s), moves the top LIFO event into the current status byte and event code registers. If the buffer is empty, the current status byte is changed to "No Status To Report" and event code 400 is written to the current event code register. In effect, EVENT? causes the GPIB event handler to update its hardware and software registers, and make the next event (if any) available for subsequent serial polls or EVENT? queries.

Event Reporting When GPIB RQS is On

When a new event is passed to the event handler, the same operations are executed as when GPIB RQS is off. The only difference is that bit 7 of the status byte of the new event is set, causing the DSA to assert SRQ after writing the status byte to the serial poll register.

Note that when GPIB RQS is on, the polled event code register is significant. At power-on or whenever RQS is turned on, this register is initialized with event code 0, which is referred to as the NULL event. The description string of the NULL event is:

```
"RQS is ON...status byte pending"
```

Reading the GPIB Current Event Registers (RQS On)

When GPIB RQS is on, it is the IEEE-STD-488 serial poll (not the EVENT?) that causes the event handler to update its event registers.

When the DSA asserts SRQ, an external controller must first serially poll the DSA to read the status byte of the system event that just occurred. The DSA responds to the serial poll by moving the current event code register contents into the polled event code register. The DSA next checks for pending events in the LIFO buffer. If found, the DSA moves the status byte of the top event into the hardware serial poll register, thus updating it and causing the DSA to generate another SRQ. At the same time, the event code for top event is moved into the current event code register, thus updating it. However, if no events are pending in the LIFO buffer, the DSA moves a status byte into the hardware serial poll register that indicates No Status To Report, and its corresponding event code 400 is moved into the current event code register. No SRQ is generated under these conditions.

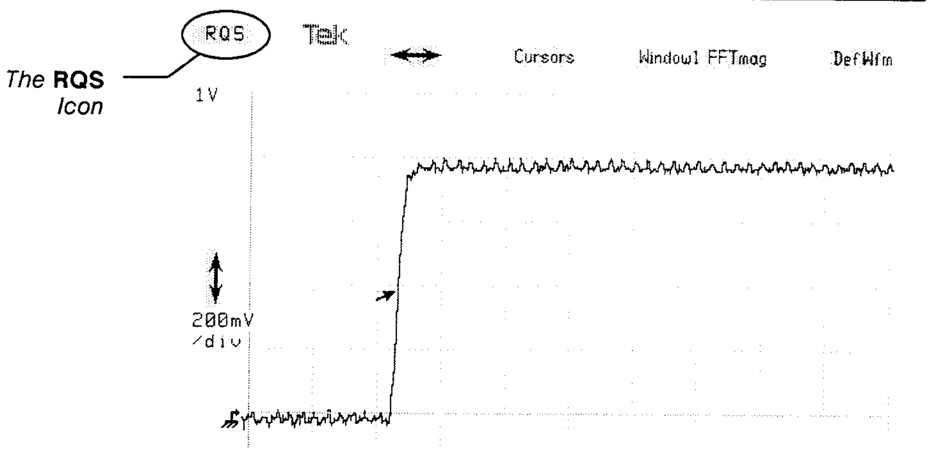
If a controller sends an EVENT? following the serial poll, the DSA returns the contents of the polled event code register and initializes it to the NULL event. Then, at the next serial poll, the DSA again moves the contents of the updated current event code register into the polled event code register. This operation ensures that the status byte and the polled event code match.

Summary of Important Points When RQS is On

- The EVENT? query returns the contents of the polled event code register.
- The proper sequence for reading event registers is to first serial poll the DSA and then, if more information is desired, follow up with an EVENT? query.
- When EVENT? returns the NULL event, the DSA is signaling that a new event has occurred and its status byte must first be polled before its event code can be queried.
- If more than one event is pending and the DSA is serially polled twice with no intervening EVENT?, the event code associated with the first polled status byte is lost.

Turning On the RQS Icon with SRQMASK USER

The SRQMASK USER command allows you to make a Request for Service (RQS) from the front panel. When SRQMASK USER is on at either the GPIB or RS-232-C port, the DSA displays an RQS icon on its front panel. When initially displayed, the RQS icon is not highlighted and is not selected. When you touch the RQS icon, the icon is highlighted and an event 403 ("Front panel RQS icon selected") is reported to the ASCII port. When SRQMASK USER is off at both ports, the icon is not displayed. Since both USER masks are off by default at power-on, the RQS icon is not visible at that time.



