TEK OPERATORS MANUAL

070-6614-00
Product Group 37

2432 PROGRAMMERS REFERENCE GUIDE

Please Check for
CHANGE INFORMATION
at the Rear of This Manual

First Printing JUNE 1987

Tektronix®
COMMITTED TO EXCELLENCE

Printed in U.S.A. Specification and price change privileges are reserved.

INSTRUMENT SERIAL NUMBERS

Each instrument has a serial number on a panel insert, tag, or stamped on the chassis. The first number or letter designates the country of manufacture. The last five digits of the serial number are assigned sequentially and are unique to each instrument. Those manufactured in the United States have six unique digits. The country of manufacture is identified as follows:

B000000	Tektronix, Inc., Beaverton, Oregon, USA
100000	Tektronix Guernsey, Ltd., Channel Islands
200000	Tektronix United Kingdom, Ltd., London
300000	Sony/Tektronix, Japan
700000	Tektronix Holland, NV, Heerenveen, The Netherlands

Contents

Page

C Event Tables

D GPIB Concepts

Illustrations

Figure *Page*

Tables

Table *Page*

1

Introduction

Introduction

The Tektronix 2432 Digital Oscilloscope is a portable, dual-channel instrument with a maximum digitizing rate of 100 megasamples per second. The programmable features of the 2432 allow you to perform a wide variety of automatic testing activities. All of the testing capabilities of the 2432 can be controlled remotely via the GPIB.

How to Use this Manual

This manual is designed to help test system programmers interface the Tektronix 2432 Digital Oscilloscope with GPIB systems. For a more general overview of 2432 operation please refer to the operator's manual. Here are a few of the major topics which this manual will discuss:

Establishing GPIB Communication—This section shows how to set up the 2432 for GPIB operation and how to communicate with a controller.

How to Write Programs for the 2432—This section explains basic command syntax, types of commands, and various programming features of the 2432.

Debugging Your Programs—This section gives hints for fixing your 2432 programs if they have bugs and explains how to use the 2432 to help debug other portions of your GPIB program.

Command Tables—Appendix A gives syntax definitions and brief explanations of all GPIB commands for the 2432.

Waveform Transfers—Appendix B explains how to transfer waveform data to or from the 2432's internal storage areas and what that data actually means.

GPIB Introduction—Appendix D gives a brief overview of basic GPIB theory.

Other Appendices—These provide various information useful in 2432 programming such as SRQ's, power-up states, etc.

Tips for New Users

If you have never worked with GPIB devices we recommend that you start by reading the GPIB Concepts (Appendix D), then come back and read the main sections of this manual. Give special attention to the sections on how to write programs for the 2432 and debugging the programs you've written. Also, use the HELP feature of the 2432 to learn about any areas that you might have questions about. To activate HELP, push the STATUS/HELP button below the CRT and then push the menu button labeled HELP.

2

Establishing GPIB Communication

Establishing GPIB Communication

The GPIB is the communications link between the 2432 and your controller. The first thing to do is make sure a functional GPIB cable is connected from the 2432 to the controller. Then set the 2432 GPIB parameters so they fit the 2432 into your GPIB system. The GPIB parameters include the mode, terminator, and address and are all accessed from the front panel of the 2432. This section discusses how to set up these various GPIB parameters.

Setting the GPIB Mode

The 2432 can be set up to be a talker, a listener, or both. It can also be taken "off bus". In addition, the 2432 can output waveform data to printers and plotters. Any of these different functions are selected by changing the GPIB Mode of the 2432. In general, you will set the Mode to be talker/listener. This allows the 2432 to accept commands from a controller over the GPIB and to send answers (in response to queries) back to the controller. The 2432's GPIB Mode cannot be changed by remote programming.

To make the mode selection you first need to get to the MODE menu. This is done by pressing the OUTPUT front panel button and then pushing the SETUP menu button followed by the MODE menu button. Once the MODE menu comes up there are several possible selections. These selections are Talk-Only, Listen-Only, Talk-Listen, Devices, and Off-Bus. We will now look at each of these options.

Talk-Only

Pushing the T/ONLY selection puts the 2432 into the Talk-Only GPIB mode. In this mode the 2432 assumes that it is always addressed to talk, and will neither respond to commands nor assert the SRQ line. This mode is useful for configuring simple systems with one Talk-Only device (the 2432) connected to one or more Listen-Only devices (such as a printer, another oscilloscope, or another 2432). When you press the T/ONLY selection, the front-panel ADDR indicator should light, and the T/ONLY submenu should appear on the screen. The T/ONLY sub-menu lets you select what data the 2432 will send when it talks. All three talk-only options (CURVE, WAVFRM, and SEND PRGM) require at least one listener to be present on the GPIB bus. If there is no listener present on the bus at the time the instrument is ready to transmit the 2432 will display an error message on its CRT and refuse to transmit. The three (mutually exclusive) options available in this menu are:

CURVE. With this selection the 2432 will send only the curve portion of waveform data (as it would in response to a CURVE? query). After making this selection you may start transmission by pressing the OUTPUT front panel button followed by the TRANSMIT menu button. The 2432 will then transmit the CURVE portion of all displayed waveforms to all listeners on the bus.

WAVFRM. With this selection the 2432 will send both the preamble and the curve data (as it would in response to a WAVFRM? query). After making this selection you may start transmission by pressing the OUTPUT front panel button followed by the TRANSMIT menu button. The 2432 will then transmit both the PREAMBLE and CURVE portions of all displayed waveforms to all listeners on the bus.

SEND PRGM. With this selection the 2432 will transmit a sequencer program. The 2432 will send whichever sequencer program is currently selected within the RECALL menu (press the PRGM front panel button followed by the RECALL menu button). To transmit the sequencer program simply press the SEND PRGM button, followed by the OUTPUT button, followed by the SENDPRGM button.

Listen-Only

Pushing the L/ONLY menu selection button places the 2432 into the Listen-Only mode. The front-panel ADDR indicator should light. In this mode the 2432 acts only as a listener (LADS). The user may send setup data (previously queried from the 2432 or constructed independently) to the 2432 to implement a "canned" test setup. When in Listen-Only mode the 2432 can only accept commands over the GPIB, it cannot send any messages or assert SRQ. This mode can be used when sending data from one 2432 to another. The 2432 that is in listen only mode will be the one data is sent "to".

Talk-Listen

Pushing the T/L menu selection button places the 2432 into the Talk-Listen mode. In this mode the 2432 acts as both a Talker and a Listener. This is the "normal" GPIB mode, and the mode most people will use for day to day testing activities. In this mode the 2432 can accept data and commands and reply to queries.

Devices

Pushing the DEVICES button brings up a menu which allows you to tell the 2432 what type of hard copy device (if any) you are using. The 2432 can format waveform information for several types of output devices. The devices supported are: some devices that accept HPGL (Hewlett Packard Graphics Language), and the HP Thinkjet printer (model HP-2225A).

To actually make a printout, first connect the device to the 2432 and make sure the device is in Listen Always mode. Make sure all other GPIB instruments are disconnected since we will be operating in a controller- less system. To start the printout, just press the 2432 OUTPUT front-panel button, followed by the PRINT menu button. While the 2432 is printing the PRINT button becomes an ABORT button. Press this button at any time to terminate the printout. To disable the printing capability of the 2432, you must return to the MODE menu and select another mode. "Making Printer or Plotter Copies" in this manual talks about setting up for different plotters and printers.

Off Bus

Pushing the OFF BUS selection makes the 2432 bus-transparent. When the 2432 is OFF BUS the TRANSMIT selection disappears from the OUTPUT menu. You may alter certain GPIB parameters (TERM, ADDR, and ENCDG) but these selections will not take effect until a GPIB Mode other than OFF BUS is selected.

Setting the GPIB Message Terminator

To get to the GPIB Terminator menu press the front panel OUTPUT button, followed by the SETUP menu button, followed by the TERM menu button. Within the GPIB Terminator menu the options are either EOI or LF/EOI.

Selecting EOI in the TERM menu causes the End or Identify (EOI) line to be asserted simultaneously with the last byte of the message. LF/EOI causes Carriage-Return (CR) and Line-Feed (LF) characters to be added to the end of each message with EOI asserted simultaneously with LF.

Which terminator should you choose?

The 2432 always recognizes EOI as a message terminator, no matter what byte it is asserted with. You may also program the 2432 to recognize a line feed as a terminator. This is useful when using a controller which sends a line-feed character as the last byte of a message instead of asserting EOI with the last byte of a message.

When the 2432 is set up to recognize the line-feed character as the terminator, it will assume that an incoming message is terminated if it detects either EOI or a line-feed character. On output, the 2432 will send a carriage return followed by a line feed with EOI asserted as the last bytes of each message. The 2432 always asserts EOI along with the last byte of a message, no matter what terminator has been selected.

A potential ambiguity exists if you select the line-feed terminator for use with either BINARY waveform transfers or LLSET. The ambiguity occurs because the line-feed character is a valid number in binary waveform and LLSET strings. This could confuse the controller into prematurely assuming an end of message. When using a controller that does not recognize EOI only as the message terminator, it is best to use ASCII format for waveform transfers and SET?.

Setting the GPIB Primary Address

To get to the GPIB Address menu press the front panel OUTPUT button, followed by the SETUP menu button, followed by the ADDR menu button. This menu allows you to increment or decrement to any GPIB address. Pushing the ↑ increments the talk and listen address by one. Pushing the ↓ decrements the address. The selected GPIB address establishes both primary talk and listen addresses for the 2432. It can be set to any value from 0 to 30, inclusive.

Sample Controller Program

This short program allows you to send commands or queries to the 2432 and prints responses from the 2432. Though the program is written in 4041 BASIC, it illustrates basic principles that should carry over to other languages.

```
10 scope=1
20 !===============================
30 ! first enter the 2432's address and then open the
40 ! hardware port to talk to the GPIB; finally enable
50 ! the srq interrupt and point to the handler routine
60 !===============================
70 print " Enter 2432 address:" ;
80 input addr
90 if addr>30 then goto 70
100 if addr<0 then goto 70
110 open #scope:" gpib0(pri=" &str$(addr)&" ,eom=<0>):"
120 dim answr$ to 7000
130 dim comm$ to 200
140 enable srq
150 on srq then gosub 350
160 !===============================
170 ! Now get the command or query to send. A query
180 ! has a " ?"  right after the header. If it is a
190 ! query then talk the 2432 and print the response.
200 !===============================
210 wait .25
220 print " Command?  " ;
230 input comm$
240 print #scope:comm$
250 query = pos(comm$," ?" ,1)
260 if not(query)  then goto 210
270 input  #scope:answr$
280 print " 2432 response is: " &answr$
290 goto 210
300 !===============================
310 ! This is the code that is executed when an srq
320 ! happens. We poll the bus and print the status
330 ! byte and the corresponding event query response.
340 !===============================
350 poll stb,dev
360 print #scope:" path off;event?"
370 input #scope:event
380 print " srq received: status = " ;stb,"  event = " ;event
390 print #scope:" path on"
400 resume
410 end
```

Debugging the Interface When it Doesn't Work

This section helps you debug the interface when you can't talk to the 2432 with your controller. First and foremost, make sure the GPIB cable is securely connected to both the 2432 and the controller. Any doubts about the cable? If so, get another one to make sure it is not causing the problem.

Next, go look at the GPIB status menu. To get there simply push the OUTPUT button on the front panel and then the STATUS (left most) menu button. This brings up a screen full of information showing how the GPIB portion of the 2432 is set up. Now make sure the mode is set to T/L (both talk and listen), and the terminator and address are set up correctly. Look at the section on Establishing Communication if you need to change any of these GPIB parameters.

Also, make sure that the debug mode is not "paused". A message is printed near the top of the status screen when debug is paused. If you don't see a message debug mode is all right. When debug is "paused" the 2432 will not listen to any bytes from the bus, so this could be the problem. To fix it, turn pause off by pushing the OUTPUT button, then the DEBUG menu button, and then turn debug off by pushing the DEBUG ON\OFF menu button.

GPIB Status Menu

Now a word about the GPIB status menu. This is a very useful menu to refer to when you have questions about how the GPIB is set up. There are a few areas on this menu screen that deserve more explanation. This menu is the only place you can find out what the Fast Transmit status is. This is because as soon as the Fast Transmit mode is turned on the 2432 will only talk waveform data and will not respond to queries.

The BINWFM to SCOPE section (on the lower left of the menu screen) indicates how the 2432 will interpret binary data sent to it. The DATA ENCDG status (just above BINWFM) shows format of binary data the 2432 will send out. For more information see Appendix B on waveforms.

The events column (on the right of the menu screen) shows the first 7 (of 10) events contained in the 2432's event buffer. The topmost event number will be returned in response to the next EVENT? query. If the letters SRQ appear to the right of an event number then that event number had an SRQ associated with it. The instrument was serial polled to handle the service request but the corresponding event code was not requested. If an EVENT? query is sent to the 2432, the event code pointed to will be returned.

What Do the GPIB Status LED's Mean?

The 3 lights over the 2432 CRT give an indication of GPIB activity. Look at these LED'S if a steady state problem arises (such as dead front-panel controls). An explanation of the LED's meaning follows.

LOCK. If this LED is lit the the 2432 will not respond to any front panel controls. The 2432 itself will lock out the front panel while doing a self calibration, an AUTOsetup, and during certain portions of setup sequences. The LOCK LED will light during any of these operations. The controller can lock the front panel in one of two ways. First, it can send the 'LOCK ON' command to the 2432. Or it can send the 'LOCK LLO' command to the 2432 and follow that with the GPIB universal command LLO. The LLO universal command is an IEEE-488 command while in Appendix A.

SRQ. This LED is lit while the 2432 is asserting the SRQ GPIB hardware line to request service. When the controller performs a serial poll this LED will go out.

ADDR. This LED is lit when the 2432 is addressed to talk or listen or, more specifically, when it is in either the TADS or LADS state.

3

How To Write Programs

How to Write Programs for the 2432

This section describes how to send commands to and get responses from the 2432. A few guidelines can help make your programming task easier:

1. If possible, use full command and argument names. This enhances readability and makes upgrading easier (new releases of firmware may have slightly different abbreviations, usually longer since there are more commands).
2. Try to write instrument specific software in a modular fashion. Use subroutines, if possible. This makes it easier to modify your program when something beyond your control changes.

Sending a Command to the 2432

To change the 2432 operating state you must send it a command. For example, to change the volts per division value to 5 volts send the command, 'CH1 VOLTS:5'.

A message is composed of one or more commands separated by semicolons and ending with the message terminator. Except where noted, carriage returns, spaces, and tabs are ignored by the 2432 when receiving a message. In this section you will learn what a command is made of and how to compose and terminate messages.

Building Commands

A command consists of a header and arguments. Building a command is like climbing a tree, you start at the trunk and move up different branches. Finally, you end up at the end of some branch and you cannot go any further. Some trees have lots of branches, some only have a few. There are many different "trees" you can climb to tell the 2432 what to do. Here are three examples of 2432 commands for you to refer to as we discuss what headers and arguments are.

```
1. RUN SAVE
2. CH1 VOLTS:5,POSITION:−1
3. CURSOR XPOS:ONE:2.5
```

HEADERS. All commands have, at the very least, a header. In the examples above RUN, CH1, and CURSOR are all headers. The header can be thought of as the trunk of a tree made up of a bunch of similar setups for the 2432. For instance, by using the CURSOR header (see example 3) the 2432 will know that you want to change a cursor position and not the vertical position. Some headers have no arguments (branches) beneath them and define the entire command by themselves. MANTRIG is an example of this type of header.

ARGUMENTS. Some commands require the addition of arguments (branches) to the header to describe exactly what the 2432 is to do. You will use an argument if you need to follow the "tree" farther than just the header to completely define the command. In the first example above, sending the header RUN cannot tell the 2432 whether to actively acquire or go to save mode. Therefore the argument SAVE is added to the header to indicate that the 2432 should be in save mode. To add an argument to a header just separate the header from the argument by one or more spaces.

In some cases one argument level below the header is not enough to completely define what the 2432 is to do. So we add another level of arguments (called link arguments), and separate them from the first argument with a colon. We can keep adding link arguments to the command until the action is completely defined. Both the second and third examples contain link arguments with the second containing just 1 level of link (the 5 and −1) and the third containing 2 levels (ONE:2.5).

Multiple arguments can be included with the same header by separating each argument with a comma (,). The second example shows both the VOLTS and POSITION arguments (along with their associated link arguments) attached to the same header by being separated with commas.

Numeric arguments use the ANSI X3.42 standard format. This format states that there are three types of numbers; integers, reals, and reals with exponents (these are called NR1, NR2, and NR3 respectively). Each type of number is composed of ASCII digits with the most significant digit sent first. Any of these three number types is acceptable whenever a numeric argument is required. Here are some examples of each of the three number types:

NR1	375,	0,	−23
NR2	+12.589,	1.37592,	−00037.5
NR3	−1.51E+03,	+51.2E−07,	+00.0E+00

Composing Messages

Complete messages are constructed by stringing one or more individual commands together. Multiple commands within a message are separated by a semicolon. For example, we can combine the first two examples above into one long message by inserting a semicolon between them like this: RUN SAVE;CH1 VOLTS:5,POSITION:−1. Extra spaces may be included to increase readability:

```
RUN SAVE;  CH1 VOLTS: 5, POSITION: −1
```

Terminating Messages

Both talking and listening devices must agree on how to end a message. Obvious difficulties can arise when talker and listener don't agree. For example, if the listener thinks the message has ended too soon, it garbles part of the message. On the other hand if the listener doesn't think the message has ended when it actually has, it "hangs" the bus waiting for a message that will never come.

There are two common methods for ending a message. Some controllers assert EOI concurrently with the last data byte; others use only the line-feed character as a terminator. You need to determine which type of controller you are using and make sure that the 2432's terminator is set appropriately.

If you select EOI as terminator the 2432 will interpret any data byte received with EOI asserted as the end of the input message. The 2432 will assert EOI simultaneously with the last data byte of an output message.

If you select the LF character as terminator the 2432 will interpret the LF character without EOI asserted (or any data byte received with EOI asserted) as the end of an input message. The 2432 will transmit a carriage return followed by a line-feed (LF) with EOI asserted to terminate output messages.

Asking the 2432 a Question

There are many times when a controller needs to obtain information about the operating state of the 2432. To do this the controller sends a "query" to the 2432 to ask it a question. Queries consist of a header followed by a question mark. For example, to find out whether the 2432 is currently acquiring or in save mode, simply send the query 'RUN?' to the 2432. If the response message is 'RUN ACQUIRE', then the 2432 is acquiring. If the response is 'RUN SAVE', then the 2432 is in save mode.

You may query the 2432 for any headers listed in Appendix A, except those specified as "COMMAND ONLY".

The query header may be followed by an argument to further specify the type of response desired. For example, the query 'CH1?' causes a response containing all of the information about that channel. Whereas the query 'CH1? POSITION' obtains only the position information. In general, any query can be further specified using arguments, up to one level above the "bottom" of the command tree (where the "bottom" of the tree is the argument in the rightmost column of Appendix A tables).

Using PATH Mode

When you issue a query, you know what type of result should be returned. For example, on a 'CH1? POSITION' query you expect a number that represents the CH1 vertical position. With PATH set to ON, a typical answer to this query would be 'CH1 POSITION:2'. The "CH1 POSITION:" portion of this response is called the "path". To get at the number you're after (the "2") you must strip off the "path" portion. You can eliminate this extra processing by setting PATH to OFF before sending the query to the 2432. When PATH is OFF the 2432 will not send the "path" portion of the response. So with PATH set to OFF the answer to the query 'CH1? POSITION' would be 2. This value can be read directly into a numeric variable used by your program.

If you are querying the 2432 to obtain a setup which you wish to send back to the 2432 (to recreate that setup) be sure to set PATH to ON. Without the "path" portion the 2432 will be unable to interpret the command. In the above example, returning the string 'CH1 POSITION:2' to the 2432 would set the CH1 position to 2 divisions above center screen. Returning only the string '2' would result in confusion.

Using LONG Mode

Headers and arguments can be used at full length, or they can be abbreviated to reduce typing and bus traffic. The command tables in Appendix A show all essential characters in bold uppercase with the optional characters in lowercase. If LONG is ON the 2432 will send the full representation for each header and argument contained in a query response. If LONG is OFF only the essential characters (the abbreviations) are returned, resulting in a shorter response string. LONG does not affect commands being sent to the 2432. The addition of new features to the 2432 may result in a longer abbreviation for some headers or arguments in future firmware releases. Therefore we recommend that you set LONG to ON if you wish to run your programs with future versions of 2432 firmware.

How to Use Service Requests (SRQs)

The 2432 can issue a Service Request (SRQ) to interrupt the controller and let it know that something "interesting" has happened. To issue an SRQ the 2432 simply asserts the SRQ bus management line on the GPIB (see Appendix D for more information on the GPIB). The SRQ indicator LED, above the display screen, will be on while the 2432 is asserting SRQ.

When a controller detects an SRQ it performs a serial poll of each instrument currently connected to the GPIB. When an instrument is serial polled it returns a status byte to the controller. If bit 7 (of 8) of the status byte is set, then this is the instrument that asserted the SRQ. Whenever the 2432 sends a status byte it will indicate, in a general way, the reason for the SRQ. For example, a status byte with a value of 97 indicates that the 2432 encountered a command error while interpreting the last command string sent by the controller. An 'EVENT?' query sent to the 2432 at this time will return a number that contains more information about what happened. In the example above, the event code might be a 156 which indicates that one of the symbols (headers and or arguments) making up the command was mis-typed and the 2432 could not make sense of it.