RQS Icon on the Front Panel Display

The RQS icon changes from selected to not selected under any one of these circumstances:

- Both GPIB SRQMASK USER and RQS are on and a GPIB controller serially polls (and thereby clears) the status byte associated with event code 403.
- The GPIB SRQMASK USER is on and RQS is off and a GPIB controller uses EVENT? to read (and thereby clear) event code 403 from the GPIB event stack.
- The RS-232-C SRQMASK USER is on and an RS-232-C controller uses EVENT? to read event code 403.
- The GPIB SRQMASK USER is on and DCL (Device Clear) or SDC (Selected Device Clear) is received at the GPIB port. In this situation, all pending events (including event 403) are discarded. RS-232-C DCL has the same effect (assuming the RS-232-C SRQMASK USER is on).

- The GPIB SRQMASK USER is on and event code 403 is discarded from the GPIB stack. This situation arises when a GPIB controller does not query the GPIB event stack and subsequent DSA events cause the stack to overflow. When event code 403 is discarded, the message **Request for external service ignored** appears on the screen.

  If the RS-232-C SRQMASK USER is on and the above condition appears at the RS-232-C port, the DSA takes the same actions as it did for the GPIB interface.

Events Reported at Instrument Power-On

When the DSA is powered on, diagnostic tests automatically execute (unless bypassed with hardware straps). When diagnostics complete, nondiagnostic firmware in the DSA takes over and the remote interfaces are activated. The DSA then reports power-on status: event 401 (power on) if diagnostics passed or were bypassed, or event 394 if diagnostics failed. Specific information about diagnostic failure can be obtained with the DIAG? query-only command.

Following the power-on status report, the integrity of the DSA nonvolatile RAM (NVRAM) is checked and, if found to be unsatisfactory, one of the following events is reported:

- Event 657 – NVRAM was completely initialized and all stored settings (if any) were discarded. This event is typically reported when the NVRAM battery fails.

Event 658 is typically caused by bad settings being sent to the DSA. In this case, event 658 is reported when the DSA is next powered on.

- Event 658 – This is the same as event 657, except that the following conditions are not initialized from the factory settings: mainframe link of the UID? command, the number of times the DSA has been powered on, and the length of time the DSA has been powered on.

Appendix G: Sampling Rates and Intervals

This appendix contains tables of the sampling rates in samples per second, followed by tables of sampling intervals in seconds, for all combinations of {TBMAIN | TBWIN} TIME and LENGTH.

Sampling Rate (samples-per-second), for LENGTHs 512 to 5120

	LENGTH Values				
TIME	**512**	**1024**	**2048**	**4096**	**5120**
100.0 s	0.5 S	1.0 S	2.0 S	5.0 S	5.0 S
50.0 s	1.0 S	2.0 S	4.0 S	10.0 S	10.0 S
20.0 s	2.5 S	5.0 S	10.0 S	25.0 S	25.0 S
10.0 s	5.0 S	10.0 S	20.0 S	50.0 S	50.0 S
5.0 s	10.0 S	20.0 S	40.0 S	100.0 S	100.0 S
2.0 s	25.0 S	50.0 S	100.0 S	250.0 S	250.0 S
1.0 s	50.0 S	100.0 S	200.0 S	500.0 S	500.0 S
500.0 ms	100.0 S	200.0 S	400.0 S	1.0 kS	1.0 kS
200.0 ms	250.0 S	500.0 S	1.0 kS	2.5 kS	2.5 kS
100.0 ms	500.0 S	1.0 kS	2.0 kS	5.0 kS	5.0 kS
50.0 ms	1.0 kS	2.0 kS	4.0 kS	10.0 kS	10.0 kS
20.0 ms	2.5 kS	5.0 kS	10.0 kS	25.0 kS	25.0 kS
10.0 ms	5.0 kS	10.0 kS	20.0 kS	50.0 kS	50.0 kS
5.0 ms	10.0 kS	20.0 kS	40.0 kS	100.0 kS	100.0 kS
2.0 ms	25.0 kS	50.0 kS	100.0 kS	250.0 kS	250.0 kS
1.0 ms	50.0 kS	100.0 kS	200.0 kS	500.0 kS	500.0 kS
500.0 µs	100.0 kS	200.0 kS	400.0 kS	1.0 MS	1.0 MS
200.0 µs	250.0 kS	500.0 kS	1.0 MS	2.5 MS	2.5 MS
100.0 µs	500.0 kS	1.0 MS	2.0 MS	5.0 MS	5.0 MS
50.0 µs	1.0MS	2.0 MS	4.0 MS	10.0 MS	10.0 MS
40.0 µs	-	-	-	-	-
20.0 µs	2.5 MS	5.0 MS	10.0 MS	25.0 MS	25.0 MS

Sampling Rate (samples-per-second), for LENGTHs 512 to 5120 (Cont.)

	LENGTH Values				
TIME	**512**	**1024**	**2048**	**4096**	**5120**
10.0 µs	5.0 MS	10.0 MS	20.0 MS	50.0 MS	50.0 MS
8.0 µs	-	-	-	-	-
5.0 µs	10.0 MS	20.0 MS	-	100.0 MS	100.0 MS
4.0 µs	-	-	50.0 MS	-	-
2.5 µs	-	-	-	-	-
2.0 µs	25. 0 MS	50.0 MS	100.0 MS	250.0 MS	250.0 MS
1.0 µs	50.0 MS	100.0 MS	-	500.0 MS	500.0 MS
800.0 ns	-	-	250.0 MS	-	-
500.0 ns	100.0 MS	-	-	1.0 GS	1.0 GS
400.0 ns	-	250.0 MS	500.0 MS	-	-
250.0 ns	-	-	-	2.0 GS	2.0 GS
200.0 ns	250.0 MS	500.0 MS	1.0 GS	-	-
100.0 ns	500.0 MS	1.0 GS	2.0 GS	5.0 GS	5.0 GS
50.0 ns	1.0 GS	2.0 GS	4.0 GS	10.0 GS	10.0 GS
25.0 ns	2.0 GS	-	-	-	-
20.0 ns	-	5.0 GS	10.0 GS	25.0 GS	25.0 GS
10.0 ns	5.0 GS	10.0 GS	20.0 GS	50.0 GS	50.0 GS
5.0 ns	10.0 GS	20.0 GS	40.0 GS	100.0 GS	100.0 GS
4.0 ns	-	-	-	-	-
2.0 ns	25.0 GS	50.0 GS	100.0 GS	250.0 GS	250.0 GS
1.0 ns	50.0 GS	100.0 GS	200.0 GS	500.0 GS	500.0 GS
500.0 ps	100.0 GS	200.0 GS	-	1.0 TS	1.0 TS
400.0 ps	-	-	500.0 GS	-	-
200.0 ps	250.0 GS	500.0 GS	1.0 TS	-	-
100.0 ps	500.0 GS	1.0 TS	-	-	-
50.0 ps	1.0 TS	-	-	-	-