When the 2432 has an event to report, it first looks to see if it is already trying to assert SRQ for a previous event. If it is, then the new event is saved in a backup position. This way, when the controller eventually gets around to reading the first status byte, the backup byte will become active and try to assert SRQ. The 2432 only saves 2 events which are capable of asserting SRQ at any one time. When these 2 "slots" are full, subsequent events (up to 8) are placed into an event buffer. If the event buffer becomes full then the oldest stored event is dropped and the current one is entered. The event number returned by an EVENT? query is determined by first returning the events associated with the two SRQ 'slots' and if those are unfilled, the contents of the event buffer are returned starting with the newest entry. If you send an EVENT? query and the number returned is 459, the 2432 is currently asserting SRQ on the bus and will insist that the controller poll the 2432 for its status byte before returning an event code. This is done to make sure that event codes and status bytes correspond. Because of hardware constraints, once the 2432 asserts SRQ it cannot change the status byte that will be returned by the next serial poll. If the next meaningful event code were to be returned instead of the 459 event, subsequent EVENT? queries would not correspond with the correct status bytes.

The tables in Appendix C show the different status bytes and event codes that can be returned by the 2432.

Programming With SRQs

The first thing you need to start programming with SRQs is an SRQ handling routine. This routine will be called when your controller detects an SRQ on the GPIB. This subroutine polls the instrument asserting the SRQ (optionally extracting the event code from the instrument) and takes appropriate action. A simple SRQ handling routine is included within the sample program in the "Establishing GPIB Communications" section of this manual.

Next you need to determine what kind of information you will need for your application. The 2432 allows the programmer to select the type of happening that will assert an SRQ. For example, sending the command 'CER ON' to the 2432 will result in command errors asserting SRQ. If you wish to keep the 2432 from asserting any SRQs, send the command 'RQS OFF'. For a complete list of commands of this type refer to the "Service Request Commands" section of Appendix A.

One common use for SRQs is to have the 2432 assert SRQ when it has finished a task. For example, you might have the 2432 assert SRQ when it completes a single-sequence operation. For a complete list of 'tasks' that the 2432 can report with an SRQ see the "Operation Complete Event Codes" in Appendix C. Let's walk through some of the steps a controller program would go through in order to wait until the 2432 has finished an acquisition.

1. Turn this type of SRQ on by sending the 'OPC ON' command to the 2432.
2. Clear out all previous events and SRQs by sending the command 'INIT SRQ'.
3. Set the 2432 control states appropriately to make the desired measurement.
4. Make sure the trigger mode is "single sequence" by sending the command 'ATRIGGER MODE:SGLSEQ'.
5. Tell the 2432 to start acquiring by sending the command 'RUN ACQUIRE'.

At this point the controller will wait for the SRQ from the 2432. To end this wait, the SRQ interrupt handler routine you have written for your controller needs to report the receipt of the SRQ back to the main program. When this happens the program can continue.

In summary, the programmer needs to do several things to handle SRQs efficiently:

1. Set up the 2432 SRQ mask correctly. Make sure that the type of SRQ you are expecting is enabled.
2. Use a controller that can recognize or sense when SRQ is asserted and then perform a serial poll.
3. Have an SRQ handler routine that gathers the status byte and event code and communicates this information to the main body of the controlling program.

Programming Without SRQs

There are some situations in which SRQ's will not be used. For example, the programmer may not wish to use SRQs, or the controller may be unable to detect SRQs or to perform a serial poll. In such cases, some other approach is needed to find out the status of the 2432. (Actually, programming without SRQs is simpler to understand and to implement, but it does not allow the controller to perform other tasks while the 2432 is completing its task.) Before you begin programming without SRQs you must first send the 'RQS OFF' command to make sure the 2432 will not generate any SRQs.

When SRQs are not used, the status of the 2432 is determined by sending the EVENT? query. First clear out all the current events and SRQs which might be present by sending an 'INIT SRQ' command to the 2432. Then have the controller set the 2432 to the desired operational state and wait for the 2432 to finish. During this waiting period the controller repeatedly sends an EVENT? query to the 2432 until an event code associated with the current task is returned. See the "Operation Complete Events" section in Appendix C for a list of event codes. As long as the 2432 has nothing to report, it will return EVENT 0 in response to the EVENT? query. An EVENT query must be used because the 2432 will not update the status byte if RQS is OFF. The 2432 does not know when a status byte is read by a controller unless SRQ is asserted with the status byte. The 2432 never changes the current status byte because it does not know when the controller is ready for a new status byte. This is similar to the situation discussed earlier when event number 459 is returned.

Here is an example scenario summarizing what we've just described. Let's say that the controller has set the 2432 into single-sequence mode. The controller must wait until the sequence is complete. During this wait the controller sends EVENT? queries to the 2432 until the response changes from EVENT 0 to EVENT 461.

There are other ways to determine that the 2432 has done what you requested without using SRQs or events. For example, one solution to the single sequence "problem" is to inquire about the RUN state by sending a 'RUN?' query to the scope. While the 2432 is acquiring, the response will be RUN ACQUIRE. When the single sequence operation is complete, the 2432 enters Save mode and the response will change to RUN SAVE.

Making Measurements With the 2432

The 2432 waveform parameter extraction (WPE) feature provides a convenient method for making measurements on waveforms. When combined with Auto Setup, WPE provides a method to set up for and characterize unknown signals. Commands are provided to control how the 2432 extracts, calculates, and displays the waveform parameters, and for setting up the "window" and determining the base and top levels.

To familiarize yourself with the operation of the WPE features, see the "Controls, Connectors, and Indicators" section of the Operators Manual which gives a description of front-panel operation. Appendix C of the Operators Manual describes how measurements are actually calculated.

The 2432 also provides waveform data commands that allow you to make custom measurements when the measurement you need is not already available as one of the built in WPE measurements. Commands are available to specify the waveform of interest, the part of the waveform you wish to look at, the minimum and maximum values, the average value, and the crossing points. You can combine these various commands and queries to make many specialized measurements.

Using the WPE Feature

Three types of GPIB commands are provided for WPE operations. The first type of commands control HOW the 2432 calculates the waveform parameters. The second type of commands control WHAT parameters are calculated and returned via GPIB. The third type provides GPIB control of the screen DISPLAY of extracted waveform parameters. For a list of measurements see the "Measurement Commands" in Appendix A.

In addition to the commands, a significant number of error and warning messages are available to tell whether the measurement is valid and whether the confidence level of the particular measurement is high or low. Refer to "How to Use Service Requests" and to Appendix C for more information on error reporting.

DEFINING THE MEASUREMENT WINDOW. The "window" defines what portion of the waveform is used when extracting and calculating parameters. If the window function is turned off or the time cursors are not displayed, then any selected parameter will be extracted from the entire targeted waveform. Only that part of the waveform between the two time cursors will be used for making measurements when both the window function and time cursors are on. For example, to create a window on CH1 between points 200 and 500 send the following commands to the 2432:

```
'MEASUREMENT WINDOW:ON'
'CURSOR FUNCTION:TIME,TARGET:CH1,UNITS:BASE'
'CURSOR TPOS:ONE:200,TPOS:TWO:500'
```

DETERMINING BASE AND TOP LEVELS. Time-related parameters are calculated from where the targeted waveform crosses certain user definable levels or thresholds. If any of the thresholds are defined as a percentage, then the base (0 percent) and top (100 percent) levels must first be determined. This is performed according to the user-selected Measurement Method.

The Min-Max method (the factory and init-panel defaults) sets the base level to the minimum waveform value found in the active record. The top level is set to the maximum value found in the waveform.

The Cursor method uses the values of the VOLTS cursors as the base and top values. The lower cursor becomes the base and the upper cursor the top. Although volts and time cursors may not be simultaneously selected, Cursor Method may still be used in conjunction with the Window feature. First, set the base and top levels for the Cursor Method using the volts cursors. Next, switch to the time cursors with Window on and set the time cursors to the desired sub-section of the waveform.

The Histogram method builds a histogram of each point in the targeted waveform and sets the base level to the most common lower level and the top level to the most common upper level. For a description of how histograms are created see the corresponding appendix in the Operator's Manual.

DETERMINING TIME REFERENCE CROSSING LEVELS. The time reference locations are found by searching for waveform crossings at the Proximal (near the base level), Mesial (near the middle), and Distal (near the top level) voltage levels. The Proximal, Mesial, and Distal levels (initially set to 10, 50, and 90 percent, respectively) may be set individually, to percentages from Base to Top, or to absolute voltage levels. For example, to set the Mesial level to 45 percent send the following command to the 2432:

```
'MEASUREMENT MESIAL:UNITS:PERCENT,MESIAL:PLEVEL:50'
```

WHAT IS SENT TO THE CONTROLLER. Once the calculation has been set up, the desired parameter may be extracted using the VALUE? Query. The target of the parameter extraction is set using the DATA SOURCE command. See the list of "Automatic Feature Commands" in Appendix A for the various parameters which may be extracted using the VALUE? query. For example, to make a frequency measurement on CH1 send the following command and query to the 2432:

```
'DATA SOURCE:CH1;VALUE? FREQUENCY'
```

MEASUREMENT DISPLAY. Up to four parameters may be selected for display on-screen. Visual voltage threshold crossing indicators ("Marks") are available for time measurements. For example, to cause a frequency measurement of CH1 to appear on screen and have "Marks" placed on the crossing points of CH1 send the following commands to the 2432:

```
'MEASUREMENT DISPLAY:ON'
'MEASUREMENT MARK:ON'
'MEASUREMENT ONE:TYPE:FREQUENCY,ONE:SOURCE:CH1'
```

ERROR AND WARNING CONDITIONS. If the 2432 cannot calculate the requested parameter a value of "99e99" is returned. Events may then be queried, by issuing an 'EVENT?' to the 2432, to determine why the the parameter could not be calculated. If the parameter value is not 99e99, warning conditions may still have occurred. If you are programming using SRQs and the EXW mask is enabled, the warning will generate an SRQ. If you are programming without SRQs then you may want to use 'EVENT?' following each 'VALUE?' to detect any warning conditions. Warnings are issued, for example, if a waveform was of low vertical amplitude so that vertical resolution was compromised, or if there were not enough points on an edge so that horizontal resolution was compromised.

EXAMPLE MEASUREMENT PROGRAM. The following 4041 Basic program will automatically set the 2432 to acquire and display an unknown signal, optimize it for a rise time measurement, and return the rise time value. A wait loop is provided to continue querying the rise time while the waveform fills, as indicated by a return value of 99e99 and an event code of 269. With a fast rising edge the 2432 will be set into its repetitive sampling mode and will take about one second to fill at 5 ns/div.

If event codes other than 269 are returned, the event (an error or a warning) is printed out and the program stops. The 2432 is assumed to be at address 1 for this example.

```
100 Scope=1
110 Dim resp$ to 30
120 Dim event$ to 9
130 Open #scope:" gpib0(pri=1,eom=<0>):"
135 Print #scope:" rqs off;init srq"
140 Print #scope:" autoset mode:rise"
150 Print #scope:" autoset resolution:hi"
160 Print #scope:" autoset execute"
170 Print #scope:" measurement method:histogram"
180 Print #scope:" data source:ch1"
190 Print #scope:" value? rise"
200 Input #scope:resp$
210 If pos(resp$," 99e99" ,1) = 0 then goto 260
220 Input #scope prompt " event?" :event$
230 If event$<>" event 269" then goto 280
240 Print " waiting for fill"
250 goto 190
260 print resp$
270 goto 290
280 print event$
290 end
```

Making Custom Measurements

Waveform data commands and queries can be used to extract both voltage and timing information from a particular area of a waveform. This is useful when the automatic measurements described above do not return a useful answer because the algorithm used is not specific enough. Using waveform data commands and queries allows you to extract pertinent data about the waveform quickly and use that information in the controller. An example program which uses these commands and queries to count the number of pulses between the time cursors is found in Appendix J.

SELECTING THE WAVEFORM. The waveform data commands and queries operate on only one waveform at a time. This waveform is selected with the DATA SOURCE command. For example, to extract data from the CH1 waveform send the 'DATA SOURCE:CH1' command to the 2432.

The segment of the targeted waveform that is analyzed when making measurements is designated by the START and STOP commands. This is similar to the "window" feature of WPE. For example, to look at the part of the waveform between points 200 and 500 send the command 'START 200;STOP 500'. To include the whole waveform send 'START 1;STOP 1024'.

MAKING VOLTAGE MEASUREMENTS. To obtain the minimum voltage (for the segment of the waveform specified by START and STOP) send the command VMINIMUM? to the 2432. To obtain the maximum or average voltage send VMAXIMUM? or VAVG?, respectively. Or, to obtain the same parameters in terms of "digitizing levels" use the commands MINIMUM?, MAXIMUM?, or AVG?.

MAKING TIMING MEASUREMENTS. Timing measurements involve measuring the time between various points on the waveform. These "points" are selected by determining where the waveform crosses some arbitrary voltage level. This voltage level is user selected with the LEVEL command. To find out where a crossing occurs within the present window, send the PCROSS (for a positive-going crossing) or the NCROSS command (for a negative-going crossing).

Now, as an example, let's walk through a pulse width measurement using this technique. The pulse width is the time between the crossing of the 50% point on the rising edge and the crossing of the 50% point on the falling edge. Here are the basic steps to determine pulse width:

1. Use VMAXIMUM? and VMINIMUM? to determine the full amplitude of the signal.
2. Set LEVEL to the 50% voltage of the pulse.
3. Use PCROSS? to find the first waveform point whose value is at or above the 50% level (positive-going).
4. Use NCROSS to find the first waveform point whose value is at or below the 50% level (negative-going).
5. Calculate the pulse-width time based upon the number of points between PCROSS and NCROSS and the present time per division setting. (There are 50 points per division.)

The HYSTERESIS command provides an added feature for use with the PCROSS and NCROSS commands. The Hysteresis command specifies the number of digitizing levels below LEVEL (for PCROSS) or above LEVEL (for NCROSS) the waveform must go before a valid crossing is recognized. You may set the hysteresis level with the command 'HYSTERESIS #', where "#" is the hysteresis value. Hysteresis acts like a "noise filter" which insures that small variations in the waveform pattern will not be mistaken as actual crossing points.

Here is an example of how HYSTERESIS affects PCROSS. Let's say that your signal ranges from 100 digitizing levels to −20 digitizing levels. If you set the crossing level to 50 and set hysteresis to 5, the 2432 will start searching for a location that is at 45 digitizing levels or below (that's 5 below the crossing level). After finding this location, the 2432 will proceed forward to the first location that is at 50 digitizing levels or above. This location will be returned as the PCROSS value.

Using the same example, but changing crossing level to −15 and the hysteresis to 10, the 2432 would never find a crossing since it would first search for a location that was at −25 digitizing levels. Since the signal only ranges from 100 to −20 such a point does not exist. In this case the 2432 will return a value of 0 in response to PCROSS?.

The DIRECTION command allows you to choose the direction the search for the crossings will follow. To search from "left" to "right" in the data use PLUS. To search backwards in time select MINUS.

How to Use the 2432 Sequencing Capabilities

The 2432 can save and recall multiple front-panel setups along with associated instrument actions. Each setup and associated set of user specified actions is called a "step". One or more steps can be put together to form a sequence. Sequences may be run by the 2432, sent back to the controller, or sent to another 2432. Any valid front-panel setup can be saved in a step. The setup may be prepared using the instrument front panel or loaded via the GPIB. A setup may include the selection of parameter measurements and Save-On-Delta.

Individual front panels can be saved using the same procedure as outlined for entire sequences. The only difference is that a sequence used for storing just one front panel has only one step.

Constructing a Sequence

Sequences are built out of "steps". Each step consists of one front-panel setup with some associated "actions" which will be performed when the step is recalled. Saving a sequence involves setting up the 2432 as desired for the first step, specifying what actions are to take place on this step, and then issuing the GPIB command to save the sequence. Subsequent steps are added to the sequence in the same manner. Let's walk through these steps one at a time.

SETTING UP THE FRONT PANEL. You may set up the front panel using GPIB commands just as you would in a normal controller program. The 2432 should be set to the desired operating mode including all measurements, autoset modes, and output device specifications. This operating mode makes up the front panel. Lines 5 and 6 of the screen are also considered part of the front panel which is saved into the sequencer. You may use these lines to display special messages during this step. For example, you could tell the user what type of test is running, or what stage a test is at. Use the MESSAGE command during the set up procedure to specify what message will appear on the screen with each step.

WHAT ARE ACTIONS? Each step has a set of actions that will be done when that step in the sequence is run. Some of these actions occur before the front panel is loaded while others are run after. For example, Self-Test and Self-Cal are actions which, if selected, will be done before the front panel is loaded to make sure the 2432 is healthy. The actions Autoset, Measurements, Print/plot, Bell, SRQ, and Pause are executed after the front panel is loaded.

Each action has been assigned a number. After selecting the actions which are desired, add up their assigned numbers and send the sum to the 2432 with the command 'SETUP ACTIONS:sum'. If no actions are desired then send 'SETUP ACTIONS:0'. For example, if the Bell and Pause action are needed then the command to send is 'SETUP ACTION:160' (32 + 128 = 160).

Here is the definition of the various actions:

Repeat (1) After the last step of a sequence, control is transferred to the most recent (highest numbered) step which has repeat enabled. This forms an infinitely repeating loop.

Self-Cal (2) Self calibration is performed on the 2432 prior to loading the programmed front-panel settings. If self calibration fails, the sequence is aborted and the extended diagnostics menu is displayed.

Self-Test (4) Self diagnostics is performed on the 2432 prior to loading the programmed front-panel settings. If self-diagnostics fail, the sequence is aborted and the extended diagnostics menu is displayed.

Autoset (8) An autoset will be performed immediately after setting up the saved front panel. The type of autoset will depend upon the mode selected in the saved front panel.

Print/Plot (16) Turning on the print/plot action causes data to be sent to the currently selected external printing or plotting device. The data printed (waveforms, measurements etc.) depends upon the selections made in the saved front panel. If there is no device on line this action is ignored.

Bell (32) The internal 2432 bell will ring at the end of the step. A step is considered complete after front panel setups have been changed, an acquisition has been made, measurements done, and data output to any selected devices.

SRQ (64) If OPC is ON, an SRQ is generated at the end of each step with this action set. If this step is the last step in a sequence then the event code will indicate sequence complete otherwise it will show the end of a step. See "Operation Complete Events" in Appendix C for event code numbers.

Pause (128) After the completion of all other actions, the 2432 will pause and wait for a step command before going to the next step. The step command may come from the GPIB ('STEP'), the front panel PRGM button, or the rear panel Sequencer Input. Pause does not occur on the last step in a sequence unless a repeat loop has been programmed.

Protect (256) If protect is set on the first step of a sequence, the sequence is protected from accidental deletion. Setting protect on other steps has no effect. To delete a protected sequence it is necessary to edit the first step and change the protection, or to override the protection by sending a "SETUP FORCE:ON" GPIB command.

The activity of each action and its relationship to previous and successive actions is automatically synchronized by the 2432. Self-Cal and Self-Test actions are always completed before loading the instrument setup. The Autoset action is always completed before any data is acquired. An acquisition cycle including averaging and repet mode filling is completed before any measurements are made or before the Print/Plot, Bell, SRQ, or Pause actions are done. This means that any measurement or waveform observed at the end of a step occurred as a result of that step.

SAVING THE SEQUENCE. After the actions have been set and the 2432 is in the desired operating mode, you may save the sequence step by sending the 'SETUP SAVE:"seq name"' command. This will save the current front panel in a sequence named "seq name". For example, to save the current front panel in a sequence named "TEST1" you would send the command 'SETUP SAVE:"TEST1"' to the 2432. The name can be up to 6 characters in length and consist of any uppercase alpha-numeric character including blanks. Leading blanks are significant and are stored as part of the name while trailing blanks are ignored.

If the sequence name currently exists then the 2432 appends the current front panel onto the existing sequence as a new step. Long sequences are built one step at a time. If the name does not exist, a new sequence is created with the current front panel as the first step.

Five special names: "ONE", "TWO", "THREE", "FOUR", "FIVE" are reserved for saving only single step sequences with no actions. This is to maintain compatibility with previous 2430 instruments. Saving a step with one of these reserved names will replace an existing step rather than being appended to it. These names may be used without quotes just as on previous 2430 instruments. For example, 'SETUP SAVE:"ONE"' is equivalent to 'SETUP SAVE:ONE'.

The size of the selection menu limits the number of sequences to 40 because only 40 names can be displayed on screen at once. The size of the Sequencer memory limits the total number of steps which make up those sequences. Each front-panel setup is "compressed" before becoming a new step in a sequence. Different front-panel setups have different sizes ranging from 12 to 200 bytes in length, so as many as 800 and as few as 100 steps can be stored. To find out how many bytes of memory are available to the sequencer send a 'SETUP? MEMORY' query. To obtain a list of the names of all currently saved sequences, use the 'SETUP? NAMES' query.