Sample Rates (samples-per-second), for LENGTHs 8192 to 32768

	LENGTH Values				
TIME	**8192**	**10240**	**16384**	**20464**	**32768**
100.0 s	10.0 S	10.0 S	20.0 S	20.0 S	50.0 S
50.0 s	20.0 S	20.0 S	40.0 S	40.0 S	100.0 S
20.0 s	50.0 S	50.0 S	100.0 S	100.0 S	250.0 S
10.0 s	100.0 S	100.0 S	200.0 S	200.0 S	500.0 S
5.0 s	200.0 S	200.0 S	400.0 S	400.0 S	1.0 kS
2.0 s	500.0 S	500.0 S	1.0 kS	1.0 kS	2.5 kS
1.0 s	1.0 kS	1.0 kS	2.0 kS	2.0 kS	5.0 kS
500.0 ms	2.0 kS	2.0 kS	4.0 kS	4.0 kS	10.0 kS
200.0 ms	5.0 kS	5.0 kS	10.0 kS	10.0 kS	25.0 kS
100.0 ms	10.0 kS	10.0 kS	20.0 kS	20.0 kS	50.0 kS
50.0 ms	20.0 kS	20.0 kS	40.0 kS	40.0 kS	100.0 kS
20.0 ms	50.0 kS	50.0 kS	100.0 kS	100.0 kS	250.0 kS
10.0 ms	100.0 kS	100.0 kS	200.0 kS	200.0 kS	500.0 kS
5.0 ms	200.0 kS	200.0 kS	400.0 kS	400.0 kS	1.0 MS
2.0 ms	500.0 kS	500.0 kS	1.0 MS	1.0 MS	2.5 MS
1.0 ms	1.0 MS	1.0 MS	2.0 MS	2.0 MS	5.0 MS
500.0 μs	2.0 MS	2.0 MS	4.0 MS	4.0 MS	10.0 MS
200.0 μs	5.0 MS	5.0 MS	10.0 MS	10.0 MS	25.0 MS
100.0 μs	10.0 MS	10.0 MS	20.0 MS	20.0 MS	50.0 MS
50.0 μs	20.0 MS	20.0 MS	-	-	100.0 MS
40.0 μs	-	-	50.0 MS	50.0 MS	-
20.0 μs	50.0 MS	50.0 MS	100.0 MS	100.0 MS	250.0 MS
10.0 μs	100.0 MS	100.0 MS	-	-	500.0 MS
8.0 μs	-	-	250.0 MS	250.0 MS	-

Sample Rates (samples-per-second), for LENGTHs 8192 to 32768 (Cont.)

	LENGTH Values				
TIME	**8192**	**10240**	**16384**	**20464**	**32768**
5.0 µs	-	-	-	-	1.0 GS
4.0 µs	250.0 MS	250.0 MS	500.0 MS	500.0 MS	-
2.5 µs	-	-	-	-	2.0 GS
2.0 µs	500.0 MS	500.0 MS	1.0 GS	1.0 GS	-
1.0 µs	1.0 GS	1.0 GS	2.0 GS	2.0 GS	5.0 GS
800.0 ns	-	-	-	-	-
500.0 ns	2.0 GS	2.0 GS	4.0 GS	4.0 GS	10.0 GS
400.0 ns	-	-	-	-	-
250.0 ns	-	-	-	-	-
200.0 ns	5.0 GS	5.0 GS	10.0 GS	10.0 GS	25.0 GS
100.0 ns	10.0 GS	10.0 GS	20.0 GS	20.0 GS	50.0 GS
50.0 ns	20.0 GS	20.0 GS	40.0 GS	40.0 GS	100.0 GS
25.0 ns	-	-	-	-	-
20.0 ns	50.0 GS	50.0 GS	100.0 GS	100.0 GS	250.0 GS
10.0 ns	100.0 GS	100.0 GS	200.0 GS	200.0 GS	500.0 GS
5.0 ns	200.0 GS	200.0 GS	-	-	1.0 TS
4.0 ns	-	-	500.0 GS	500.0 GS	-
2.0 ns	500.0 GS	500.0 GS	1.0 TS	1.0 TS	-
1.0 ns	1.0 TS	1.0 TS	-	-	-
500.0 ps	-	-	-	-	-
400.0 ps	-	-	-	-	-
200.0 ps	-	-	-	-	-
100.0 ps	-	-	-	-	-
50.0 ps	-	-	-	-	-