Recalling a Sequence

Once you have a sequence safely stored away in the 2432, the next thing to do is run that sequence. To run a sequence send the command 'SETUP RECALL:"seq name"', where "seq name" is the name of the existing sequence you want to recall. When you issue a recall command, the 2432 will find the desired sequence, load the first step, and begin executing the actions associated with the first step. See "What Are Actions" for more information on what each action does. If the 2432 cannot find the named sequence that you are recalling, it will issue an SRQ, and ignore the command.

Removing Unwanted Sequences

There are two ways to delete sequences. An individual sequence can be deleted by sending the 'SETUP DELETE:"seq name"' command to the 2432. This will remove the sequence called "seq name". All sequences currently saved in the 2432 may be deleted by sending the 'SETUP CLEAR' command.

Any sequence can be protected from deletion by setting the protect action on the first step of the sequence. Neither of the above methods of deleting sequences will remove a "protected" sequence unless specifically told to do so. If you wish to delete a "protected" sequence you can override the protection by sending the 'SETUP FORCE:ON' command to the 2432. This will allow any sequence to be removed regardless of its "protected" status. This override remains in effect until turned off by sending the 'SETUP FORCE:OFF' command. The override only effects sequence deletions from the GPIB. The front panel user is still unable to delete a protected sequence. Remember, deleted sequences are gone forever.

Using the Sequencer

The sequencer in the 2432 can be used to store many front-panel states for recall under controller command. The sequencer can also do many tests by itself without the controller once the sequencer has been programmed. In both cases the overall test time is reduced because the time it takes to set up the 2432 becomes much less significant.

CONTROLLING THE SEQUENCER. Several methods of maintaining synchronization between the 2432's sequencer and the controller are provided. These let the controller know what state the sequencer is in at any given time so that the total test can be managed, as well as allowing the controller to command the sequencer.

At the conclusion of each sequencer step which has an SRQ action set, an SRQ is generated to let the controller know what stage the test is at. For instance, the "step complete SRQ" would be helpful when using the Pause action on a step to allow a technician to change an external test condition before allowing the test to continue. When the technician is finished, he would indicate this to the controller which would continue the sequence.

At the end of each step the sequencer automatically goes on to the next step in the sequence unless the Pause action is set. In this case the sequencer will wait until it receives a command from the front panel, the GPIB, or the rear-panel BNCs to continue. From the GPIB, the controller would send the 'STEP' command to get the sequencer to continue. To stop a currently executing sequence, the controller may send a 'HALT' command. This will stop any acquisition in progress and terminate the sequence at the end of the current step. For more information on the rear-panel BNCs or the front-panel controls see the operators manual.

HOW CAN THE SEQUENCER HELP? The sequencer of the 2432 helps automate stand alone testing and increases test system throughput by reducing bus traffic between the controller and the 2432. In many cases, the controller can be eliminated from the system to create a portable tester for field work. It can also speed up stand alone testing by removing the need for you to change settings; you just select the next step. And even when you still need the controller, the sequencer helps in a number of ways. It can store many front panel states so a new test setup is just a matter of proceeding to the next step. Just sending the SAVE RECALL command to the 2432 is enough to invoke a completely new setup. The sequencer also manages each acquisition cycle and will proceed to the next "action" only when the oscilloscope is ready. This relieves the controller of constantly monitoring the 2432 for intraprocess updates.

Sending and Receiving Sequences

The 2432 can send stored sequences to a controller, to a printer, or to another 2432. The controller can request the 2432 to send (in either ASCII or binary) any sequence that is currently stored. One sequence at a time can be sent to a printer or another 2432, but the sequence must be sent in ASCII. This feature is useful for making hard copies of a sequence or updating another 2432 in your test system with the latest sequences. There are two basic modes for transferring sequences, binary and ASCII.

BINARY TRANSFERS. Binary transfers are done using the LLPRGM command. This command transfers sequences in a compact low level form. This method is preferred for high speed transfer in a production environment. To read a sequence from the 2432 send the 'LLPRGM? "seq name"' query to the 2432. This will return a binary block of data that makes up the sequence named in the "seq name" part of the query. If the sequence name is not specified, a block containing all the sequences currently stored in the 2432 will be returned.

To send a sequence or sequences to the 2432, send the data block (as is) returned by the LLPRGM query back to the 2432. When sequences are entered, their names are checked against existing sequences in the instrument. Sequences with the same name will not be replaced unless the sequence is first deleted from the 2432 memory.

The binary format used by the 2432 in compressing its sequences is dependent on the firmware version of the instrument that the sequence originally came from. This format can change. If it does, binary format sequences from older firmware versions will not work with instruments with newer firmware versions. We recommended that you save an ASCII version of the sequence as an "archive" so that a new binary format block can be created in the future.

ASCII TRANSFERS. ASCII transfers are done using the PRGM command. ASCII transfers of sequences are essentially the high level commands necessary to set up the sequence steps. Because of this, ASCII transfers are readable and upward compatible with other 2430 series instruments. ASCII transfers are preferred for purposes of documenting or archiving sequences.

To read a sequence using an ASCII transfer send the 'PRGM? "seq name"' query to the 2432. The 2432 will return an ASCII block of information detailing the sequence whose name is "seq name". If no name is specified all sequences will be returned. For each step in the named sequence, approximately 2500 bytes of information are returned. This is very similar in length to the response of a 'SET?' query. To send this sequence back to the 2432, just return the data block.

To improve readability of ASCII sequence blocks a formatting command has been provided. If formatting is turned on, the sequence output will contain carriage returns, line feeds, and extra spacing to make it more readable. To request a formatted version of the ASCII sequence output send the 'FORMAT ON' command to the 2432. To discontinue formatting send the 'FORMAT OFF' command.

NOTE

Controllers which terminate input on line feeds will be unable to read formatted output.

SENDING SEQUENCES TO A PRINTER OR ANOTHER 2432. This is very similar to the "talk-only" mode of the 2432 except that a formatted ASCII sequence is sent instead of a waveform. The device to which this information is sent may be either a printer or another 2432.

To configure the 2432 to send sequences first push the OUTPUT front panel button followed by the SETUP menu button. Then push the MODE menu button followed by the T/ONLY menu button and finally make your selection by pushing the SEND PRGM menu button.

If you are sending sequences to a printer, first configure the printer to listen always mode (see "Making Printer or Plotter Copies" for help). If the "to" device is not a printer but another 2432 then the "to" 2432 must be configured to listen always mode before any sequences can be sent. This is done just like configuring the "from" 2432 above except that instead of pushing the T/ONLY menu button push the L/ONLY menu button and you are done.

Select which sequence you would like to send just as if you are recalling that sequence. First, push the PRGM front panel button. Then, push the RECALL menu button and use the up and down arrow keys to underline the chosen sequence. Thats it.

Now make sure there is a GPIB cable between the "from" 2432 and the "to" device (printer or 2432). To begin the transfer, push the OUTPUT front-panel button and then the SENDPRGM menu button.

Making Printer or Plotter Copies

The 2432 can print hard copies of its waveforms on a Hewlett Packard Thinkjet printer and some HPGL plotters such as the Tektronix HC100 plotter and the Hewlett Packard HP7470A. (HPGL is an abbreviation for Hewlett Packard Graphics Language. Most Hewlett Packard plotters and many plotters made by other vendors accept HPGL.) Printing may be initiated from the front panel or by sending a PRINT command over the GPIB. When instructed to print the 2432 will format its screen and then send the formatted data to the selected device. A Thinkjet printout takes about 75 seconds, while a plotter can take from 1 to 5 minutes (depending on what is being plotted and the type of plotter being used).

In this section we discuss how to make a hard copy (with or without a controller) and introduce a special mode for use with the HC100 plotter.

Making Copies Without a Controller

There are several steps to making a print or plot without a controller. You will need to know how to set up the hard copy device, how to configure the 2432 to make the copy, how to connect the two together, and how to start the copy process.

SETTING UP THE PRINTER/PLOTTER. The basic idea is to make the hard copy device a "listener". This is normally the controller's responsibility, but since we are operating without a controller this must be done by the user. Different devices may have different methods of setting the listen always mode. We'll use the Thinkjet and HC100 as examples.

Many devices, including the Thinkjet and HC100, have a DIP (dual in-line package) switch which sets their operating mode. Five of the switches are used to set the device address, the rest are used to specify device functions. Many devices, such as the Thinkjet, use one of the non-address switches to activate the listen-always mode. Other devices go into listen-always mode if all 5 of their address switches are set to "1". This would result in a GPIB address of 31. But since the GPIB does not define address 31, many devices use this setting to specify the listen-always mode. The HC100 operates this way. Here is how the DIP switch settings should look for the Thinkjet and HC100 for listen-always mode:

	8 7 6 5 4 3 2 1	
Thinkjet	x x x x x 1 x	x = don't care
		0 = Off
HC100	1 1 1 1 1 0 0 0	1 = On

NOTE

In making these changes, note that many devices "read" their address DIP switches only at power-up. To make sure a device is actually in the state you've set you should turn the device off, change the DIP switches to the new mode and then turn the device back on.

CONFIGURING THE 2432. There are two steps involved in setting the 2432 to print or plot. Step one is selecting which hard copy device you are using. Step two is selecting what you wish to copy. To make either selection first push the OUTPUT front panel button and then push the SETUP menu button. Then push the MODE menu button followed by the DEVICES menu button. Pushing the DEVICES menu button should turn on the ADDR indicator LED. This shows that the 2432 is in "talk-only" mode, ready to talk to the selected hard copy device.

To select the type of device, push the menu button labeled HPGL PLOTTER or the button labeled THINKJET PRINTER.

To select what gets printed, push the SETUP menu button and "compose" the copy. For further information on these selections read the help text on this subject. (To get help, push the STATUS/HELP front-panel button, then push the OUTPUT front-panel button.)

CONNECTING THE 2432 AND THE PRINTER. The 2432 and the printer/plotter should be the only things connected to the GPIB when a controller is not being used. Connect a GPIB cable from the 2432 to the hard copy device and you are ready to print.

STARTING THE COPY. To initiate a copy, push the OUTPUT front-panel button, and then push the far right menu button. This button will be labeled PRINT if you selected the Thinkjet printer or PLOT if you selected the HPGL plotter mode. After you have pushed the PRINT or PLOT button, the button label will change to ABORT. Push this button if you want to stop the print or plot after it has already started. If an error message appears on screen at this point, you've either configured the printer/plotter incorrectly, or the cable has fallen off. Check for these two potential problems.

Making Copies With a Controller

A controller can also initiate a print or plot. Using a controller you may print the Help text, GPIB and status screens, and the measurement Snapshot display. We will now walk through the same setup steps as we went through to make a print/plot without a controller.

SETTING THE PRINTER/PLOTTER. Since a controller is present on the bus, we must assign an address to the printer or plotter. This allows the controller to make the printer or plotter a "listener" so that it can receive and print the hard copy from the 2432. The address is assigned in the same way as above (for printing with no controller) except that the DIP switch settings are different. Here are example switch settings to assign the device an address of 5. You may use whatever address is appropriate for your system.

	8	7	6	5	4	3	2	1	
Thinkjet		1	0	1	0	0	0	x	x = don't care
									0 = Off
HC100	1	0	1	0	0	0	0	0	1 = On

CONFIGURING THE 2432. If you are already using a controller to talk to the 2432 then the GPIB setup of the 2432 will not change for printing/plotting. If you need to start from scratch see the section on changing GPIB Mode in "Establishing GPIB Communication" on how to set the 2432 to the correct address and to Talk/Listen mode.

The only other configuration information which must be selected is which printer or plotter format the 2432 should send. The format is selected by sending 'DEVICE TYPE:HPGL' or 'DEVICE TYPE:THINKJET' to the 2432. The DEVICE command also allows you to compose the print in much the same way as the DEVICES menu button (see "Making Copies Without a Controller").

CONNECTING THE 2432 AND THE PRINTER. Make sure there is a GPIB path between the 2432 and the printer/plotter. (If both instruments can communicate with the controller, this path exists.)

STARTING THE COPY. There are 3 things the controller must do to start a hard copy output from the 2432. First, the 'PRINT' command must be sent to the 2432 (to let it know what to send out the next time it is made a talker). Second, the printer/plotter must be "listen addressed". Third, the 2432 must be "talk addressed". As the 2432 sends data, the printer/plotter should receive that data and begin printing/plotting.

Here is a sample program written in 4041 Basic which initiates a hard copy to a Thinkjet printer. Statements 130 and 140 actually initiate the print by sending the 'PRINT' command, turning the Thinkjet printer into a listener and the 2432 into a talker. Line 120 tells the 2432 to send data for the graticule and waveform, but not to send data for plot text or settings.

```
10  scope=1
20  print " Enter 2432 address:" ;
30  input addr
40  if addr>30 then goto 30
50  if addr<0 then goto 30
60  open #scope:" gpib0(pri=" &str$(addr)&" ,eom=<0>):"
70  print " Enter printer address:" ;
80  input prntaddr
90  if prntaddr>30 then goto 80
100 if prntaddr<0 then goto 80
110 print #scope:" device type:thinkjet"
120 print #scope:" device settings:off,grat:on,text:off,wavfrm:on"
130 print #scope:" print"
140 wbyte atn(unt,unl,prntaddr+32,addr+64)
150 end
```

KNOWING WHEN A COPY IS FINISHED. There are several methods for determining whether the print or plot has completed. At the conclusion of the hard copy the 2432 will issue an Operation Complete SRQ with an associated event code of 463 (OPC SRQs are enabled with the command 'OPC ON'). Another method is to serial poll the 2432 and check the BUSY bit in the status byte. The 2432 sets this bit while it is doing the hard copy and clears it when it is done. See "Using SRQs and Events" for more information on the status byte and Operation Complete SRQs. The last method is to time the hard copy and insert a wait of that length in your program. This method can be dangerous since the hard copy time can vary depending on the amount of data sent.

Once the print or plot has started, the controller can abort it by sending a Device Clear (DCL) to the 2432.

Save-On-Delta Hard Copies

When used without a controller, the 2432 can automatically initiate a hard copy when a Save-On-Delta event occurs. This provides a useful "babysitting" mode which produces a printed record of what has happened. This is possible because the 2432 brings the "delta" event to center screen before going to Save mode. This means that the event you need a copy of is always on screen before the hard copy starts. To select this mode set the 2432 up for a print or plot without a controller present. Then each time a Save-On-Delta event occurs, the 2432 will automatically make a hard copy, time-stamp it, and re-arm Save-On-Delta.

Special HC100 Mode

In addition to understanding HPGL, the HC100 plotter can plot Positive Integer Entire Binary Format waveforms directly (see Appendix B for more information on waveform format). In this case the formatting is done by the plotter, not by the 2432. This mode has two advantages. First, it plots all 1024 points (20 divisions) of a waveform instead of just the 512 points (10 divisions) contained on the screen that the other modes print. Second, it can be faster because the plotter does its own formatting. This reduces bus traffic considerably since fewer bytes are sent over the bus and the HC100's input buffer can hold most of the bytes. This frees the 2432 for other tasks.

For instance, it takes over 2 minutes to plot one waveform using the HC100 in HPGL mode. During this time the 2432 is dedicated only to this task. On the other hand, the plot time for one waveform using the Positive Integer waveform format is about 2.5 minutes, but the 2432 only sends data to the HC100 for about 35 seconds. Therefore, the 2432 is free for other tasks after only 35 seconds, instead of 2 minutes.

To utilize this special HC100 plotter mode make sure the plotter is working correctly by following the above procedures for an HPGL plotter; set up the configuration (controller or no controller) as you wish. Then turn the power off and change the DIP switch settings as shown below before turning the power back on:

8	7	6	5	4	3	2	1
a	a	a	a	a	1	1	0

where: 0 = Off
1 = On
a = leave alone

To make a special mode HC100 plot without a controller just follow these steps.

1. Set the 2432 up to send the Waveform Preamble with a waveform by pushing the OUTPUT front panel button then the SETUP menu button. Push the MODE menu button followed by the T/ONLY menu button. Finally, select the preamble mode by pushing the W/WFMPRE menu button.
2. Select Positive Integer data format by pushing the OUTPUT front panel button and then the SETUP menu button. Push the ENCDG menu button and make the format selection by pushing the RP menu button listed under the WHOLE WFMS header (the second from the left).
3. Start the plot by pushing the OUTPUT front panel button followed by the TRANSMIT menu button.

If you are using a controller you must modify the sample program above by replacing lines 110 through 130 (delete line 120) with the following statements:

```
110 print #scope:" data encdg:rpbinary"
130 print #scope:" wavfrm?"
```

This will cause the 2432 to send both the preamble and curve data in the correct format and start the plot.

How to Save Instrument Settings

This section explains various methods for saving and restoring 2432 operational settings. When using a controller, there are two methods for saving and restoring settings. Without a controller, settings may be saved using the built-in sequencer. In addition, the 2432 can restore one of two user selectable front panel setups at power-up. Let's walk through these various save/restore methods.

Power-up Default Setups

You may select one of two setups as power-up default. The options are either the LAST power-down front panel, or the factory INIT front panel. To select either of these simply push the EXTENDED FUNCTIONS front-panel button, the SYSTEM menu button, the PANEL menu button, and finally the LAST¦INIT menu button.

Using a Controller

Using a controller to store front panel settings involves commanding the 2432 to send a data string representing its front panel setup, and then storing that string in controller memory. The 2432 can send this string in two different formats. One format is ASCII, which looks like a series of regular 2432 commands. The other format is a low level binary representation which is not human readable. This binary format is much shorter than the ASCII string.

Because future versions of firmware for the 2432 may contain different "feature sets", the abbreviations for some symbols (and the binary version of the front-panel state) may vary between versions. For this reason, we recommend that you store an ASCII format "archive" of each front-panel state sent to the 2432 during the course of your controller program. To do this simply send 'LONG ON;SET?' to the 2432, then query the 2432 and save its response.

ASCII FRONT PANEL SETUP. The SET? query is used to obtain the ASCII form of the 2432's front-panel state. The string returned by the 2432 will be about 2300 bytes in length if LONG is ON, or about 1750 bytes if LONG is OFF. Use this query when you want a human-readable copy of the current settings. ASCII format also allows you to alter small portions of the command string before sending it back to the 2432. The SET? query is the most time consuming method for saving a front-panel state but it is also the safest in terms of compatibility with newer instruments.

You should use SET? when:

1. Archiving front panel settings (set LONG to ON).
2. Loading, editing, and then restoring settings.
3. Loading settings from one 2432 and restoring them to another 2432 with different options or firmware versions.
4. Loading and restoring settings when time is not critical.
5. Loading and restoring settings using a controller that only understands the <LF> character as the end of message indicator.

BINARY FRONT PANEL SETUP. You use the 'LLSET?' query to extract a binary version of the 2432's front panel. LLSET stands for Low-Level-SET. The response to a LLSET? query is a binary block with the same structure as the "Entire Binary Waveform Format" detailed in Appendix B. This block is about 275 bytes long and is a packed representation of the current front-panel state of the 2432. Because this block is directly translated into a 2432 hardware configuration, any error in the order or values of the data can have expensive consequences. To insure that front panel data is in the correct format, the 2432 attaches an internal checksum byte to the block which is checked when the block is returned to the 2432. If there is a difference, an SRQ is returned and the block is discarded. The complete block must be used just as it was originally sent by the 2432.

When speed is important or when memory space on your controller is at a premium, use the LLSET? response instead of the SET? response. The 2432 sends and interprets the LLSET? response much faster than the the SET? response. This is the main advantage. However, there are two potential problems to watch out for. First, since the line-feed (LF) character can be a valid byte in an LLSET? response, controllers which interpret the LF as end of message should not use LLSET?. In this case use the SET? query instead. Secondly, the LLSET? response string may change with different firmware versions of the 2432. If you decide to use this method, keep a backup copy of the front panel in the SET? query format.

Using the Sequencer

The last method of saving and recalling front-panel setups of the 2432 uses the built in sequencer. This is the fastest way to recall front panel setups. This method is particularly attractive if you have time to load setups at the start of a test, but want them to run quickly once the test is underway. (The initial loading phase of the test may be unnecessary since the sequencer stores front-panel setups in non-volatile memory so they are not lost when power is turned off.)

The first step in using the sequencer is to create a sequence which contains the front-panel state you wish to save. You may do this from the front panel (use the HELP mode for assistance) or from the controller. From the controller, set up the 2432 with the settings you wish to save and then issue the command 'SETUP SAVE:"name"' to the 2432. The 2432 will store the present front panel setup into a sequence called "name".

Now, retrieve the sequence you've just created and store it in the controller so it may be loaded when the test first begins. To do this issue a 'PRGM? "seq name"' query and store the ASCII string which comes back.

To reload the "name"ed sequence into the 2432's sequencer memory, simply send the ASCII string (as is) back to the 2432. It takes around 8 seconds to reload the sequence into the 2432 sequencer. Once the sequence is in the 2432's sequencer memory it takes less than 1 second to be recalled. This is done using the 'SETUP RECALL:"seq name"' command. For more information, see the section "Using the Sequencer" as well as the "Sequencer Commands" section of Appendix A.

Using Group Execute Trigger

When the 2432 receives a Group Execute Trigger (GET) it executes a task that has been predefined using the DT command. The advantage of using GET over just sending the same defined set of commands to the 2432 is that GET speeds the execution time of the selected task. This is because the 2432 does not have to read and interpret the command before preforming it. It already knows what to do. See Appendix D for an explanation of GET.

To use the GET function, first determine what you want the 2432 to do upon receipt of the GET, then issue the appropriate DT command. For example, let's say you want the 2432 to set up for a Save-On-Delta operation, and then start acquiring. To do this simply send the command 'DT SODRUN' to the 2432. Then, when you need this action to occur, just issue the GET command. Refer to manuals for your controller to learn how to cause your controller to send a GET. See "Miscellaneous Commands" in Appendix A for more information on the DT command.

4

Debugging Your Programs

Debugging Your Programs

The 2432 has a DEBUG mode to help users make their GPIB programs run. You may use this feature to display all messages sent over the GPIB, or only messages sent to the 2432. The messages are scrolled horizontally across the 2432 display. You may control the scrolling speed. You may also 'freeze' the screen at any time to analyze a message.

Note that Appendix G contains the answers to many common programming questions. The problem you are trying to solve may be discussed there.

Familiarization

In order to familiarize yourself with the various debug modes, we encourage you to write a very short program (for your controller) which sends just one or two commands to the 2432 over and over again. This will give you something with which to experiment.

Next, push the OUTPUT front-panel button followed by the DEBUG menu button. You will then see the DEBUG menu displayed on the lower 3 lines of the screen. Push the far left button so that "ON" is underlined. DEBUG mode is now active.

Interacting With the Display

To monitor messages sent to the 2432 push the second button from the left until "SCOPE" is underlined. Now, any mistake in an input message will cause an error message to be printed on screen with an arrow showing where the error is. When an error occurs, the 2432 will automatically "pause" the message to allow you to see what the mistake is. To start the message scrolling again, press the PAUSE menu button. You may stop the scrolling at any time by pressing the PAUSE menu button, it acts as a toggle switch.

You may observe both incoming and outgoing messages. The type of message is selected by pushing the third menu button from the left. While IN is underlined, only messages being sent to the 2432 are displayed. While OUT is underlined, both incoming AND outgoing messages will be displayed. Outgoing messages (responses from queries) are underlined while they scroll across the screen.

Characters which appear on screen are the 2432's version of the ASCII character set. To translate between the two character sets use the conversion charts in Appendix F. For example, an incoming carriage return is shown as a lower-case "n". Any lower-case letters in incoming messages will be changed into upper-case. Whenever the message terminator is found (whether line-feed or EOI) it will be changed into a "box" character before it is displayed.

Character Update Rate

You can select the rate at which characters appear on screen by pushing the fourth menu button from the left. While SLOW is underlined the update rate is reduced to allow you time to read the text. Push the PAUSE button to 'freeze' the message on screen. You may then take as much time as you need to analyze the message.

One word of caution when using either the SLOW or the PAUSE modes. Most controllers have a message timeout in their GPIB driver. If this timeout period is exceeded, the controller will think something is wrong with the 2432 and abort the message prematurely. To prevent this problem, increase your controller timeout value or remove it altogether whenever using the Debug Mode.

Using the 2432 as a Bus Monitor

You can use the 2432 to monitor all GPIB traffic by pushing the second button from the left until BUS is underlined. The 2432 must be in Listen Only mode before the transition to BUS can be made. If you get an error message while attempting to make the BUS selection, go to the section on setting up the 2432 and change the mode to Listen Only.

In BUS monitor mode the 2432 accepts and displays characters WITHOUT interpreting them. You could use this feature to help debug any controller program which sends out GPIB messages. (Even a program that controls devices other than the 2432!)

To use the 2432 as a bus monitor, connect it to the GPIB cable. All messages sent over the bus will be displayed on the screen. The SLOW and PAUSE selections will work in BUS monitor mode also but the same caution about the controller timeout discussed above applies. The IN/OUT selection has no effect while monitoring bus traffic.

Appendix A

GPIB Commands

NOTE

Throughout this appendix, headers and arguments are listed in a combination of bold uppercase and nonbold lowercase letters. The instrument accepts any abbreviated header or argument containing at least all the characters shown in bold uppercase. Any characters added to the abbreviated (uppercase) version must be those shown in lowercase. For a query, the question mark (?) must immediately follow the header. Link arguments shown in brackets ([]) are defaults. In any command that has a default, omitting the link argument sets the default. Do not send the brackets as part of the argument. For example, 'RUN ACQUIRE' and 'RUN' are equivalent.

Vertical Commands

Header	Argument	Argument	Description
CH1(or **CH2**)	**VOL**ts	<NR3>	Sets the "screen" Volts/Div in a 1-2-5 sequence to the value of the argument. The screen Volts/Div takes attached probes into account. For example, sending a 50 V/div setting in over the bus while a 100X probe is attached will result in setting the actual hardware gain to 0.5 V/div. <NR3> will be rounded and limited to a legal hardware setting. If **EXW** is **ON** and rounding or limiting occurs, a warning SRQ will be issued. Live expansion of non-Average waveforms is not allowed. A command to change to 1 mV, 500 μV and 200 μV will be rejected unless acquiring in Average mode. Also, if expanding in Save Mode with Volts/Div setting less than 2 mV, it will be reset to 2 mV when Acquire Mode is selected. Each of these events causes a warning SRQ to be issued if **EXW** is **ON**.

EXAMPLES

Query	**Response**
CH1? VOLTS	CH1 VOLTS:1
CH1? FIFTY	CH1 FIFTY:OFF

Vertical Commands (cont)

Header	Argument	Argument	Description
CH1(or **CH2**) (cont)	**VAR**iable	<NR3>	<NR3> will range from 0 to 100 with a resolution of 0.125. Zero is fully calibrated and 100 is fully uncalibrated. <NR3> will be limited to legal values, and if **EXW** is **ON**, a warning SRQ will be issued. This is not a calibrated feature and is meant to be used as a reference only.
	POSition	<NR3>	<NR3> ranges from -10 to +10 with a resolution of 0.01 and will set the position of ground (in divisions) with "0" being center screen. If <NR3> is limited to the valid range, and **EXW** is **ON**, a warning SRQ will be issued.
	COUpling	**AC** **DC** **GND**	
	FIFty	**[ON]** **OFF**	An SRQ will be issued whenever a 50 Ω overload is detected if **INR** is **ON**. If **EXR** is **ON**, an error SRQ will be issued if an attempt is made to set **FIF**ty to **ON** with an overload still present. If **COU**pling is **AC** and **FIF**ty is turned **ON**, **COU**pling will be set to **DC**. Similarly, if **FIF**ty is **ON** and **COU**pling is then set to **AC**, **FIF**ty will be turned **OFF**.
	INVert	**[ON]** **OFF**	

EXAMPLES

Query	**Response**
CH1?	CH1 VOLTS:1,VARIABLE:0,POSITION:7.60E-1, COUPLING:DC,FIFTY:OFF,INVERT:OFF
CH2? POSITION	CH2 POSITION:1.51

Vertical Commands (cont)

Header	Argument	Argument	Description
PROBe?	**CH1** **CH2** **EXT1** **EXT2**		QUERY ONLY. Will return the probe value attached to the indicated input BNC connector. The values can be: 1, 10, 100, or 1000.
BWLimit	**TWE**nty **FIF**ty **FUL**l		
VMOde	**CH1**	**[ON]** **OFF**	**VMO**de turns the display for the chosen waveform **ON** or **OFF**. Even though the display is off, the scope still acquires waveforms. A **VMO**de change only affects the display of waveforms.
	CH2	**[ON]** **OFF**	
	ADD	**[ON]** **OFF**	ADD and MULT are mutually exclusive. The selection of one causes the other to be turned off.
	MULt	**[ON]** **OFF**	
	DISPlay	**XY** **YT**	

EXAMPLES

Query	**Response**
PROBE?	PROBE CH1:10,CH2:1,EXT1:1,EXT2:1
PROBE? CH1	PROBE CH1:1
BWLIMIT?	BWLIMIT FULL
VMODE?	VMODE CH1:ON,CH2:OFF,ADD:OFF,MULT:OFF, DISPLAY:YT

Trigger Commands

Header	Argument	Argument	Description
ATRigger	**MOD**e	**AUTOL**evel	Sending an **AUTOL**evel command while **AUTOL**evel is selected forces a recalculation of the trigger level. See **INITA**t50 command also.
		AUTO	**AUTO** switches to Roll at Sec/Div settings of 100 ms and slower. There is no **AVG** Acquire Mode during Roll.
		NORmal	
		SGLseq	A single sequence is started by issuing the **RUN ACQ**uire command. An SRQ will be issued when the transition to Save Mode is made if **OPC** is **ON**. This is the way to do a hold next.
	SOUrce	**CH1** **CH2** **LIN**e **VER**tical **EXT1** **EXT2**	For any **SOU**rce selection, if **LOG**src is **WOR**d, then **LOG**src is **OFF**, and a warning SRQ is issued if **EXW** is **ON**.
	LOGsrc	**WOR**d **A.B** **OFF**	If **WOR**d is selected and the word probe is not present, a warning SRQ will be issued if **EXW** is **ON**, and **LOG**src will change to **OFF**. Likewise, if **WOR**d is currently active and the probe is disconnected, the an SRQ will be issued and the source will change to **VER**tical. **A.B** is the logical AND of A and B triggers.

EXAMPLES

Query	Response
ATRIGGER?	ATRIGGER MODE:AUTO,SOURCE:CH1, LOGSRC:OFF,COUPLING:DC,LEVEL:-9.88E-1, SLOPE:PLUS,POSITION:16,HOLDOFF:0,ABSELECT:A
ATRIGGER? MODE	ATRIGGER MODE:SGLSEQ

Trigger Commands (cont)

Header	Argument	Argument	Description
ATRigger (cont)	**COU**pling	**AC** **DC** **LFR**ej **HFR**ej **NOI**serej **TV**	Choosing the **TV** selection turns on the Video Option using the coupling choices in the **SETTV** command. An SRQ is issued if the Video Option is not present and **EXR** is **ON**. **TV** coupling and **A.B** or **WOR**d logical sources are incompatible. Selecting **TV** coupling forces **LOG**src to **OFF**; selecting any **LOG**src argument forces **COU**pling to **DC**. If either is done, a warning SRQ will be issued if **EXW** is **ON**.
	LEVel	<NR3>	Level is set in volts with no range limitations, but the actual effective level (which is based on the current trigger source) is limited to ±18 divisions in CH1 and CH2 and ±9 divisions otherwise. This allows the user to set the absolute trigger level of interest and let the instrument determine what its hardware can supply at any given Volts/Div setting. The Line Source level is considered to be 20 Volts/Div.
	SLOpe	**PLU**s **MINU**s	
	POSition	<NR1>	Sets number of data points which will be acquired prior to the trigger with (1023-[NR1×32]) points acquired after the trigger. <NR1> can range from 1 to 30, with 1 meaning 32 pretrigger and 992 post-trigger points and 30 meaning 960 pretrigger and 64 post-trigger points. If necessary, <NR1> will be limited to the nearest legal setting and, if **EXW** is **ON**, a warning SRQ will be issued.

EXAMPLES

Query	**Response**
ATRIGGER? COUPLING	ATRIGGER COUPLING:AC

Trigger Commands (cont)

Header	Argument	Argument	Description
ATRigger (cont)	**HOL**doff	<NR3>	<NR3> is settable with a resolution of 1/16 in the range 0 to 100, with 0 being the minimum holdoff (one screen) and 100 representing the maximum. Numbers outside these limits will be set equal to the value of the closest boundary and, if **EXW** is **ON**, an SRQ will be issued. Holdoff will always change to the minimum value on an A Sec/Div change.
	ABSElect	**A** **B**	Selects which trigger level the front panel Trigger Level pot controls. It also controls which trigger status is displayed on screen and on the front-panel Trigger Status indicators.
	MINImum		QUERY ONLY. Returns the current minimum level of the A Trigger channel in volts. Data is only valid immediately following completion of an auto-level cycle. An auto-level cycle is forced by sending in the **INITA**t50 command.
	MAXimum		QUERY ONLY. Returns the current maximum level of the A Trigger channel in volts. Data is only valid immediately following completion of an auto-level cycle (See **MINI**mum).
	STATe		QUERY ONLY. Shows the most "advanced" state the trigger system has reached since the last **CLR**state command. These are the responses (in order of least to most advanced state): ARMED, READY, ATRIG, RTRIG, SAVE.
	CLRstate		COMMAND ONLY. Resets the trigger-state variable to **A**RMED.

Trigger Commands (cont)

Header	Argument	Argument	Description
INITAt50			COMMAND ONLY. Will perform an autolevel cycle which will set the selected trigger level halfway between the maximum of the signal and the minimum.
BTRigger	**MOD**e	**RUNS**aft **TRI**gaft	Selects whether B sweep free runs (**RUNS**aft) or must be triggered to run (**TRI**gaft).
	EXTCLk	[**ON**] **OFF**	When **ON**, the B trigger source becomes the time base for the scope. The rate is one sample/cycle.
	SOUrce	**CH1** **CH2** **WOR**d **VER**tical **EXT1** **EXT2**	See **ATR**igger **SOU**rce comments.
	COUpling	**AC** **DC** **LFR**ej **HFR**ej **NOI**serej	See **ATR**igger **COU**pling comments.
	LEVel	<NR3>	See **ATR**igger **LEV**el comments.
	SLOpe	**PLU**s **MINU**s	See **ATR**igger **SLO**pe comments.
	POSition	<NR1>	See **ATR**igger **POS**ition comments.

EXAMPLES

Query	Response
ATRIGGER? MINIMUM	ATRIGGER MINIMUM:-9.01E+1
ATRIGGER? STATE	ATRIGGER STATE:RTRIG
BTRIGGER?	BTRIGGER MODE:TRIGAFT,EXTCLK:OFF, SOURCE:CH1,COUPLING:DC,LEVEL:-9.53E-2, SLOPE:PLUS,POSITION:16

Trigger Commands (cont)

Header	Argument	Argument	Description
SETTV	**ICO**upling	**FLD1** **FLD2** **ALT** **TVL**ine	Shows the type of display when the video signal applied is an interlaced signal. If any **SETTV** selections are made and the Video option is not present, an SRQ will be issued, if **EXR** is **ON**, and the command ignored.
	NICoupling	**FLD1** **TVL**ine	Shows the type of display when the video signal applied is a noninterlaced signal.
	INTERlaced		QUERY ONLY. Returns **ON** if the Video Option detects that the applied video signal is interlaced and **OFF** if it detects that the signal is noninterlaced.
	TVClamp	[**ON**] **OFF**	"Locks" trigger level of composite video back porch at 0 volts.
	TVLine	<NR1>	<NR1> gives the line number to be triggered on in the selected field maximum number of lines for the field (in **ALT**, the field will be **FLD2**). If outside this range, a warning SRQ will be issued (if **EXW** is **ON**) when TV Trigger is selected.
	LCNTReset	**F1O**nly **BOT**h	Reset the line count only on field 1. Reset the line count on both field 1 and field 2.
	LCNTStart	**PRE**fld **ATF**ld	Line count begins three lines before the field-sync pulse. This is the SYSTEM-M selection that will be made in the Extended Functions menu from the front panel. Line count begins at the sync pulse (non-SYSTEM M).
	SYNc	**PLU**s **MINU**s	Show direction of the sync pulse pulse relative to the baseline.

Trigger Commands (cont)

Header	Argument	Argument	Description
SETWord	**RAD**ix	**OCT** **HEX**	Shows which numbering scheme will be used in the display.
	CLOck	**ASY**nc	**ASY**nc allows the probe to "freerun" at approximately 10KHz.
		FALl **RIS**e	**FAL**l or **RIS**e selects the appropriate edge for synchronous operation.
	WORd	<ascii binary data>	<ascii binary data> will be a Y or #y followed by 17 characters, each one representing a bit in the trigger word plus the qualifier bit (the MS bit). The characters can be 0, 1, X, or x. Spaces will be ignored, but any other characters will cause an error to be generated. If an error occurs, the command will be ignored, and an SRQ will be sent if **EXR** is **ON**.
	PROBe		QUERY ONLY. Returns **ON** if the word probe is attached or **OFF** if the probe is not connected.
MANtrig			COMMAND ONLY. This command forces a trigger. The trigger will only be effective if the scope is in the READY trigger state where all pretrigger data has determined by sending the scope an ATRIGGER? STATE query.
EXTGain	**EXT1**	**DIV1** **DIV5**	
	EXT2	**DIV1** **DIV5**	

EXAMPLES

Query	**Response**
SETTV? TVLINE,SYNC	SETTV TVLINE:1,SYNC:MINUS
SETWORD?	SETWORD RADIX:HEX,CLOCK:ASYNC, WORD:#Y11110000000111001

Horizontal Commands

Header	Argument	Argument	Description
HORizontal	**MOD**e	**ASW**eep **AIN**tb **BSW**eep	If in Roll mode (Auto trigger at slow Sec/Div settings), selecting **AIN**tb or **BSW**eep causes A Trig Mode to become Normal. If **EXW** is **ON**, a warning SRQ will be issued.
	POSition	<NR3>	Position will move point <NR3> of the waveform(s) to the center of the display. In order to account for expanded sweeps, the range for <NR3> will be from 0 to 1023 with a resolution of 0.01. If the display is expanded 10X, points are "named" like this: 0.0, 0.1,... 1022.9,1023.0. If <NR3> requires rounding or limiting, a warning SRQ will be issued if **EXW** is **ON**.
	ASEcdiv	<NR3>	<NR3> can range from 1E-9 to 5 in the standard 1-2-5 sequence. If <NR3> does not match a valid Sec/Div setting, it will be set to the closest valid setting. If rounding or limiting is required, and **EXW** is **ON**, an SRQ will be issued. If the A sweep is set faster than the B sweep, they will "lock" with an SRQ being sent if **EXW** is **ON**.
	BSEcdiv	<NR3>	<NR3> range is the same as for **ASE**cdiv. The B Sec/Div setting can be set whether being used or not. If an attempt is made to set B slower than the A Sec/Div setting, it will be set equal to A, and an SRQ will be issued if **EXW** is **ON**. During subsequent changes in the A Sec/Div setting, the B Sec/Div setting will be locked to the A Sec/Div setting.

EXAMPLES

Query	**Response**
HORIZONTAL? POSITION	HORIZONTAL POSITION:5.12000E+2

Horizontal Commands (cont)

Header	Argument	Argument	Description
HORizontal (cont)	**EXTE**xp	<NR1>	<NR1> can be any of the following: 1, 2, 5, 10, 20, 50, or 100 and shows the expansion factor to be applied to an externally clocked A waveform in Save Mode. The expansion factor is reset to 1 if not in Save Mode or not working with externally clocked waveforms. Any <NR1> entered will be rounded and limited to one of the legal values and, if **EXW** is **ON**, an SRQ will be issued.
DLYTime	**DELT**a	[**ON**] **OFF**	If **DELT**a is **ON**, the delay time is the difference between **DLY1** and **DLY2**. If it is off the the delay time is **DLY1**.
	DLY1 **DLY2**	<NR3> <NR3>	**DLY1** is set to <NR3> seconds with the range and effective resolution depending on the Sec/Div setting. Both delays can be set at any time. With **DELT**a **OFF**, only **DLY1** will be used. **DLY2** sets the second delay relative to the first (it is the difference between the two). If the time base is being externally clocked, 1 Sec/Sample will be used to set the delay time. Delay time essentially becomes delay-by-events in external clock mode.
DLYEvts	**MOD**e	[**ON**] **OFF**	
	VALue	<NR1>	<NR1> ranges from 1 to 65536 and shows the number of events by which the A record trigger will be delayed. It will be limited to match a legitimate value. If limiting occurs and **EXW** is **ON**, a warning SRQ will be issued.

EXAMPLES

Query	**Response**
HORIZONTAL?	HORIZONTAL MODE:ASWEEP,POSITION:5.12000E+2, ASECDIV:1E-5,BSECDIV:1E-5,EXTEXP:1
DLYTIME?	DLYTIME DELTA:OFF,DLY1:1.8400E-5,DLY2:0
DLYEVTS?	DLYEVTS MODE:OFF,VALUE:124

Acquisition Commands

Header	Argument	Argument	Description
RUN	[**ACQ**uire] **SAV**e		The **ACQ**uire argument will cause the scope to start an acquisition. **SAV**e will cause an immediate transition to Save Mode.
ACQuire	**MOD**e	**NOR**mal **ENV** **AVG**	If set to **AVG** in delay time with **DELT**a set to **ON**, only Delay 1 will be displayed, and a warning SRQ will be issued if **EXW** is **ON**.
	REPet	[**ON**] **OFF**	When **ON**, enables equivalent time sampling at sweep speeds of 200 ns and faster. When **OFF** at these same speeds, points between samples will be interpolated. When **OFF**, the fastest sweep speed is 5 ns. **REP**et mode has no effect at sweep speeds of 500 ns and slower.
	ERAse		COMMAND ONLY. Causes the acquisition sequence currently running to restart similarly to a single-sequence reset.
	NUMEnv	<NR1> **CON**t	<NR1> is the number of envelope sweeps done before resetting. It is settable to one of the following: 1, 2, 4, 8, 16, 32, 64, 128, or 256. If it does not match, the value will be rounded to coincide with one of the above and an SRQ will be issued if **EXW** is **ON**. The number of envelopes is ignored in Roll Mode.