Sampling Intervals, for LENGTHs 512 to 5120

	LENGTH Values				
TIME	**512**	**1024**	**2048**	**4096**	**5120**
100.0 s	2.0 s	1.0 s	500.0 ms	200.0 ms	200.0 ms
50.0 s	1.0 s	500.0 ms	250.0 ms	100.0 ms	100.0 ms
20.0 s	400.0 ms	200.0 ms	100.0 ms	40.0 ms	40.0 ms
10.0 s	200.0 ms	100.0 ms	50.0 ms	20.0 ms	20.0 ms
5.0 s	100.0 ms	50.0 ms	25.0 ms	10.0 ms	10.0 ms
2.0 s	40.0 ms	20.0 ms	10.0 ms	4.0 ms	4.0 ms
1.0 s	20.0 ms	10.0 ms	5.0 ms	2.0 ms	2.0 ms
500.0 ms	10.0 ms	5.0 ms	2.5ms	1.0 ms	1.0 ms
200.0 ms	4.0 ms	2.0 ms	1.0 ms	400.0 μs	400.0 μs
100.0 ms	2.0 ms	1.0 ms	500.0 μs	200.0 μs	200.0 μs
50.0 ms	1.0 ms	500.0 μs	250.0 μs	100.0 μs	100.0 μs
20.0 ms	400.0 μs	200.0 μs	100.0 μs	40.0 μs	40.0 μs
10.0 ms	200.0 μs	100.0 μs	50.0 μs	20.0 μs	20.0 μs
5.0 ms	100.0 μs	50.0 μs	25.0 μs	10.0 μs	10.0 μs
2.0 ms	40.0 μs	20.0 μs	10.0 μs	4.0 μs	4.0 μs
1.0 ms	20.0 μs	10.0 μs	5.0 μs	2.0 μs	2.0 μs
500.0 μs	10.0 μs	5.0 μs	2.5μs	1.0 μs	1.0 μs
200.0 μs	4.0 μs	2.0 μs	1.0 μs	400.0 ns	400.0 ns
100.0 μs	2.0 μs	1.0 μs	500.0 ns	200.0 ns	200.0 ns
50.0 μs	1.0 μs	500.0 ns	250.0 ns	100.0 ns	100.0 ns
40.0 μs	-	-	-	-	-
20.0 μs	400.0 ns	200.0 ns	100.0 ns	40.0 ns	40.0 ns
10.0 μs	200.0 ns	100.0 ns	50.0 ns	20.0 ns	20.0 ns
8.0 μs	-	-	-	-	-

Sampling Intervals, for LENGTHs 512 to 5120 (Cont.)

	LENGTH Values				
TIME	512	1024	2048	4096	5120
5.0 μs	100.0 ns	50.0 ns	-	10.0 ns	10.0 ns
4.0 μs	-	-	20.0 ns	-	-
2.5 μs	-	-	-	-	-
2.0 μs	40.0 ns	20.0 ns	10.0 ns	4.0 ns	4.0 ns
1.0 μs	20.0 ns	10.0 ns	-	2.0 ns	2.0 ns
800.0 ns	-	-	4.0 ns	-	-
500.0 ns	10.0 ns	-	-	1.0 ns	1.0 ns
400.0 ns	-	4.0 ns	2.0 ns	-	-
250.0 ns	-	-	-	500.0 ps	500.0 ps
200.0 ns	4.0 ns	2.0 ns	1.0 ns	-	-
100.0 ns	2.0 ns	1.0 ns	500.0 ps	200.0 ps	200.0 ps
50.0 ns	1.0 ns	500.0 ps	250.0 ps	100.0 ps	100.0 ps
25.0 ns	500.0 ps	-	-	-	-
20.0 ns	-	200.0 ps	100.0 ps	40.0 ps	40.0 ps
10.0 ns	200.0 ps	100.0 ps	50.0 ps	20.0 ps	20.0 ps
5.0 ns	100.0 ps	50.0 ps	25.0 ps	10.0 ps	10.0 ps
4.0 ns	-	-	-	-	-
2.0 ns	40.0 ps	20.0 ps	10.0 ps	4.0 ps	4.0 ps
1.0 ns	20.0 ps	10.0 ps	5.0 ps	2.0 ps	2.0 ps
500.0 ps	10.0 ps	5.0 ps	-	1.0 ps	1.0 ps
400.0 ps	-	-	2.0 ps	-	-
200.0 ps	4.0 ps	2.0 ps	1.0 ps	-	-
100.0 ps	2.0 ps	1.0 ps	-	-	-
50.0 ps	1.0 ps	-	-	-	-