EXAMPLES

Query	**Response**
RUN?	RUN ACQUIRE
ACQUIRE?	ACQUIRE MODE:NORMAL,REPET:ON,NUMENV:1, NUMAVG:2,SAVDEL:OFF
ACQUIRE? MODE	ACQUIRE MODE:AVG

Acquisition Commands (cont)

Header	Argument	Argument	Description
ACQuire (cont)	**NUMAV**g	<NR1>	<NR1> is the number of sweeps averaged before starting over. It is settable to any of these: 2, 4, 8, 16, 32, 64, 128, or 256 (handled similarly to **NUME**nv).
	SAVDel	[ON] **OFF**	Turns Save-On-Delta mode **ON** or **OFF**. If **ON** and the scope detects a difference, it will go to Save Mode and issue an SRQ if **OPC** is **ON**. A controller can look for the **OPC** SRQ or see if **BUS**y? is **OFF** or see if the Atrigger State (**ATR**igger? **STAT**e) is **SAV**e to know when **SAVD**el has terminated.
	NUMACq		QUERY ONLY. The returned number will indicate the number of acquisitions that took place before Save was activated. Turning **SAVD**el to **ON** will reset the **NUMAC**q number. An intermediate number will be returned if Save has not been entered yet.
SMOoth	**ON** **OFF**		Turns smoothing **ON** or **OFF**. If **ON** a 5 point auto regressive moving average is performed on any displayed waveforms except references. Waveforms sent using a **CURV**e? are affected by smoothing.

EXAMPLES

Query	Response
SMOOTH?	SMOOTH OFF
ACQUIRE? MODE,REPET	ACQUIRE MODE:ENV,REPET:OFF
ACQUIRE? SAVDEL	ACQUIRE SAVDEL:OFF

Saveref Commands

Header	Argument	Argument	Description
SAVERef	[**STAC**k] **REF1** **REF2** **REF3** **REF4**		COMMAND ONLY. If **STAC**k is selected, an automatic reference transfer is done (See Operators Manual for order). For the others, the waveform indicated by the **REFF**rom pointer is put into the reference memory indicated by the argument. If the waveform pointed to by **REFF**rom is invalid, an error SRQ will be issued if **EXR** is **ON**.
REFFrom	**REF1** **REF2** **REF3** **REF4** **CH1D**el **CH2D**el **ADDD**el **MULTD**el **CH1** **CH2** **ADD** **MUL**t		Selects the waveform source for transfer to a reference waveform using the **SAVER**ef command.

EXAMPLES

Query	**Response**
REFFROM?	REFFROM CH1
REFDISP?	REFDISP REF1:EMPTY,REF2:OFF,REF3:ON, REF4:OFF
REFDISP? REF2	REFDISP REF2:OFF
REFPOS?	REFPOS MODE:INDEPENDENT,REF1:5.12000E+2, REF2:4.60000E+2,REF3:5.59000E+2, REF4:4.57000E+2

Saveref Commands (cont)

Header	Argument	Argument	Description
REFDisp	**REF1**	[**ON**] **OFF**	If **ON**, the selected reference is displayed. Because only 6 waveforms can be displayed at one time, turning on a reference does not guarantee that it will be displayed. If the reference is empty, an error results, and an SRQ is issued if **EXR** is **ON.**
		EMPty	The **EMP**ty command will erase the contents of the selected reference.
	REF2	[**ON**] **OFF** **EMP**ty	
	REF3	[**ON**] **OFF** **EMP**ty	
	REF4	[**ON**] **OFF** **EMP**ty	
REFPos	**REF1** **REF2** **REF3** **REF4**	<NR3> <NR3> <NR3> <NR3>	<NR3> ranges between 0 and 1023 with a resolution of 1 and labels the point on the reference that will be at center screen if **MOD**e is **IND**ependent. If **VMO**de is **XY** then **REF1** and **REF2** horizontal positions are "locked" (not independently selectable).
	MODe	**IND**ependent	**IND**ependent allows each reference to be horizontally positioned separately.
		LOCk	**LOC**k slaves the reference position to the live horizontal position. This mode is particularly useful when using Save-On-Delta mode for bringing the event of difference to center screen automatically.

Display Commands

Header	Argument	Argument	Description
INTENSIty	**DISP**lay **REA**dout **GRA**t **INTENS**	<NR3> <NR3> <NR3> <NR3>	<NR3> for all the Intensity controls will be a number from 0 to 100 with a resolution of 0.25; 0 being off and 100 being the brightest. <NR3> will be limited and, if **EXW** is **ON**, a warning SRQ will be issued. **INTENS** sets the intensity of the intensified zone in A intensified Horizontal Mode.
	VECtors	[**ON**] **OFF**	Determines whether the display is drawn using vectors (**ON**) or dots (**OFF**).
REAdout	[**ON**] **OFF**		Turns all readout display **ON** or **OFF**.
MENuoff			COMMAND ONLY. Clears the lower three lines of readout and prevents updates from occurring with each menu button push. A useful command for creating custom menus (see **MESS**age command). This command must be sent in order for any menu button SRQ's to work.

EXAMPLES

Query / **Response**

```
INTENSITY?     INTENSITY DISPLAY:3.200E+1,READOUT:4.200E+1,
               GRAT:1.800E+1,INTENS:6.825E+1,VECTORS:ON

MESSAGE?       MESSAGE 1:" READOUT\ $\ $\ $DISP\ $\ $\ $
               INTENS\ $\ $\ $\ g\ n\ a\ t\ $\ $\ $\ $\ o\ n\      |OFF",
               2:" INTENSITY",3:"\ $",4:"   ",5:"  ",6:"  ",7:"  ",8:"  ",
               9:"  ",10:"  ",11:"  "12:"  ",13:"  ",
               14:" RF3  1.00 V   10\ <\ 7  ",15:"       ",
               16:" CH1    1V         A 10\ <\ 7  -1.23 V"
```

Display Commands (cont)

Header	Argument	Argument	Description
MESSage	<NR1>	"string"	<NR1> is settable in the range 1 to 16 and will designate which row in the readout the "string" will go (line 1 is at the bottom of the screen). <NR1> values will be limited and, if **EXW** is **ON**, an SRQ will be issued. The string will be left justified in the display and will be blank filled or truncated to 40 characters long.
	CLRstate		COMMAND ONLY. Clears out all 16 lines of readout from the instrument display. *NOTE* *The top two and bottom three lines of readout are used extensively by the scope. Even though the front panel is locked out, remote changes may still cause updates to rewrite any of the top two or bottom three lines (see **MEN**uoff command). In addition, any of the following can cause an overwrite to occur:* *1. 50 Ω overload.* *2. word probe is disconnected while word-trigger source is selected.* *3. when cursors are on, line 14 is constantly overwritten.*

EXAMPLES

Query	**Response**
MESSAGE? 3,4,14	MESSAGE 14:" RF3 1.00 V 10\ <\ 7" , 4:" " ,3:" "
MESSAGE? 5	MESSAGE 5:" \ t\ h\ i\ s IS UNDERLINED"

Cursor Commands

NOTE

***V.T** cursors will return meaningful values only when the time cursors (set with **TPOs**) are on screen. The voltage values at a particular time cursor location are extracted from the display so the useful range is limited to the 512 points currently being displayed.*

Header	Argument	Argument	Argument	Description
CURSor	**FUN**ction	**VOL**ts **V.T** **SLO**pe **TIM**e **ONE**/Time **OFF**		**V.T** and **SLO**pe will place cursors only on the waveform selected using the **TAR**get arguments. There are no **ABSO**lute cursors in **SLO**pe. If **SLO**pe is requested with **ABSO**lute cursors on, **MOD**e will be changed to **DELT**a, and a warning SRQ will be issued if **EXW** is **ON**.
	TARget	**CH1** **CH2** **ADD** **MUL**t **REF1** **REF2** **REF3** **REF4** **CH1D**el **CH2D**el **ADDD**el **MULTD**el		This command selects which waveform the user would like the cursors to to appear on. The target waveform needs to be displayed; if not, an error SRQ will issued if **EXR** is **ON**, and the current target will not change.
	UNIts	**TIM**e	**BAS**e **PERC**ent **DEG**rees	**BAS**e units are seconds.
		SLOpe	**BAS**e **PERC**ent **DB**	**BAS**e units are volts/second.
		VOLts	**BAS**e **PERC**ent **DB**	**BAS**e units are volts.

Cursor Commands (cont)

Header	Argument	Argument	Argument	Description
CURSor (cont)	REFVolts	UNIts	V VV DIV	V = Volts, VV = Volts squared, and DIV = Divisions.
		VALue	<NR3>	
	REFSlope	XUNit	SEC CLKs V VV DIV	
		YUNit	V VV DIV	
		VALue	<NR3>	
	REFTime	UNIts	SEC CLKs	
		VALue	<NR3>	
	NEWref			COMMAND ONLY. Use the current position of the cursors to set up the new reference for making percentage, dB, and degree measurements.
	XPOs	ONE TWO	<NR3> <NR3>	<NR3> sets the vertical volts cursors in XY display mode. The range is ±5.1 divisions with a resolution of 0.01 divisions. <NR3> values will be limited, and if **EXW** is **ON**, a warning SRQ will be sent.

Cursor Commands (cont)

Header	Argument	Argument	Argument	Description
CURSor (cont)	**YPO**s	**ONE** **TWO**	<NR3> <NR3>	<NR3> sets the horizontal volts cursor position in divisions. The range for <NR3> is ±4.1 with a resolution of 0.01 divisions. <NR3> values will be limited, and if **EXW** is **ON**, a warning SRQ will be sent.
	TPOs	**ONE** **TWO**	<NR3> <NR3>	<NR3> sets the vertical time cursor position to the waveform point location selected. The range for <NR3> is 0 to 1023 with a resolution of 0.01 (see **HOR**izontal **POS**ition). <NR3> values will be limited , and if **EXW** is **ON** a warning SRQ will be sent.
	MODe	**DELT**a **ABSO**lute		In **DELT**a, both cursors are active and values displayed are the voltage or time differences between the two cursors. In **ABSO**lute, only the active cursor is displayed with the readout values referenced to the trigger point for time values and ground for voltage values.
	DISPlay	**VAL**ue **UNI**ts		QUERY ONLY. **VAL**ue will return the <NR3> number from the cursor readout field of the display. **UNI**ts will return the symbolic "units" string associated with the **VAL**ue. If the cursor function is off, the units returned will be **OFF**.
	SELect	**ONE** **TWO**		Command selects which cursor is the active one.

EXAMPLES

Query	Response
CURSOR? TARGET,TPOS	CURSOR TARGET:CH1,TPOS:ONE:3.12000E+2, TPOS:TWO:7.12000E+2

Automatic Feature Commands

Header	Argument	Argument	Argument	Description
AUTOSetup	**MOD**e	**VIE**w **PERI**od **RIS**e **FALl** **PUL**se		In **AUTOS**etup the scope automatically sets itself up to get a good "picture" of an incoming waveform. The **MOD**e parameter tells the scope what type of a display or measurement the user desires. For example, **RIS**e focuses on the rising slope of the waveform while **PERI**od focuses on the full cycle.
	EXEcute			Causes one autosetup cycle to take place. The scope will not act on any new commands until this cycle is complete.
	RESolution	**HI** **LO**		When **RES**olution is set to **HI** the autosetup cycle will "spread" the waveform out over the entire 20 division acquisition window so that a measurement can be made with as much resolution as possible. When **RES**olution is set to **LO** the autosetup cycle will optimize the horizontal resolution to fit the waveform on the screen. **RES**olution is ignored in **VIE**w mode.

EXAMPLES

Query	**Response**
AUTOSETUP?	AUTOSETUP MODE:VIEW,RESOLUTION:LO
AUTOSETUP? MODE	AUTOSETUP MODE:RISE

Automatic Feature Commands (cont)

Header	Argument	Argument	Argument	Description
VALue?	<types>			QUERY ONLY. Returns result of specified <type> of measurement on selected waveform. The waveform to be measured is selected with the **DAT**a **SOU**rce or **DAT**a **DSOU**rce commands. A new measurement calculation is done each time the **VAL**ue query is received. <types> available are: **DIST**al, **PROX**imal, **MESI**al, **MINI**mum, **MAX**imum, **MID**, **TOP**, **BAS**e, **MEAN**, **PK2**pk, **OVE**rshoot, **UND**ershoot, **WID**th, **PERI**od, **FRE**quency, **DUT**y, **RIS**e, **FAL**l, **RMS**, **ARE**a, **DELA**y, and **DME**sial. If the scope returns a value of 99E99 then an error has occurred. Check the event code to determine which one (see **EVE**nt?).
UNIts?	<types>			QUERY ONLY. The same as **VAL**ue? except the units associated with any measurement type are returned.

EXAMPLES

Query	Response
VALUE?	VALUE DISTAL:-8.000E-2,MESIAL:-1.080, PROXIMAL:-2.120, AREA:-1.586E-4, TOP:2.000E-1,BASE:-2.400,OVERSHOOT:0, UNDERSHOOT:2.600E+1,MAXIMUM:2.480, MINIMUM:-1.600E-1,MEAN:8.341E-1, MID:1.160,PK2PK:2.640,RMS:-9.363E+4, DELAY:99e99,FALL:4.000E-7, RISE:2.000E-7,PREIOD:2.120E-5, FREQUENCY:4.717E+4,DUTY:4.528E+1, WIDTH:9.600E-6
VALUE? RISE	VALUE RISE:2.000E-7

Automatic Feature Commands (cont)

Header	Argument	Argument	Argument	Description
MEASurement	**DISP**lay	[ON] **OFF**		Controls the display of measurements on the scope display screen.
	MARk	[ON] **OFF**		Turns the display of threshold crossing marks on or off. All time related measurements (i.e. **DUT**y, **RIS**e, **FAL**l, etc) are taken from between these marks.
	WINdow	[ON] **OFF**		Turns windowing on or off. Windowing limits measurement calculations to within the time cursor area if Time Cursors are on.
	METhod	**CURS**or **HIS**togram **MINM**ax		Selects the method for determining Top and Base. **CURS**or uses the current Volts cursor values. **HIS**togram will set Top and Base to values calculated based on a histogram. **MINM**ax will set Top and Base to the maximum and minimum respectively.

EXAMPLES

Query	Response
MEASUREMENT? WINDOW	MEASUREMENT WINDOW:OFF
MEASUREMENT? METHOD	MEASUREMENT METHOD:HISTOGRAM
MEASUREMENT? DISPLAY	MEASUREMENT DISPLAY:OFF

Automatic Feature Commands (cont)

Header	Argument	Argument	Argument	Description
MEASurement (cont)	**ONE**	**TYP**e **SOU**rce **DSO**urce	<types> <data src> <data src>	Specifies <type> and "source" of the value to be shown in measurement display line number one. Four measurements (from the <types> list) may be displayed on the screen constantly. Use commands **ONE**, **TWO**, **THR**ee, and **FOU**r, to tell the scope what type of measurement (if any) to display on each of these four lines. <data src> are the same arguments as those for the DATA SOURCE command.
	TWO	**TYP**e	<types>	Specifies <type> and "source" of the value to be shown in measurement display line number two.
	THRee	**TYP**e **SOU**rce **DSO**urce	<types> <data src> <data src>	Specifies <type> and "source" of the value to be shown in measurement display line number three.
	FOUr	**TYP**e **SOU**rce **DSO**urce	<types> <data src> <data src>	Specifies <type> and "source" of the value to be shown in measurement display line number four.

EXAMPLES

Query	Response
MEASUREMENT? ONE	MEASUREMENT ONE:TYPE:OFF, ONE:SOURCE:CH1,ONE:DSOURCE:CH1
MEASUREMENT? ONE:TYPE	MEASUREMENT ONE:TYPE:OFF
MEASUREMENT? ONE,TWO	MEASUREMENT ONE:TYPE:OFF, ONE:SOURCE:CH1,ONE:DSOURCE:CH1, TWO:TYPE:ON,TWO:SOURCE:CH1, TWO:DSOURCE:CH2

Automatic Feature Commands (cont)

Header	Argument	Argument	Argument	Description
MEASurement (cont)	**DIST**al	**UNI**ts	**PERC**ent **VOL**ts	Changes the distal value used in calculating measurement results. **PLE**vel is the value used when **UNI**ts is set to **PERC**ent while **VLE**vel is used when **UNI**ts is **VOL**ts. The distal value is most "distant" from the base. A typical value is 90%.
		PLEvel	<NR3>	
		VLEvel	<NR3>	
	MESIal	**UNI**ts	**PERC**ent **VOL**ts	Changes the mesial value used in calculating measurement results. The mesial value is in the "middle". A typical mesial value is 50%.
		PLEvel	<NR3>	
		VLEvel	<NR3>	
	PROXimal	**UNI**ts	**PERC**ent **VOL**ts	Changes the proximal value used in calculating measurement results. The proximal value is in closest "proximity" to the base. A typcial proximal value is 10%.
		PLEvel	<NR3>	
		VLEvel	<NR3>	
	DMEsial	**UNI**ts	**PERC**ent **VOL**ts	Changes the mesial value used in calculating where on the delay waveform to measure to.
		PLEvel	<NR3>	
		VLEvel	<NR3>	

EXAMPLES

Query	**Response**
MEASUREMENT? MESIAL	MEASUREMENT? MESIAL:UNITS:PERCENT, MESIAL:PLEVEL:5.00E+1,MESIAL:VLEVEL:1.300
MEASUREMENT? DISTAL	MEASUREMENT? DISTAL:UNITS:PERCENT, DISTAL:PLEVEL:9.00E+1,DISTAL:VLEVEL:2.400

Sequencer Commands

Header	Argument	Argument	Description
SETUp	**SAV**e	**ONE** **TWO** **THR**ee **FOU**r **FIV**e "ascii string"	COMMAND ONLY. The **SETU**p **SAV**e command is used to store a front panel setting into **ONE** through **FIV**e or create a sequence of multiple front panels to be stored into "ascii string". The original **SAV**e command creates the sequence with subsequent **SAV**e commands appending new steps to the sequence. This is the normal procedure for creating a sequence: 1. Send a **SETU**p **ACT**ion:<NR1> command. If the "action" will not change this step may be omitted. 2. Use normal programming commands or front panel controls to set the scope up as desired for this particular step. 3. Send a **SETU**p **SAV**e:"ascii string" command. This will create the first step in the sequence. 4. Repeat steps 2 and 3 to add more steps to the sequence. Include the **ACT**ion command (step 1) again if you wish to give unique "actions" to any particular step.

Sequencer Commands (cont)

Header	Argument	Argument	Description
SETUp (cont)	**REC**all	**ONE** **TWO** **THR**ee **FOU**r **FIV**e "ascii string"	COMMAND ONLY. Use the **SETU**p **REC**all command to recall a sequence (named **ONE** to **FIV**e, or "ascii string"). Recalling a sequence will automatically load and run that sequence.
	ACTion	<NR1>	The **SETU**p **ACT**ion command saves a bit encoded number which tells the scope what "actions" to associate with a sequence or with any particular step within that sequence. The current "action" number is associated with the next step or sequence. Only change the "action" number when new "actions" are needed. The <NR1> number sent with the **ACT**ion command may be calculated as shown in this example: For "actions" Selfcal, SRQ, Pause, and Bell the **ACT**ion number would be 2 + 64 + 128 + 32 = 226. At pwr-up, the "action" = 0. 1 = Repeat current sequence from this step to the end. 2 = Selfcal before loading step. 4 = Selftest before loading step. 8 = Do an Autosetup. 16 = Print/Plot at end of step. 32 = Bell at end of step. 64 = SRQ at end of step. 128 = Pause at end of step. 256 = Will protect this sequence if included on the first step.

Sequencer Commands (cont)

Header	Argument	Argument	Description
SETUp (cont)	**FORC**e	[**ON**] **OFF**	The **SETU**p **FORC**e command overrides the protect "action" (See **ACT**ion command). This allows any protected step to be altered from the GPIB. The front panel protect is still effective.
	DELEte	"ascii string"	COMMAND ONLY. Removes the named sequence from Sequencer memory. An error SRQ is returned if the named sequence is not present and **EXR** is **ON**.
	MEMory		QUERY ONLY. This query returns the number of bytes left in the Sequencer memory. **BUS**y is returned if the sequencer is being used by the front panel.
	NAMes		QUERY ONLY. This query returns the names of all sequences present in the Sequencer. **BUS**y is returned if the sequencer is being used by the front panel. **NON**e is returned if there are no stored sequences. The format is: SETUP NAMES:"name1", NAMES:"name2"
	CLEar		COMMAND ONLY. This command deletes sequences from the sequencer. If **FORC**e is **ON** then all sequences will be deleted. If **FORC**e is **OFF** then only sequences without a "protect" action are deleted.

EXAMPLES

Query	Response
SETUP?	SETUP ACTION:0,FORCE:OFF
SETUP? MEMORY	SETUP MEMORY:12062
SETUP? NAMES	SETUP NAMES:" TEST1" ,NAMES:" TEST2"

Sequencer Commands (cont)

Header	Argument	Description
LLPrgm	"ascii string"	When used as a query the sequence named "ascii string" will be returned in a low level binary block form. When that binary block is returned to the scope it will reconstruct the named sequence. If the incoming sequence has the same name as one currently in memory then the incoming sequence is ignored. If no "ascii string" name is sent then all sequences will be returned. If the named sequence is not present an error SRQ will be returned if **EXR** is **ON**. *NOTE* *The LLPRGM binary block form speeds up sequence transfers in a production environment, but an 'archive' taken using the **PRG**m? query is recommended for upward compatibility with future firmware releases. The binary images of the scope setup may change with future releases.*
PRGm?	"ascii string"	QUERY ONLY. Will return a list of high level commands to reconstruct the named sequence. If "ascii string" is not included, all sequences present will be sent. If **FORM**at is **ON** then formatting characters will be inserted into the text to make a very readable listing of the sequence (see **FORM**at command). The complete path (see **PAT**h) will always be printed but the longform state (see **LON**g) will be observed.
FORMat	[ON] **OFF**	If **FORM**at is **ON** then formatting characters including (CR,LF, and spaces) will be inserted into the text returned for a **PRG**m? query. The power-up default is **OFF**. *NOTE* ***FORM**at should be set to **OFF** if the controller in use terminates on linefeeds.*

EXAMPLES

Query	Response
FORMAT?	FORMAT ON

Output Commands

Header	Argument	Argument	Description
DEVIce	**TYP**e	**THI**nkjet **HPGl**	Shows which device the scope will format its output for when a **PRI**nt command is received.
	SETTIngs	[**ON**] **OFF**	Determines whether instrument settings are printed or not.
	GRAt	[**ON**] **OFF**	Determines whether the graticule is printed or not.
	TEXt	[**ON**] **OFF**	Determines whether text is printed or not. Text consists of lines 4 through 14 and, if a **MEN**uoff command has been sent, lines 1 through 3.
	WAVfrm	[**ON**] **OFF**	Determines whether waveforms are printed or not.
	PAGesize	**US** **A4**	Determines whether the pagesize is **US** (8.5 x 11) or the European **A4** size.
PRInt			COMMAND ONLY. Causes the scope to output a string, formatted for a device whose type is set using the **DEVI**ce command. A Device Clear will abort the print or plot. If the busy bit in the status byte is set or an **OPC** SRQ has not been issued by the scope then printing or plotting is ongoing. The sequence for using the **PRI**nt command is: first send the PRInt command to scope, then listen address the printer or plotter, and finally talk address the scope.

EXAMPLES

Query	**Response**
DEVICE?	DEVICE TYPE:THINKJET,SETTINGS:ON,GRAT:ON, TEXT:ON,WAVFRM:ON,PAGESIZE:US
DEVICE? TYPE	DEVICE TYPE:HPGL

Miscellaneous Commands

Header	Argument	Description
ID?		QUERY ONLY. Returns the message "ID TEK/2432, V81.1, <string>". Option(s) present will extend the **ID** string to show the configuration. Example: "...,<string>,TVTRIG" The <string> will be composed of the firmware release date, the firmware version, and the waveform processor firmware version. Example of <string>: "10-JAN-87 V1.00/1.0"
DEBug	[**ON**] **OFF**	Command controls the debug option. See the description of Debug Mode for further information.
HELp?		QUERY ONLY. Will return a list of all valid command headers available to the user.
INIT	**PAN**el	COMMAND ONLY. Will cause the scope to go to a factory preset front-panel setup. This command is useful when beginning a new test setup for initializing the scope to a known setup from which to make changes.
	GPIb	COMMAND ONLY. Causes all bus unique commands to be initialized to known states as follows: **PATh ON**; **DEB**ug **OFF**; **LON**g **ON**; **OPC ON**; **CER ON**; **EXW ON**; **EXR ON**; **PID OFF**; **LOC**k **LLO**; **INR ON**; **DEVD**ep **ON**; **USE**r **OFF**; **DAT**a **ENC**dg:**RIB**inary; **DAT**a **TAR**get:**REF1**, **DAT**a **SOU**rce:**CH1**; **FAS**txmit **OFF**; **FAS**txmit 1; **FAS**txmit **ENC**dg:**RIB**inary; **STAR**t 256; **STO**p 512; **LEV**el 0; **HYS**teresis 5; **DIR**ection **PLU**s; **SETU**p **FORC**e:**OFF** **SETU**p ATTRIBUTE:0; **DT OFF**; It also clears the event buffer.
	SRQ	COMMAND ONLY. Clears all pending SRQs and event codes. This is normally used to initialize the scope so waiting for an **OPC** event code is a compare if event is not zero.
	[**BOT**h]	Does both the **PAN**el and **GPI**b initializations.

EXAMPLES

Query	**Response**
ID?	ID TEK/2432,V81.1," 20-JAN-87 V1.20/1.2"

Miscellaneous Commands (cont)

Header	Argument	Description
LONg	[ON] **OFF**	Command determines whether a response to a query is given with unabbreviated symbols (**ON**) or not.
SET?		Returns an ascii string that reflects the current instrument state and can be returned to the scope to recreate that state. The **LON**g command will increase the size of the **SET**? response.
LLSet	<binary block>	<binary block> will consist of a low-level "readout" of the instrument state which will be shorter in length than a human readable version. The **LLS**et command is much faster than the **SET**? query. *NOTE* *The **LLSet** binary block form will speed up front panel transfers in a production environment but an 'archive' taken using the **SET?** query is recommended for upward compatibility with future firmware releases. The binary images of the scope setup may change with future releases.*
PATh	[ON] **OFF**	When **PAT**h is **ON**, the full path name is returned for a query response. If **PAT**h is **OFF**, just the last item is sent. For example, with **PAT**h and **LON**g both **ON**, the query INTENSITY? DISPLAY would return: "INTENSITY DISPLAY:10.5;". With **PAT**h **OFF**, just "10.5;" would be returned. This will allow the programmer to read just the scope setup part of the response into his program without having to "strip" off unwanted parts.

EXAMPLES

Query	**Response**
HELP?	HELP AUTOSETUP,CH1,CH2,VMODE,ATRIGGER, ACQUIRE,DLYTIME,CURSOR,RUN,BTRIGGER, HORIZONTAL,SETWORD,EXTGAIN,REFFROM, REFDISP,BWLIMIT,DLYEVTS,INTENSITY, MEASUREMENT,DEVICE,READOUT,REFPOS, SMOOTH,CER,DATA,DEBUG,DEVDEP,DIRECTION, DT,EXR,EXW,FORMAT,HYSTERESIS,INR,LEVEL, LOCK,LONG,OPC,PATH,PID,RQS,SETUP,START, STOP,USER,SETTV,BELL,BUSY,CURVE, EXECUTE,FASTXMIT,HALT,INIT,INITAT50, LLPRGM,LLSET,LOOP,MANTRIG,MENUOFF

Miscellaneous Commands (cont)

Header	Argument	Description
BELl		COMMAND ONLY. Rings the bell.
REM	"ascii string"	COMMAND ONLY. Does nothing by throwing away the "ascii string". Useful for commenting user programs.
TIMe?		QUERY ONLY. Returns the internal clock time in an ascii string. This is the same number shown on screen when the save button is pushed. The format is "hh:mm" where hh is hours and mm is minutes. This clock cannot be reset by a user and any elapsed time numbers must be calculated from a starting time.
DT	**OFF**	The DT command tells the scope what to do with any Group Execute Trigger bus command it might receive over the GPIB. The **DT OFF** command causes the scope to ignore any Group Execute Trigger command sent to it. **DT OFF** is the power-up state.
	RUN	Does the same as a **RUN ACQ**uire command when a Group Execute Trigger is received.
	SODRUN	Executes the following command set when a Group Execute Trigger is received: "**ACQ**uire **SAVD**el:**ON**;**RUN ACQ**uire".
	STEp	When a Group Execute Trigger command is received the Sequencer will go on to the next step if it is currently pausing.
	"ascii string"	When a Group Execute Trigger command is received the scope will load the named sequence into its Sequencer memory and start the execution of the sequence.

EXAMPLES

Query	Response
TIME?	TIME "19:54"
DT?	DT SODRUN

Waveform Commands

Default values for the waveform preamble will not be sent or received, but they are needed when using the preamble information to associate waveform scaling with the waveform data in the message. The defaults are:

XZERO=0 YZERO=0 BIT/NR=8 BYT/NR=1 CRVCHK=CHKSM0

Waveforms acquired in Roll Mode do not have a trigger point. The current trigger position will be returned in a **WFM**pre **PT.O**ff query.

Header	Argument	Argument	Description
WAVfrm?			QUERY ONLY. A **WAV**frm? query will cause both **WFM**pre? and **CURV**e? queries to be generated for the waveform specified by the **DAT**a **SOU**rce pointer.
CURVe	<wfm data>		The **CURV**e command or query is used to send or receive just waveform data. The **DAT**a **SOU**rce pointer shows which data to send when queried, the **DAT**a **TAR**get pointer shows where to put data on a command, and the **DAT**a **ENC**dg pointer shows which format to send it in. <wfm data>, as defined by Codes and Formats, can be either ascii or binary format. For more information see Appendix B. *NOTE* *The curve data sent is that which would go into a reference memory if the Saveref button was pressed. If **SMO**oth is **ON** then the data returned will be "smoothed".*

Waveform Commands (cont)

Header	Argument	Argument	Description
DATa	**ENC**dg	**ASC**ii	
		RPBinary	Rpbinary is positive integer (0 to 255).
		RIBinary	Ribinary is twos complement (-128 to 127).
		RIPartial	Ripartial is partial format for Ribinary.
		RPPartial	Rppartial is partial format for Rpbinary For more information on these formats and how to use them see Appendix B.
	TARget	**REF1** **REF2** **REF3** **REF4**	Targets which reference memory will receive the next waveform sent to the scope.
	SOUrce	<types>	Points to the waveform that a **CURV**e?, **WFM**pre?, or **WAV**frm? query will return information on. If the waveform of interest is not valid (empty reference) an error will be returned when the information is requested if **EXR** is **ON**. <types> available are: **CH1, CH2, ADD, MULt, REF1, REF2, REF3, REF4, CH1D**el, **CH2D**el, **ADDD**el, and **MULTD**el.
	DSOUrce	<types>	The same as **DAT**a **SOU**rce.

EXAMPLES

Query	Response
DATA?	DATA ENCDG:RIBINARY,TARGET:REF1, SOURCE:CH1,DSOURCE:CH1
DATA? ENCDG	DATA ENCDG:RPPARTIAL
DATA? ENCDG,TARGET	DATA ENCDG:RPBINARY,TARGET:REF1

Waveform Commands (cont)

Header	Argument	Argument	Description
WFMpre	**WFId**	"ascii string"	The **WFId** field in the **WFM**pre is left undefined by Codes and Formats to allow individual instruments the ability to communicate information not included elsewhere in the preamble. In the scope this section will include labeling information to help the user remember key features about the waveform and will include vertical mode, coupling, Volts/Div, Sec/Div, and what the particular acquisition mode was. The scaling information is the same as the preamble but is given in scope "units." See examples at the end for the form of this argument. This argument will be ignored if sent as a command.
	NR.Pt	1024	During waveform input, the scope will always expect waveforms with 1024 points. (All received waveforms are placed in reference memory.) Waveforms received with less than 1024 points will have the remainder filled with the last data point sent. All waveforms output by the scope will contain 1024 points. This argument will be ignored if sent as a command.
	PT.Fmt	**Y**	Point format defines how to interpret the curve data. Y format means that x information is implicit and the data points sent are the y values. Each point will be sequential in time (older to younger).
		ENV	Format used for envelope waveforms. The data is sent in the form: y1max,y1min,y2max.... Data sent to the scope with this format will be treated as an envelope waveform.

Waveform Commands (cont)

Header	Argument	Argument	Description
WFMpre (cont)	**XUN**it	**SEC**	If the argument is **SEC**, the **XIN**cr **CLK**s value has units of seconds. If it is is **CLK**s, the scaling is for **EXTCL**k.
	XINcr	<NR3>	<NR3> will give the time interval between points (sampling rate). It is calculated by assuming 50 pts/div and dividing that into the sweep rate. It will range from 1.0E-1 (5 sec/div) to 2.0E-11 (1 ns/div). <NR3> *50 will be rounded and limited to match a legitimate Sec/Div setting; and, if done, a warning SRQ will be issued if **EXW** is **ON**. For a query response with an unknown sec/div (as for External Clock), <NR3> will be set to 1.
	PT.Off	<NR1>	Shows the location of Rtrig within the waveform. <NR1> can vary between 0 and 1023 in increments of 32. A number that does not fit the increments or limits will be rounded and limited to match. If rounding or limiting occurs, a warning SRQ will be issued if **EXW** is **ON.** 0 means the trigger point is off the record to the left, and 1023 indicates that it is off the record to the right.
	YUNit	**V** **VV** **DIV**	Gives magnitude in volts when associated with **YMU**lt.

EXAMPLES

Query	**Response**
WFMPRE?	WFMPRE WFID:" CH1 DC 1V 100ms ENV" , NR.PT:1024,PT.OFF:512,PT.FMT:ENV, XUNIT:SEC,XINCR:2.000E-3,YMULT:4.000E-2, YOFF:1.425E+1,YUNIT:V,BN.FMT:RI, ENCDG:BINARY
WFMPRE? YOFF	WFMPRE YOFF:5.450E+1

Waveform Commands (cont)

Header	Argument	Argument	Description
WFMpre (cont)	**YMU**lt	<NR3>	This value gives the vertical "step" size of the digitizer (volts between points); computed from the Volts/Div setting by assuming 25 points per division vertically.
	ENCdg	**ASC**ii **BIN**ary	Shows the encoding type that will used on the next **CURV**e query. **BIN**ary includes any of these modes: ribinary, rpbinary, ripartial or rppartial.
	YOFf	<NR3>	**YOF**f locates the ground in digitizing levels relative to data 00. If the **YOF**f is positive, then ground is above center screen <NR3> can range from -2500 to 2500 with a resolution of 0.25. Incoming values will be limited to this range; and, should limiting occur, a warning SRQ will be issued if **EXW** is **ON**.
	BN.Fmt	**RI** **RP**	**BN.F**mt tells the scope how to handle binary curve data coming in. **RI** says treat it as a twos complement representation with the MSB being interpreted as a sign bit. Digitized values being output will range from 80 (-128) to 0 to 7F (+127). **RP** indicates positive integers (0 to 255). *NOTE* *The scope powers-up expecting **RI** binary data. If **RP** is to be sent, first send a **BN.F**mt command with **RP** as the argument to change the scope.*

EXAMPLES

Query	Response
WFMPRE? YMULT	WFMPRE YMULT:4.000E-2
WFMPRE? ENCDG	WFMPRE ENCDG:ASCII

Waveform Commands (cont)

Header	Argument	Argument	Description
FAStxmit	<NR1>		COMMAND ONLY. Controls the fast transmit mode of the scope. This mode allows the fastest waveform transfer rate for the scope but is not very user friendly. Please read Appendix I before using this mode. <NR1> selects the number of waveforms to return. The **DELT**a and **NOR**mal selections determine what waveform to send and the **ENC**dg selects how to send it. There is a speed penalty for using **RPB**inary or **RPP**artial because the native format is **RIB**inary and the conversion takes time.
	DELTa	**CH1**	
		CH2	
		BOTh	
	NORmal	CH1	
		CH2	
		BOTh	
	OFF		
	ENCdg	**RIB**inary	
		RPBinary	
		RIPartial	
		RPPartial	

Waveform Data Commands

The **DAT**a **SOU**rce variable shows which waveform will be looked at for all measurement queries. The most positive points (127) and the most negative points (−128) are discarded before any calculations are done on the waveform data.

Both **PCR**oss and **NCR**oss use hysteresis. Let's use **PCR**oss as an example. In order to find a positive crossing of a level (set using **LEV**el) there needs to be a point "below" the crossing level before looking for a point at or above the crossing level. The hysteresis number indicates how many digitizing levels below the crossing level that point has to be before it is considered to be "below". The amount of hysteresis is settable using the **HYS**teresis command.

Header	Argument	Description
STARt	<NR1>	<NR1> varies between 1 and 1024 and sets the start of the interval for the actual measurement queries. **STAR**t must be less than **STO**p; if they are not of the proper size relationship, they will be swapped before they are used. A warning SRQ will be issued if <NR1> is outside the allowable range and **EXW** is **ON**.
STOp	<NR1>	<NR1> sets the end of the interval for the waveform measurement queries (see **STAR**t).
LEVel	<NR1>	<NR1> varies between -127 and 126 and sets the vertical level for **PCR**oss and **NCR**oss.
MAXimum?		QUERY ONLY. Returns an <NR1) number from -127 to 126, which represents the maximum value of the acquired waveform between **STAR**t and **STO**p. If all waveform points are 127 or all points are -128 then that number is returned to indicate direction of the "out of bounds" waveform.

EXAMPLES

Query	Response
START?	START 1
STOP?	STOP 1024
MAXIMUM?	MAXIMUM 65
LEVEL?	LEVEL 23

Waveform Data Commands (cont)

Header	Argument	Description
VMAximum?		QUERY ONLY. Similar to **MAX**imum? except **VMA**ximum returns a value in the same units shown by the volts cursor display. 99e99 is returned if all points are "out of bounds" or the units are shown as "DIV".
MINImum?		QUERY ONLY. Returns an <NR1> number, from -127 to 126, which represents the minimum value of the acquired waveform between **STAR**t and **STO**p. If all waveform points are 127 or all points are -128, then that number is returned to indicate direction of the "out of bounds".
VMInimum?		QUERY ONLY. Similar to **MINI**mum? except **VMI**nimum returns a value in the same units shown by the volts cursor display. 99e99 is returned if all points are "out of bounds" or the units are shown as "DIV".
AVG?		QUERY ONLY. Returns an <NR3> number that represents the average of the points between **STAR**t and **STO**p. <NR3> will vary from -128 to 126, with -128 indicating that no valid points (-127 to 126) are in the requested interval.
VAVg?		QUERY ONLY. Similar to **AVG**? except except **VAV**g returns a value in the same units shown by the volts cursor display. 99e99 is returned if all points are "out of bounds" or the units are shown as "DIV".

EXAMPLES

Query	**Response**
VMAXIMUM?	VMAXIMUM 4.200E-1
VMAXIMUM?	VMAXIMUM 99e99
VMINIMUM?	VMINIMUM -2.180
AVG?	AVG 3.55370E+1
VAVG?	VAVG -7.56965E-1

Waveform Data Commands (cont)

Header	Argument	Description
PCRoss?		QUERY ONLY. Returns an <NR1> number, from 0 to 1024, that represents the first waveform point in the interval of **STAR**t to **STO**p whose value is at or above **LEV**el when the previous point (left to right) was below **LEV**el. If there is no positive crossing point, a value of 0 is returned.
NCRoss?		QUERY ONLY. Returns an <NR1> number, from 0 to 1024, that represents the first waveform point in the interval of **STAR**t to **STO**p whose value is at or below **LEV**el when the previous point (left to right) was above **LEV**el. If there is no negative crossing point, a value of 0 is returned.
SNAp		COMMAND ONLY. Puts the current value for the time cursors into start and stop. Useful for doing partial waveform transfers (see Appendix B).
HYSteresis	<NR1>	<NR1> may range from 0 to 255 and gives the number of digitizing levels (increments) that a curve must go below (or above for **NCR**oss) the level value before the search is started for the correct crossing. <NR1> is set to 5 at power-up.
DIRection	**PLU**s **MINU**s	Shows which direction to proceed with the search for a **PCR**oss or an **NCR**oss. **PLU**s will go from **STAR**t to **STO**p and **MINU**s will go from **STO**p to **STAR**t. The definition of positive and negative crossings is the same regardless of the direction of the search.

EXAMPLES

Query	Response
HYSTERESIS?	HYSTERESIS 5
DIRECTION?	DIRECTION PLUS
PCROSS?	PCROSS 0
PCROSS?	PCROSS 302
NCROSS?	NCROSS 360

Service Request Commands

Header	Argument	Description
RQS	[ON] **OFF**	**RQS** causes the scope to assert SRQ when it has an event to report. If this feature is turned off, events are still accumulated and can be retrieved with an **EVE**nt? query.
OPC	[ON] **OFF**	Enables SRQ upon the completion of a command. The scope has several commands that use the operation complete feature, including: Single Seq, Save-On-Delta, plot complete, and self-test complete.
CER	[ON] **OFF**	If **ON**, this mask will allow the scope to assert an SRQ whenever a command error is detected. Some examples of command errors are syntax errors or invalid characters in the input.
EXR	[ON] **OFF**	If **ON**, this mask will allow the scope to assert an SRQ whenever an execution error is detected. Illegal characters sent to set up the word trigger are one example of an execution error.
EXW	[ON] **OFF**	If **ON**, this mask allows the scope to assert an SRQ whenever a warning condition is detected. Some examples of warning conditions are parameter rounding or limiting and trying to set B Sec/Div slower than the A Sec/Div.
INR	[ON] **OFF**	If **ON**, this mask allows the scope to assert an SRQ whenever an internal error is detected. An example of an internal error is a self-test failure.
USEr	[ON] **OFF**	Enables/disables an SRQ generated by a menu switch being pushed. Each switch will have a unique event code associated with it, allowing a controller to create special menus and react to user feedback. A **MEN**uoff command, to tell the scope that possible custom text has been written, should be sent before bezel button push event codes will be reported.