Sampling Intervals, for LENGTHs 8192 to 32768

	LENGTH Values				
TIME	**8192**	**10240**	**16384**	**20464**	**32768**
100.0 s	100.0 ms	100.0 ms	50.0 ms	50.0 ms	20.0 ms
50.0 s	50.0 ms	50.0 ms	25.0 ms	25.0 ms	10.0 ms
20.0 s	20.0 ms	20.0 ms	10.0 ms	10.0 ms	4.0 ms
10.0 s	10.0 ms	10.0 ms	5.0 ms	5.0 ms	2.0 ms
5.0 s	5.0 ms	5.0 ms	2.5ms	2.5ms	1.0 ms
2.0 s	2.0 ms	2.0 ms	1.0 ms	1.0 ms	400.0 µs
1.0 s	1.0 ms	1.0 ms	500.0 µs	500.0 µs	200.0 µs
500.0 ms	500.0 µs	500.0 µs	250.0 µs	250.0 µs	100.0 µs
200.0 ms	200.0 µs	200.0 µs	100.0 µs	100.0 µs	40.0 µs
100.0 ms	100.0 µs	100.0 µs	50.0 µs	50.0 µs	20.0 µs
50.0 ms	50.0 µs	50.0 µs	25.0 µs	25.0 µs	10.0 µs
20.0 ms	20.0 µs	20.0 µs	10.0 µs	10.0 µs	4.0 µs
10.0 ms	10.0 µs	10.0 µs	5.0 µs	5.0 µs	2.0 µs
5.0 ms	5.0 µs	5.0 µs	2.5 µs	2.5 µs	1.0 µs
2.0 ms	2.0 µs	2.0 µs	1.0 µs	1.0 µs	400.0 ns
1.0 ms	1.0 µs	1.0 µs	500.0 ns	500.0 ns	200.0 ns
500.0 µs	500.0 ns	500.0 ns	250.0 ns	250.0 ns	100.0 ns
200.0 µs	200.0 ns	200.0 ns	100.0 ns	100.0 ns	40.0 ns
100.0 µs	100.0 ns	100.0 ns	50.0 ns	50.0 ns	20.0 ns
50.0 µs	50.0 ns	50.0 ns	-	-	10.0 ns
40.0 µs	-	-	20.0 ns	20.0 ns	-
20.0 µs	20.0 ns	20.0 ns	10.0 ns	10.0 ns	4.0 ns
10.0 µs	10.0 ns	10.0 ns	-	-	2.0 ns
8.0 µs	-	-	4.0 ns	4.0 ns	-

Sampling Intervals, for LENGTHs 8192 to 32768 (Cont.)

	LENGTH Values				
TIME	**8192**	**10240**	**16384**	**20464**	**32768**
5.0 µs	-	-	-	-	1.0 ns
4.0 µs	4.0 ns	4.0 ns	2.0 ns	2.0 ns	-
2.5 µs	-	-	-	-	500.0 ps
2.0 µs	2.0 ns	2.0 ns	1.0 ns	1.0 ns	-
1.0 µs	1.0 ns	1.0 ns	500.0 ps	500.0 ps	200.0 ps
800.0 ns	-	-	-	-	-
500.0 ns	500.0 ps	500.0 ps	250.0 ps	250.0 ps	100.0 ps
400.0 ns	-	-	-	-	-
250.0 ns	-	-	-	-	-
200.0 ns	200.0 ps	200.0 ps	100.0 ps	100.0 ps	40.0 ps
100.0 ns	100.0 ps	100.0 ps	50.0 ps	50.0 ps	20.0 ps
50.0 ns	50.0 ps	50.0 ps	25.0 ps	25.0 ps	10.0 ps
25.0 ns	-	-	-	-	-
20.0 ns	20.0 ps	20.0 ps	10.0 ps	10.0 ps	4.0 ps
10.0 ns	10.0 ps	10.0 ps	5.0 ps	5.0 ps	2.0 ps
5.0 ns	5.0 ps	5.0 ps	-	-	1.0 ps
4.0 ns	-	-	2.0 ps	2.0 ps	-
2.0 ns	2.0 ps	2.0 ps	1.0 ps	1.0 ps	-
1.0 ns	1.0 ps	1.0 ps	-	-	-
500.0 ps	-	-	-	-	-
400.0 ps	-	-	-	-	-
200.0 ps	-	-	-	-	-
100.0 ps	-	-	-	-	-
50.0 ps	-	-	-	-	-

Glossary

Acquisition
The process of repeatedly sampling the signals coming through input channels, and accumulating the samples into waveforms.