EXAMPLES

Query	**Response**
RQS?	RQS ON
USER?	USER OFF
EXW?	EXW ON
INR?	INR OFF
CER?	CER OFF

Service Request Commands (cont)

Header	Argument	Description
DEVDep	[**ON**] **OFF**	Enables/disables a device dependent SRQ such as one generated when a Transmit or Abort button is pushed.
PID	[**ON**] **OFF**	Enables/disables an SRQ generated by a probe identify button being pushed.
EVEnt?		QUERY ONLY. Returns the most recent event held by the scope or a 0 if none exists. See the Event code tables for an interpretation of the event numbers.
BUSy?		QUERY ONLY. Returns "**ON**" if the busy bit in the status byte is set; "**OFF**" if not. The busy bit will be set when the scope is doing something whose completion might cause an **OPC** SRQ to be issued.
LOCk	**ON** **OFF** **LLO**	Controls the front panel lock state. If **ON** then the front panel controls are locked out. **LLO** is the pwr-up default and indicates that the scope will lock out the front panel whenever the universal GPIB command, LLO, is received. **OFF** indicates that the controller would not like the front panel locked out although certain commands will cause the scope to lock its panel by itself (self cal).

EXAMPLES

Query	**Response**
LOCK?	LOCK LLO
LOCK?	LOCK ON
BUSY?	BUSY OFF
EVENT?	EVENT 0
EVENT?	EVENT 467
PID?	PID OFF

Calibration and Diagnostic Commands

Header	Argument	Description
TESTType	**SELFC**al **SELFD**iag **EXTCA**l **EXTD**iag	Indicates which type of test will run upon receipt of an **EXE**cute command.
TESTNum	<NR1>	<NR1> indicates the test number within the **TESTT**ype (extended calibration or extended diagnostics) that is to be run. If **SELFC**al or **SELFD**iag is the **TESTT**ype, the **TESTN**um will be reset to 0000 when an **EXE**cute command is received.
EXEcute		COMMAND ONLY. Causes the test selected by **TESTN**um to execute. Any tests in the test hierarchy below the selected number will all be run. An SRQ will be sent upon successful completion of the test sequence if **OPC** is **ON**. If a test was unsuccessful, an SRQ will be issued if **INR** is **ON** and the **ERR**or? query may then be used to return the test status. After executing any test, send a **MEN**uoff command to reset the hardware to it original state before sending any "normal" scope commands to the scope.
ERRor?		QUERY ONLY. **ERR**or? returns a string of error numbers (up to nine) resulting from the last **EXE**cute command (or 0 if no errors exist). Only those error numbers associated with the same level in the test hierarchy as that of **TESTN**um are returned. The exact test that failed in a hierarchy is found by moving **TESTN**um to a lower level, rerunning the test, and reissuing the **ERR**or? query until the failure is isolated to a test at the lowest level.
STEp		COMMAND ONLY. Causes the current test to advance to the next step and begin execution of the new step. An SRQ will be sent to indicate completion of the step if **OPC** is **ON**. **STE**p will also cause a "paused" sequence to continue to the next step.
LOOp	**CON**t **FA**il **PAS**s **ONE**	COMMAND ONLY. Causes the next test executed to run in a loop either until a **HAL**t is received or the argument condition is met.
HALt		COMMAND ONLY. Stops any test being executed; specifically used to stop a looping test. Will also stop any sequence that is currently running.

Appendix B

Discussion on Waveforms

A “Waveform Transfer” is the exchange of a block of waveform data between the 2432 and the controller. This data block contains data points which trace the shape of the waveform. You may tell the 2432 to upload waveforms TO the controller (for analysis, printout, etc), or to download waveforms FROM the controller (for re-display, comparisons, or Save-On-Delta operations).

The actual transfer of a waveform involves several different issues. The first issue is getting the data to go where you want it to go. Then there is the data format and the meaning of each data point. And finally, there is the problem of translating the data points into a format that the controller program can use. Let’s look at each of these.

Doing a Waveform Transfer

In this section we describe transferring waveform data from a controller to a 2432 and back again. In later sections we discuss what the data actually means and how to interpret it. For now we will treat the waveforms as strings to be sent back and forth.

From the 2432 to a Controller

First you need to select which waveform you wish to transfer. Use the DATA SOURCE command to do this. See the “Waveform Commands” section of Appendix A for a description of this command. As an example lets say you want CH1. You would then send 'DATA SOURCE:CH1' to the 2432 to set the source pointer to CH1. Now you need to transfer the data. This is done by sending a 'CURVE?' query to the 2432 and reading the result. Here is a short sample program written in 4041 BASIC to transfer a CH1 waveform from the 2432 to the controller. The 2432’s address should be set at 1.

```
10  scope = 1
20  open #scope:" gpib0(pri=1,eom=<0>):"
30  dim wave$ to 6000
40  print #scope:" data source:ch1"
50  print #scope:" curve?"
60  input #scope:wave$
70  stop
80  end
```

The CH1 waveform is now contained in the string array named “wave$”.

From the Controller to the 2432

To send a waveform from a controller to the 2432 you need to decide where the waveform should be stored within the 2432. Only the four reference memories can accept incoming data. Use the DATA TARGET command to select one of these memories (this command is explained in the "Waveform Commands" section of Appendix A). For example, if you wish to store the waveform into reference memory 1 then send 'DATA TARGET:REF1'. After selecting the target memory, you may transmit the waveform to the 2432. Here is a short sample program that will take the waveform stored in string array wave$ and send it to the 2432. The program is written in 4041 BASIC. The string array could be generated by another part of the controller program or this program segment could be joined to the program shown above to return the waveform just retrieved from the 2432.

```
10  scope = 1
20  open #scope:" gpib0(pri=1,eom=<0>):"
30  print #scope:" data target:ref1"
40  print #scope:wave$
50  stop
60  end
```

There are two things to keep in mind when sending waveforms to the 2432. First, the reference being filled may not be displayed. To display the reference see the VMODE command in the "Vertical" section of Appendix A. It is faster to send a waveform to the 2432 if the target reference is not displayed. Second, if you are sending waveforms in binary format you must tell the 2432 what kind of binary data it is. Read the section on "Data Formats" later in this appendix if you don't know what kind of binary data you are dealing with. To prepare the 2432 to handle Signed Integer data send 'WFMPRE BN.FMT:RI' to the oscilloscope. For Positive Integer data send 'WFMPRE BN.FMT:RP'. At power-up the 2432 expects to receive binary data in Signed Integer (RI) binary format. See the "Waveform Commands" portion of Appendix A for more information on the WFMPRE command.

From 2432 to 2432

You may transfer waveforms from one 2432 to another 2432 using the GPIB. Waveforms can also be sent from the 2432 to other oscilloscopes that understand the waveform encoding formats the 2432 uses. The Tektronix 2430, 2430A, and 7D20 can receive 2432 waveforms.

To perform a scope-to-scope transfer you must first set the "from" oscilloscope to talk only mode and the "to" oscilloscope to listen only. (See the section on "Setting Up the 2432".) Make sure the waveform to be sent is displayed on the "from" oscilloscope's CRT. Next, push the OUTPUT front-panel button followed by the TRANSMIT menu button. The waveform should now be in the "to" oscilloscope's currently selected reference memory (i.e. the target memory most recently selected with the DATA TARGET command). To determine which memory is currently selected, look at the GPIB STATUS menu (see the section on "Debugging the Interface" for more information on the status menu).

There is one limitation in scope-to-scope transfers which you should keep in mind. You cannot use the front panel to select which reference the waveform goes into. This means that every waveform sent by the "from" scope will go into the same reference in the "to" 2432. Only the last waveform sent will remain in the selected target reference. Also, make sure the waveform encoding on the "from" scope is the same as the "to" scope. On a 2432 to 2432 transfer this information is shown in the GPIB status menu. The encoding of the "from" scope is listed to the right of ENCDG. The type of binary data the "to" scope expects is listed in the BINWFM to SCOPE field. Since the "to" scopes encoding cannot be changed you will have to change the "from" scopes encoding in the ENCDG menu if the two do not match.

Waveform Data Formats

Waveform data may be encoded in several different formats. These "Waveform Formats" simply define how a waveform is to be represented in a data block. Waveform Formats available on the 2432 include ASCII and four different BINARY formats including RIBINARY, RPBINARY, RIPARTIAL, or RPPARTIAL.

ASCII format uses Signed Integer data-point representation. RIBINARY sends a whole waveform (1024 data points) in Signed Integer data-point representation, RPBINARY sends a whole waveform in Positive Integer data-point representation. RIPARTIAL sends partial waveforms in Signed Integer data-point representation, RPPARTIAL sends partial waveforms in Positive Integer data-point representation.

You may select any one of these waveform encoding formats by using the DATA ENCDG command. For example, to set the 2432 to RPBINARY format simply send the command 'DATA ENCDG:RPBINARY'.

Data Point Representation

The 2432 has an 8 bit A-D converter which is capable of quantizing a waveform with a resolution of 1 part in 256. Waveforms are displayed on the instrument screen with a vertical resolution of 25 values per division. All 256 values are displayed in just over 10 divisions. These values can be expressed in two different ways: Positive Integer and Signed Integer.

POSITIVE INTEGER REPRESENTATION. In positive integer format the values range from 0 to 255 with 0 starting 1 division below the bottom of the 2432 screen, 127 is at center screen, and 255 is 1 division above the top of the screen.

SIGNED INTEGER REPRESENTATION. In this method, the data value at center screen is 0. Any values below center screen are negative and any above it are positive. Values will range from −128 to 0 to 127 with −128 being 1 division below the bottom of the 2432 screen, 0 being at center screen, and 127 being 1 division above the top of the screen.

ASCII Format Encoding

In ASCII format each waveform point is composed of up to four ASCII digits that describe a Signed Integer data value. For example, if a waveform point is 1 division below center screen, its Signed Integer representation is −25 (remember 25 digitizing levels per division) and the ASCII description of that point would be an ASCII minus sign ('−') followed by the ASCII number 2 followed by the ASCII number 5 ('−25'). Signed Integers that are positive do not have a plus sign prepended. The entire waveform transmission is made up of individual waveform point values separated by comma's.

Waveform data blocks transferred in ASCII may have a varying number of characters, depending on the actual waveform being sent. If all the data values in the waveform to be sent are −125, then the total number of characters that will be sent is (4 × 1024) + 'CURVE ' + 1023 comma's or approximately 5100 bytes. ASCII format is selected by sending a 'DATA ENCDG:ASCII' command to the 2432.

ASCII WAVEFORM TRANSFER EXAMPLE. Table B-1 shows how traffic over the GPIB might look during an ASCII waveform transfer. In this example the first data point value is 35 and the second is 54. Each value is separated by a comma. These two points are then followed by an additional 1021 points which are not shown but are located between the two adjacent commas. The last point is a 41 which is then followed by the selected message terminator. The length of this hypothetical waveform is not known so the byte numbers revert to XXXX. The 'CURVE' part of the header will be absent if PATH is OFF.

Table B-1
ASCII Waveform Transfer

Byte	ASCII	Decimal	EOI (1=asserted)
1	C	67	0
2	U	85	0
3	R	82	0
4	V	86	0
5	E	69	0
6	<SP>	32	0
7	3	51	0
8	5	53	0
9	,	44	0
10	5	53	0
11	4	52	0
12	,	44	0
XXXX	,	44	0
XXXX	4	52	0
XXXX	1	49	1 (Only if term=EOI)
XXXX	<CR>	13	0 (If term=LF/EOI)
XXXX	<LF>	10	1 (If term=LF/EOI)

Binary Formats

In all 2432 Binary Formats, each waveform point is represented by one 8-bit data byte. There are two different types of binary waveform transfers, "Partial waveform transfers" and "Entire waveform transfers". Each of these can use either Positive Integer or Signed Integer data representation as described above. Thus a total of four binary waveform formats are available from the 2432. These formats are called RIBINARY, RIPARTIAL, RPBINARY, and RPPARTIAL. Both RIBINARY and RIPARTIAL data encoding selections use the Signed Integer method of data representation but each has a different way of representing the entire waveform. RPBINARY and RPPARTIAL both use Positive Integer.

Since we've already discussed the two types of binary data representation, lets explain the two different ways of stringing together waveform data points.

ENTIRE WAVEFORM BINARY FORMATS. RIBINARY and RPBINARY data encoding send the entire 1024 point waveform. The format for the curve data is as follows:

```
%xxd...dc
```

Where:

% is the starting character for this block.

xx is the byte count which is the total number of all data bytes and the checksum byte. This is a 2-byte field. For example, waveforms sent from the 2432 always have the first byte set to 04 hexadecimal and the second byte set to 01 hexadecimal. The whole field is a 16-bit hexadecimal number 0401h which in decimal is 1025 which indicates that there will be 1024 waveform points followed by the one byte of the checksum.

d...d are 8-bit waveform data bytes. In the 2432 there are 1024 data bytes.

c is the 8-bit checksum. The checksum is the 2's complement of the least significant 8 bits of the sum of both bytes of the byte count and all data bytes. Here is a sample algorithm for calculating the checksum.

```
checksum = ls_byte_count
checksum = checksum + ms_byte_count
for i = 1 to end_of_wfm
begin
   checksum = (checksum + wfm_data(i) ) mod 256
   i = i+1
end
checksum = 256 - checksum
```

EXAMPLE of ENTIRE WAVEFORM BINARY FORMAT. This example uses the same hypothetical waveform used in the ASCII format example above. The checksum bytes are unknown so they are XX'ed out. The byte count for this format will always by 1025 (401h). The 'CURVE' part of the header will not be present if PATH is OFF.

Table B-2
Binary Waveform Transfer

Byte	ASCII	Decimal	Hex	EOI (1=asserted)
1	C	67	43	0
2	U	85	55	0
3	R	82	52	0
4	V	86	56	0
5	E	69	45	0
6	<SP>	32	20	0
7	%	37	25	0
8	<Bin Count MSB>	4	04	0
9	<Bin Count LSB>	1	01	0
10	5	53	35	0
11	T	84	54	0
		...		
		... more waveform data		
		...		
1033	A	65	41	0
1034	<Checksum>	XX	XX	1 (Only if term=EOI)
1035	<CR>	13	0D	0 (If term=LF/EOI)
1036	<LF>	10	0A	1 (If term=LF/EOI)

PARTIAL WAVEFORM BINARY FORMATS. RIPARTIAL and RPPARTIAL waveform formats send only part of a waveform. The portion of the acquisition sent is determined by the START and STOP values. For example, if START is set to 256 and STOP is set to 512 (power-up values) then a Partial Binary Format waveform would include the 257 (512-256+1) points starting at point 256 and ending with point 512. The points in a waveform are labeled from 1 to 1024, starting with the earliest point (point 1) in time. See the descriptions in the "Waveform Data Command" portion of Appendix A for more information on START and STOP.

You may return a partial waveform in Binary Format to the 2432. This only updates a portion of the reference waveform currently in the 2432. For example, turn the time cursors ON and move the time cursors so that they bracket the area you would like to send. Then send the 'SNAP' command to the 2432. This will set the START and STOP values equal to the time cursor positions. Then request a partial waveform of CH1. When you send this CH1 partial waveform back, the segment of the reference waveform between START and STOP will now look just like the corresponding segment of CH1.

The format for the partial curve data is:

#cx...xd...d

Where:

#	is the starting character for this block.
c	is the number of bytes in the byte-count number. This is an ASCII digit from 1 to 9.
x...x	is the byte count. These are ASCII digits from 0 to 9 with the number of digits specified in the c field. This byte count tells how many bytes are in the d...d field. Using the example above, the '#cx...x' part of the partial format would look like: #3260.
d...d	is the waveform data field. This field has several sub-fields which look like: 'Cssy...y'.

Where:

C	is one byte used to indicate the type of binary encoding used. This is 02h for Positive Integer and 01h for Signed Integer.
ss	is two bytes that indicate where in the 1024 acquisition to start putting the data bytes. In the above example where START = 256 this field would be 0100h.
y...y	is the actual waveform data bytes.

EXAMPLE OF A PARTIAL WAVEFORM IN BINARY FORMAT. This example shows what a hypothetical partial waveform would look like. It is set up with START = 256 and STOP = 512. The CURVE part of the header will not be present if PATH is OFF.

Table B-3
Partial Binary Waveform Transfer

Byte	ASCII	Decimal	Hex	EOI (1=asserted)
1	C	67	43	0
2	U	85	55	0
3	R	82	52	0
4	V	86	56	0
5	E	69	45	0
6	<SP>	32	20	0
7	#	35	23	0
8	3	51	33	0
9	2	50	32	0
10	6	54	36	0
11	0	48	30	0
12	STX	2	02	0
13	SOH	1	01	0
14	NUL	0	00	0
15	5	53	35	0
16	T	84	54	0
		...		
		... more waveform data		
		...		
271	A	65	41	1 (Only if term=EOI)
272	<CR>	13	0D	0 (If term=LF/EOI)
273	<LF>	10	0A	1 (If term=LF/EOI)

Scaling the Data Points

All data values we've talked about so far have been 8 bit numbers corresponding to the 2432's digitizing system. To convert these numbers into voltage values requires a simple calculation based upon information from the Waveform Preamble.

The Waveform Preamble is made up of various fields that provide information about where the trigger point is located in the record, where the ground value is in relation to data value 0, what the sample rate is, and what the voltage increment between points is.

The Waveform Preamble can be sent to the 2432 to scale waveform data sent to a reference memory or it can be requested from the 2432, by using the 'WFMPRE?' query, to provide the scaling on the waveform currently selected by the DATA SOURCE command. If you need more information after reading the following discussion refer to the "Waveform Commands" section of Appendix A.

Sample Conversion to Volts

One of the most frequent uses for Waveform Preamble information is to convert the waveform values from either Signed Integer or Positive Integer into voltage values. To do this you need to know the voltage between each waveform value (vertical scaling) and where ground is located in relationship to the waveform values.

The vertical scaling value comes from the YMULT portion of the preamble and is the voltage between any two adjacent waveform points. The location of ground comes from the YOFF value in the preamble and is the position of ground relative to center screen. For example, YOFF would be −25 if the ground value were 1 division below center screen. The range of YOFF is from −2500 to +2500 with a resolution of 0.25. This large range is available because the 2432 can vertically position waveforms 10 divisions above center screen and 10 divisions below. It can also vertically expand averaged waveforms up to 10 times. So with 25 waveform points per division the potential range of numbers is 10 × 10 × 25 = 2500. The resolution of 0.25 is possible since the 2432 can vertically position waveforms with a 10 bit resolution or 4 times closer than the digitizer. This is just background information, all you need to do is plug the YOFF and YMULT numbers into the formulas below.

The general idea behind converting waveform points to voltage values is to normalize the waveform point relative to ground and then apply the vertical scaling. Here is the formula for Signed Integer data representation:

Voltage = (point value − YOFF) × YMULT

Here is the formula for Positive Integer representation (it's the same calculation, except for converting Positive Integer to Signed Integer by subtracting 127):

Voltage = (point value − 127 − YOFF) × YMULT

Lets actually try these formulas for a hypothetical waveform point 1 division below center screen and scale it using this sample Waveform Preamble:

```
WFMPRE WFID: "CH1 DC 1V 10US NORMAL", NR.PTS:1024,
   PT.OFF:512, PT.FMT:Y, XUNIT:SEC, XINCR:2.000E−7,
   YMULT:4.000E−2, YOFF:2.800E+1, YUNIT:V, BN.FMT:RI,
   ENCDG:BINARY
```

The BN.FMT field tells us we are dealing with Signed Integer data (RI) so the waveform point value is −25 (25 waveform points per division). The YMULT field is the voltage per division (1 volt in this case) divided by the number of waveform points per division (25), 4.0e−2 volts per point. Inserting the correct values into the first formula gives us:

Voltage = (−25 − 28) × 4.000E−2 = −2.12 Volts

This corresponds with a waveform point 1 division below center screen (−25 points / 25 points per division = −1 division) with the ground reference being slightly more than one division above center screen (28 points / 25 points per division = 1.12 divisions).