ASCII
Acronym for American Standard Code for Information Interchange. ASCII is a standard eight-bit code used by many computers and data terminals.

Asynchronous
Relating to data transmissions which are not synchronized through a system clock. Also, errors which are not synchronized with a command.

Autoset
A means of letting the DSA set itself to provide a stable and meaningful display of a given waveform.

Averaging
Displaying a waveform that is the combined result of several acquisitions, thereby reducing random noise.

Binary Block
Tektronix-specified format for binary data transmissions: *%,<byte count> <data value> <data value> ... <data value> <checksum>*.

BNF
Acronym for Backus-Naur Form, which is a formal language structure for syntax definition.

Channel
The electrical path from a plug-in amplifier input to the digitizer, i.e., an input corresponding to one of the BNC connectors on a plug-in amplifier. Also, the smallest component of a waveform expression.

Channel Number
An identifier, usually in the form <*slot*> <*ui*>, that distinguishes plug-in unit channels, e.g., "L4."

Checksum
Checksum comparison is a serial communication operation used to verify data accuracy by comparing the sum of data received against a previously computed sum (checksum).

Complex Waveform
A waveform with a waveform expression beyond a single channel specification. Any waveform using a numeric value, a function, a reference to a stored waveform, or an arithmetic operator is a complex waveform. However, using the average function does not make a waveform complex.

Concatenate
To link commands together.

Cursor
Any of four styles of paired markers that you position with the knobs or CURSOR commands. The DSA displays the positions of the cursors and the distance between them, in axis units.

DCE
Acronym for data communications equipment. The DSA is configured as a DCE device as defined in the EIA standard RS-232-C.

Debug Mode
Copies input data from either the GPIB or the RS-232-C interface to the front panel display for program troubleshooting.

Default Measurement Parameter
A value from the default set of measurement parameters. The operator can change the default values. Whenever a waveform is created, the measurement parameters are copied from the default set.

Device-Dependent Message
Messages initiated by a controller that can only be understood by a specific device. The entire command set are the primary device-dependent messages for the DSA.

Distal
The point farthest (most distant) from a reference point. As used in the DSA, the ending measurement point for timing measurements.

DMA
Acronym for direct memory access. DMA capability is a feature available in some controllers that transfers data directly into memory by bypassing the central processing unit (CPU). The DSA comes standard with a GPIB-compatible DMA.

DTE
Acronym for data terminal equipment which is a computer or terminal as defined in EIA standard RS-232-C.

DUT
Acronym for device under test.

EIA
Acronym for Electronics Industries Association.

Enveloping
Displaying a waveform that shows the extremes of variation of several acquisitions.

Escape Character Set
An alternate character set that is accessed by including an ASCII escape character (decimal 27) in front of the appropriate ASCII character.

FFT
Fast Fourier Transform. An algorithm used to convert time domain data to frequency domain data.

FFT Window

A means of modifying time domain data prior to conversion to frequency domain data to reduce frequency leakage, which is caused by discontinuities between the end and the beginning of the time domain data.

Floating Point Value

A type of numeric argument (<*NR2*> or <*NR3*>) that includes a decimal point and may include an exponent.

GPIB

Acronym for General Purpose Interface Bus. The GPIB interface is an eight-bit parallel bus that allows remote computer control of the DSA and other synchronous devices. GPIB characteristics are specified in IEEE STD 488 1978.

Histogram

A representation of the frequency of occurrence of voltage levels where one axis represents the range of voltages and the other axis represents the frequency of occurrence within a single waveform record. The DSA internally uses a histogram of the waveform to determine topline and baseline.

IEEE STD 488

The Institute of Electrical and Electronic Engineers specification for the GPIB interface.

Initialize

Setting the DSA to a completely known set of default conditions.

LIFO

Acronym for the last-in first-out method used to process I/O buffer contents.

LSB

Acronym for least significant bit or least significant byte.

Main
Refers to the primary time base used for acquiring data. See Window.

Measurement
An automated numeric readout that the DSA provides directly from the displayed waveform in real time, without operator intervention.

Measurement Parameter
One of several control/command parameters that the DSA operator can exercise over the automated measurement process.

Measurement Statistics
The accumulation of a history of individual measurement readouts, showing the mean and standard deviation of a selected number of samples.

Measurement Tracking
The process of automatically adjusting the measurement parameters to reflect changes in the waveform.

Mesial
The middle point of a range of points. As used in the DSA, the middle measurement point between proximal and distal points for timing measurements.

MSB
Acronym for most-significant bit or most-significant byte.

Nonvolatile RAM (NVRAM)
Random access memory (RAM) with a battery backup system to prevent the loss of data in case of power failure.

Parse
To decode or analyze data according to a syntax.

Point Accumulate Mode
A mode of operation where the DSA displays newly acquired waveform data points and keeps the previously acquired data points on the screen.

Proximal
The point nearest (in closest proximity) to a reference point. As used in the DSA, the beginning measurement point for timing measurements.

Quoted String
An element of DSA command syntax (<*qstring*>). A quoted string is required by some command arguments and returned as responses to specific queries. The <*qstring*> element is enclosed by quotes and can be any of the characters defined in the DSA character set.

Record Length
The number of samples (data points) that make up a waveform.

RS-232-C
An interface that allows remote operation of the DSA via a controller or terminal. Serial asynchronous data can be transmitted between the DSA and another device as defined in EIA standard RS-232-C.

Sample Interval
The time interval between successive samples in a waveform record.

Scalar
A specific quantity that has magnitude but not direction (a real number, not a vector).

Selected Waveform
The highlighted (brightest) waveform of a multi-waveform display. The selected waveform is the waveform that is acted on by the knobs, menu selectors and commands.

Setting
The state of the system at some given time.

Signed Integer Value
A type of numeric argument (<*NR1*>) which is an integer with a leading sign.

Stored Waveform
A waveform record with attributes that is saved in a dedicated area of memory.

Synchronous
Data transmission in which timing is provided by a clock in the sending unit.

Tektronix Codes and Formats
An shortform title for the Tektronix GPIB Codes, Formats, Conventions, and Features internal standard. The DSA syntax and commands comply with this internal Tektronix standard.

Time Base
The time-dependent specifications that control the acquisition of a waveform. The time base determines when and for how long to acquire and digitize signal data points.

Trace
The visible representation of an input signal or combination of signals. Identical to waveform.

Tracking
The process of automatically adjusting the measurement parameters to reflect changes in the waveform.

Trigger
An electrical event that initiates acquisition of a waveform record and to which time attributes and measurements are referenced.

Truncate

To delete less significant digits from a number. Truncation reduces precision.

Twos-Complement

A representation of negative numbers used by digital computer systems to facilitate arithmetic processing.

Uptime

The number of hours the instrument has been powered on.

Unsigned Integer

A type of numeric argument (<*ui*>) which is an integer without a leading sign (e.g., TRACE <*ui*> or TRACE3).

Waveform

The visible representation of an input signal or combination of signals. Identical to trace.

Waveform Expression

The definition of what the waveform displays. It can include one or more channels combined arithmetically and modified by functions.

Waveform Number

A number assigned by the DSA to identify a waveform. Display waveforms are numbered 1 through 8. A new waveform is always given the lowest available number.

Waveform Preamble

A response returned from the WFMPRE? query that contains the scaling information for the waveform. A waveform preamble consists of the WFMPRE header followed by preamble arguments. All preamble data are ASCII encoded.

Window
Data acquired using a secondary time base with a higher sample rate (and therefore higher resolution) than the Main time base. (See Main.) A waveform that represents a horizontally expanded portion of another waveform.

XY Waveform
A waveform where both horizontal and vertical position of the data points reflect signal data.

Yt Waveform
A waveform where the vertical position of the waveform data points reflects signal data, and the horizontal position of the waveform data points reflects time.

Index