Waveform Acquisition and Conversion Programming Example

Here is a sample program, written in 4041 BASIC, that acquires a waveform and converts the values to voltages. The first example reads data in Signed Integer format (RIBINARY). The two program segments below the first example both read different formats and handle the data in different ways.

```
10  scope = 1
20  open #scope:" gpib(pri=1,eom=<0>):"
30  integer i
40  integer wfm(1024)
50  dim voltage(1024)
60   !===========================
70   ! Set up the 2432 for a waveform transfer.
80   ! Turn path off so all we get are the waveform
85   ! points. The data will be encoded in the RIBINARY
90   ! format. Then send a curve query to the 2432
100  !===========================
110 print #scope:" path off;data encdg:ribinary,source:ch1"
120 print #scope:" curve?"
130  !===========================
140  ! Now get the data values with the 'input'
145  ! statement. The 'using " 8%" ' part can interpret
150  ! the Entire Binary Waveform Format automatically.
160  ! Every controller will have a different way of
170  ! doing this. The idea is to get the waveform points
180  ! into a numerical array.
190  !===========================
200 input using " 8%"  #scope:wfm
210  !===========================
220  ! Now input the YOFF and YMULT scaling values.
230  ! Since path is still off, we can input these
240  ! values directly, without having to parse out
245  ! the " header" .
250  !============================
260  print #scope:" wfmpre? yoff"
270  input #scope:yoff
280  print #scope:" wfmpre? ymult"
290  input #scope:ymult
300  !============================
310  ! Now convert the data points to voltage values
320  ! using the formula we were given.
330  !============================
340  for i = 1 to 1024
350   voltage(i) = (wfm(i) − yoff) × ymult
360  next i
```

This example reads Positive Integer data and treats the waveform data as a string before creating the numeric array used in the above example. The following code segment replaces line 200 in the example above. In addition change "ribinary" to "rpbinary" on line 110 and replace the equation on line 350 with the one for Positive Integers. The "seg$" function takes a string and "segments" it. The first number is the start location and the second number is the length. The "asc" function converts an ASCII byte into its ASCII decimal equivalent, thus 'asc(" b")' would give the value 98.

```
200 dim wfm$ to 1100
201 input #scope:wfm$
202 for i = 1 to 1024
203   temp$ = seg$(wfm$,i+3,1)
204   wfm(i) = asc(temp$)
205 next i
```

This example uses ASCII encoded waveform data instead of RIBINARY and will replace line 200 in the first example. Also replace "ribinary" with "ASCII" on line 100. The "valc" routine takes a string and finds the first number starting from a specified location ("startloc" in the example below). The "valc" function is called for each waveform value, each time keeping track of where we are in the string. The second function, '" ask(" chpos")', is what keeps track of where the last number was found. It would be fairly easy to emulate this with other controller languages.

```
200 dim wfm$ to 6000
201 input #scope:wfm$
202 startloc = 1
203 for i = 1 to 1024
204   temp$ = seg$(wfm$,i+3,1)
205 wfm(i) = valic(wfm$,startloc)
206 startloc = ask(" chpos" )
207 next i
```

Appendix C

Event Tables

2432 Status Bytes

Title	Binary[a]	Decimal				Priority	
		RQS Off		RQS On		RQS Off	RQS On
		Idle	Busy	Idle	Busy		
No Status To Report	000X 0000	0	16	0	16	2	1
Power On	0R0X 0001	1	17	65	81	2	9
Operation Complete	0R1X 0010	2	18	66	82	2	3
User Request	0R1X 0011	3	19	67	83	2	8
Command Error	0R1X 0001	33	49	97	113	2	7
Execution Error	0R1X 0010	34	50	98	114	2	6
Internal Error	0R1X 0011	35	51	99	115	2	5
Execution Warning	0R1X 0101	37	53	101	117	2	4
Transmit Request	1R0X 0011	131	147	195	211	2	8
Transmit Aborted	1R0X 0100	132	148	196	212	2	8
Menuoff Pushed	1R0X 0101	133	149	197	213	2	8
Fatal Error	1R1X 0011	---	---	227	243	--	10

Device Dependent Bit
RQS Bit
Error Bit
Busy Bit

[a] "R" is set to 1 if the GPIB and RQS are on; otherwise; it is 0.

"X" is the Busy Bit and will be set if the 2432 is busy at the time the status byte is read. Anytime the 2432 is doing something for which the OPC SRQ can be sent (calibration or self test, single sequence, Save-On-Delta, or plotting) the bit will be sent true (1); otherwise, it will be a 0.

Command Error Events

NOTE

Command Errors are issued when a GPIB "grammatical" error has been made. Check the spelling and structure of the input strings. Set CER to ON to receive SRQ's when any of these events occurs. If CER is OFF the 2432 will not assert SRQ.

Code	Description
108	Checksum error in CURVE transfers.
109	Count = 0 or EOI set on byte count.
151	Symbol or number too long.
152	Invalid character or control character input.
153	EOI set on back slash.
154	Invalid number input.
155	EOI set on string character before ending quote.
156	Symbol not found.
157	Command or query argument is illegal is this syntax.
158	Character should be a colon.
159	Valid symbol, but not a legal header.
160	Character should be a comma, a semicolon, or EOI.
161	Too many query arguments.
162	Command only. May not be sent as a query.
163	Query only. May not be sent as a command.
164	EOI asserted before waveform was completed.
165	Incorrect word string input.
166	Number expected on incoming ascii waveform.
167	Comma expected on incoming ascii waveform.
168	Incoming ascii waveform has more than 1024 data points.
169	Illegal LLSET string.

Execution Error Events

NOTE

Execution errors are issued when a particular scope setting doesn't allow the current command to be executed the way the user would like. Set EXR to ON to receive SRQ's when any of these events occurs. If EXR is OFF the 2432 will not assert SRQ.

Code	Description
203	I/O buffers full, output dumped.
250	Selected recall memory is unset.
251	Selected reference memory is empty.
252	Waveform requested via GPIB is not valid or available.
253	Too many numbers were sent in (stack overflow).
254	No Video Option installed when SETTV commands issued.
255	Target selected for cursors is not displayed.
256	Clear overload condition before changing to 50 Ω coupling.
257	Waveform selected for reference source is not valid.
259	Envelope mode not available in Add or Mult.
260	No cal cmds while front panel is doing cal.
261	No sequence by that name to delete.
262	Cant save sequence—out of memory.
263	Cant send a partial waveform to an empty ref.
264	Not enough edge to work with.
265	Asked for rise time but no rising edge.
266	Asked for fall time but no falling edge.
267	These two measurements need matching Sec/Div's.
268	Mesial, distal, or proximal value is invalid.
269	Repet waveform not filled when measurement requested.
270	No measurements during live Roll—goto Save first.
271	No delay waveform measurements w/o Delta Delay on.
272	RMS measurement invalid due to 2432 internal overflow.
273	There is no envelope in multiply mode.
274	There is no envelope in add mode.
275	Sequencer currently active—new sequence commands not accepted.

Internal Errors

NOTE

Internal Errors are issued when something has happened to the hardware of the 2432 that the controller might like to know about. Set INR to ON to receive SRQ's when any of these events occurs. If INR is OFF the 2432 will not assert SRQ.

Code	Description
330	Cal execute command returns with FAIL.
331	A 50-Ω overload occurred. Input coupling switched to DC.

System Messages

NOTE

System Messages are issued to inform the controller of bus system management events. There is no way to mask these events except by setting RQS to OFF. The event 459 indicates that the 2432 is currently asserting SRQ on the bus and the controller must read the status byte out before reading the event code.

Code	Description
401	2432 was just powered on.
459	There is an SRQ pending.

User Request Events

NOTE

User Request events are issued when any of the bezel buttons on the 2432 front panel are pushed. The MENUOFF command needs to be issued before these events will be reported. This allows the user to write custom text to the screen and monitor front panel responses. Set USER to ON to receive SRQ's when any of these events occurs. If USER is OFF the 2432 will not assert SRQ.

Code	Description
450	Menu key #1 was pushed (leftmost) .
451	Menu key #2 was pushed.
452	Menu key #3 was pushed.
453	Menu key #4 was pushed.
454	Menu key #5 was pushed (rightmost).

Probe Identify Events

NOTE

Probe Identify events are reported by the 2432 when the probe identify feature found on certain probes is used. This is done manually by grounding the outer probe code ring on the front panel input BNC. Set PID to ON to receive SRQ's when any of these events occurs. If PID is OFF the 2432 will not assert SRQ.

Code	Description
455	CH1 probe identify was used.
456	CH2 probe identify was used.
457	EXT1 probe identify was used.
458	EXT2 probe identify was used.

Operation Complete Events

NOTE

Operation Complete events are issued when the controller needs to know when the 2432 has completed a task. Set OPC to ON to receive SRQ's when any of these events occurs. If OPC is OFF the 2432 will not assert SRQ.

Code	Description
461	Single Sequence has completed.
462	Save-On-Delta has detected a difference and gone to Save.
463	A print or plot is complete.
464	Current cal command started with an EXECUTE is done.
465	Step command is done.
466	Complete sequence is done.
467	Autoset search is complete.

Execution Warning

NOTE

Execution Warnings are issued when the command received has been done but the result might not be what the user wanted to see. Set EXW to ON to receive SRQ's when any of these events occurs. If EXW is OFF the 2432 will not assert SRQ.

Code	Description
540	RMS measurements need at least 1 period.
541	Not enough signal to do a timing measurement.
542	Measurement results on an Envelope waveform are different.
543	Histogram is suspect for this measurement.
544	Waveform has some points off the top.
545	Waveform has some points off the bottom.
546	Waveform has points off the top and bottom.
547	Rising edge has too few points for optimal accuracy.
548	Falling edge has too few points for optimal accuracy.
549	No detectable delay found.
550	Only Delay 1 will be displayed if in Average.

Execution Warning (cont)

Code	Description
551	Word Recognizer Probe is disconnected.
552	A and B Sec/Div are locked together.
553	More than 1024 binary points were sent; excess discarded.
554	No absolute cursors in slope.
555	A trigger coupling and logsrc changed.
556	A trigger source change forced logsrc to off.
557	No Average in Roll. Acquire mode or A Trigger mode changed.
558	No live vertical expansion unless averaging. Gain changed.
560	Volts/Div value requested was rounded or limited.
561	Variable Volts/Div value requested was limited.
562	Vertical Position value requested was limited.
563	A or B trigger level was limited.
564	Trigger Holdoff value requested was limited.
565	Horizontal Position value requested was limited.
566	A or B Sec/Div setting requested was rounded.
567	Delay by Events events number was limited.
568	Delay by Time Delay value was limited.
569	Number of Envelopes requested was rounded.
570	Number of Averages requested was rounded.
572	Cursor reference value requested was rounded.
573	Horizontal position value (XPOS) for cursors was limited.
574	Vertical position value (YPOS) for cursors was limited.
575	Intensity value requested was limited.
576	Line number of screen text message was limited.
578	The XINCR value was rounded or limited.
579	The PTOFF value was rounded or limited.
580	The YMULT value was rounded or limited.
581	Video Option line number requested was limited.
582	Trigger position number requested was limited.
583	An ascii data point was rounded to fit into 127 to −128.
584	Waveform data level value requested was limited.
585	Start or Stop number was changed.
586	The YOFF value was limited.
587	Extexp value requested was limited.
588	Hysteresis number requested was rounded.
589	Attribute number requested was rounded.

Device Dependent Message

NOTE

Device Dependent messages are issued when the front panel user of the 2432 has done something that the controller might want to know about. Set DEVDEP to ON to receive SRQ's when any of these events occurs. If DEVDEP is OFF the 2432 will not assert SRQ.

Code	Description
650	Waveform was requested from front panel.
651	Waveform transmission was aborted from front panel.
652	MENUOFF command was executed or front panel button pushed.

Fatal Error

NOTE

A Fatal Error is issued when something completely unexpected happens inside the 2432. This normally is caused by a hardware failure. There is no way to prevent this error from being reported except by turning RQS to OFF.

Code	Description
750	Fatal error.

D

Appendix D

GPIB Concepts

The IEEE 488 standard defines three aspects of an instrument's interface:

Mechanical The connector and the cable.

Electrical The electrical levels for logic signals and how the signals are sent and received.

Functional The tasks that an instrument's interface can perform, such as sending data, receiving data, triggering the instrument, etc.

Using this interface standard, instruments can be designed to have a basic level of compatibility with other instruments that meet the standard.

Mechanical

IEEE 488 specifies a standard connector and cable for linking instruments to ensure that GPIB instruments are pin-compatible. The GPIB connector (Figure D-1) has 24 pins, with 16 assigned to specific signals and 8 to shields and grounds.

Allowable Configurations

Instruments can be connected to the GPIB in linear or star configurations, or in a combination of both. A linear hookup is one where the GPIB cable is strung from one instrument to the next. A star setup is where one end of all the GPIB cables in the system are attached to one instrument.

Restrictions

Instruments connected to a single bus cannot be separated by more than two meters for each instrument on the bus. In addition, the total cable length of the bus cannot exceed 20 meters.

To maintain proper electrical characteristics on the bus, a device load must be connected for every two meters of cable length, and at least two-thirds of the instruments connected to the bus must be powered on. (For more information, see IEEE Standard 488-1978). Although instruments are usually spaced no more than two meters apart, they can be separated further if the required number of device loads are grouped at any point on the bus.

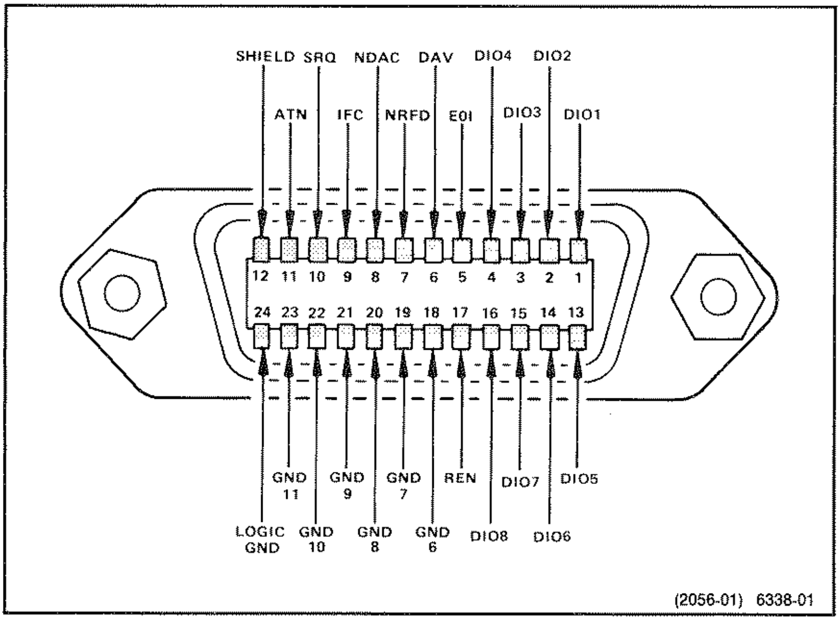

Figure D-1. GPIB connector.

Electrical Elements

The IEEE 488-1978 standard defines the voltages and current values required at connector nodes. All specifications are based on the use of TTL technology. A '0' logical state corresponds to voltages ⩾ 2.0 volts and ⩽ 5.2 volts (HI state). A '1' logical state corresponds to voltages ⩾ 0 volts and ⩽ 0.8 volts (LO state).

Messages can be sent as either active or passive true signals. Passive true and false signals occur in the HI state and must be carried on a signal line using open collector devices.

Functional Elements

Interface functions provide the facilities through which instruments send, process, and receive messages. IEEE 488 defines ten different interface functions. They are described in the following paragraphs. The abbreviations for these functions, which are in parenthesis, are taken from the IEEE 488 standard and are commonly used when describing the functions.

Acceptor Handshake (AH)

The AH function provides an instrument with the capability to guarantee proper reception of data. The AH function delays initiation or termination of a data transfer until the instrument is ready to receive the next data byte.

Source Handshake (SH)

The SH function works together with the AH function on a listening device to guarantee proper transfer of messages. The SH function controls the initiation and termination of the transfer of data bytes.

Listener (L) and Listener Extended (LE)

The L and LE functions provide an instrument with the capability to receive device-dependent data over the interface. This capability exists only when the instrument is addressed to listen. The L function uses a 1-byte address; the LE function uses a 2-byte address. In all other aspects, the capabilities of both functions are the same. The 2432 implements only the 1-byte address.

Talker (T) and Talker Extended (TE)

The T and TE functions provide an instrument with the capability to send device-dependent data over the interface. This capability exists only when the instrument is addressed to talk. The T function uses a 1-byte address; the TE function uses a 2-byte address. In all other respects, the capabilities of both functions are the same. The 2432 implements only the 1-byte address.

Device Clear (DC)

The DC function provides an instrument with the capability to be cleared or initialized, either individually or as part of a group of instruments.

Device Trigger (DT)

The DT function provides an instrument with capability to have its basic operation started, either individually or as part of a group of instruments.

Remote/Local (RL)

The RL function provides an instrument with the capability to select between two sources of input. The function indicates to the instrument that either input information from its front panel switches (local) or corresponding information from the GPIB interface is to be used.

Service Request (SR)

The SR function provides an instrument with the capability to request service from the controller that is in charge of the interface.

Parallel Poll (PP)

The PP function provides an instrument with the capability to send a status message to the controller without being addressed to talk.

Controller (C)

The C function provides an instrument with the capability to send device addresses, universal commands, and addressed commands to other instruments over the interface. It also provides the capability to determine which instruments require service. The 2432 does not implement the C function.

Instrument Addresses

Every instrument on the bus has one or more addresses. The types of addresses an instrument can have are: a primary address, a listen address, a talk address, and a secondary address.

Primary Address

Every instrument connected to the bus has a unique primary address. You can set the primary address of most instruments. No two instruments on the GPIB can have the same primary address. Valid primary addresses range from 0 to 30.

Listen Address

Every instrument that can receive device-dependent messages over the bus has a unique listen address. When an instrument senses its listen address on the bus with the ATN signal line asserted, the instrument prepares to receive data sent over the bus. When ATN is unasserted, the instrument receives data. An instrument's listen address is determined by adding 32 to its primary address. Thus an instrument with primary address 19 has a listen address of 19 + 32 = 51. Any number of instruments on the bus may be addressed to listen at the same time.

Talk Address

Every instrument that can send data over the bus has a unique talk address. When an instrument senses its talk address on the bus with ATN asserted, the instrument prepares to send data over the bus. When ATN is then unasserted, the instrument sends data. Only one instrument on the bus may be addressed to talk at a time. When an instrument addressed to talk senses another instrument's talk address on the data lines with ATN asserted, the first instrument automatically untalks itself. An instrument's talk address is determined by adding 64 to its primary address. Thus, an instrument with primary address 19 has a talk address of 19 + 64 = 83.

Secondary Address

Some instruments support a special addressing scheme called secondary addressing. Secondary addresses range from 96 to 126. Not all IEEE 488 compatible instruments support secondary addressing. The 2432 does not implement secondary addressing.

GPIB Buses

The 16 GPIB signal lines can be divided into three buses (Figure D-2): the data bus, the management bus, and the handshake bus. The data bus is composed of eight signal lines that carry data to be transferred on the GPIB. The management bus is composed of five signal lines that control data transfers. The handshake bus is composed of three signal lines that synchronize data transfers between instruments.

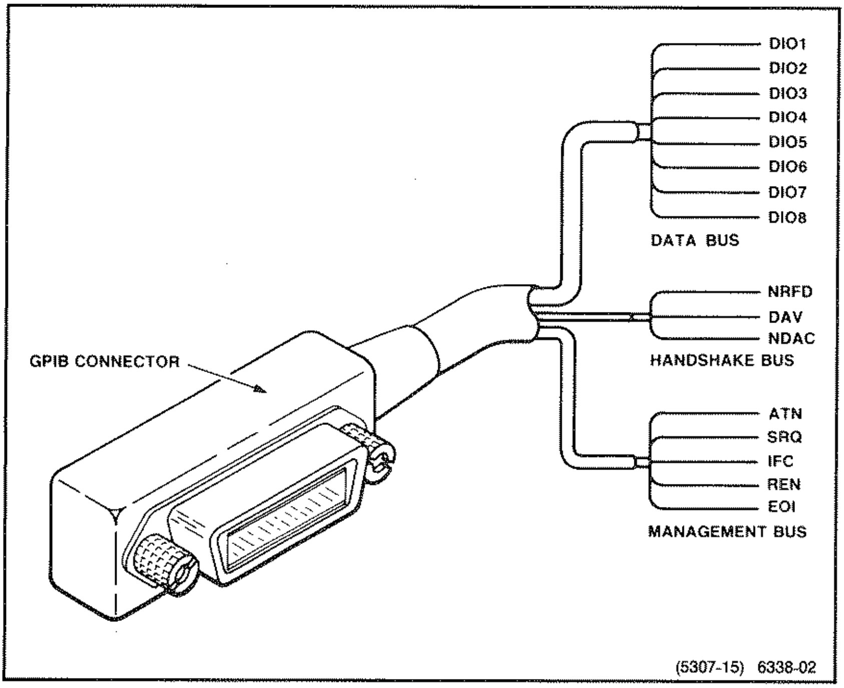

Figure D-2. GPIB Buses.

Data Bus

The data bus contains eight bidirectional signal lines, named DI01 through DI08. One byte of information is transferred over the bus at a time. DI01 carries the least significant bit of the byte and DI08 carries the most significant bit. Each byte of information transferred on the data bus represents either a command, a device address, or a device-dependent message. Data bytes can be formated in ASCII or in a device-dependent binary code.

Management Bus

The management bus is a group of five signal lines (ATN, EOI, IFC, REN, and SRQ) used in managing data transfers on the data bus. The definitions for these lines are listed in the following paragraphs.

ATTENTION (ATN). The ATN management line is activated by the controller to send universal commands and addressed commands, and to designate instruments as talkers and listeners for an upcoming data transfer. When ATN is asserted, messages sent on the data bus are interpreted as commands or addresses. When ATN is unasserted, messages sent on the data bus are interpreted as device-dependent messages. Only instruments that have been addressed by the controller to talk or listen can take part in a device-dependent data transfer.

END OR IDENTIFY (EOI). The EOI signal line can be used by any talker to indicate the end of a data transfer sequence. Talkers that use EOI activate the EOI line simultaneously with the last byte of information as it is transferred.

INTERFACE CLEAR (IFC). When the IFC line is asserted by the controller three things happen. First, all talk-addressed and listen-addressed instruments on the bus are unaddressed. Second, the controller issuing the IFC assumes controller-in-charge status. Last, all instruments are taken out of serial poll mode (same as sending SPD, which is described in "Addressed Commands" later in this section). Only the system controller can activate the IFC line.

REMOTE ENABLE (REN). The REN signal line is activated by the system controller to give all instruments on the bus the capability of being placed under remote (program) control. When the REN signal line is activated, instruments that receive their listen addresses on the data bus accept and execute commands from the controller-in-charge. When REN is deactivated, all instruments on the bus revert to front-panel control.

SERVICE REQUEST (SRQ). The SRQ line can be activated by any instrument on the bus to request service from the controller. The controller responds (if programmed to do so) by serial polling all instruments on the bus in order to find the instrument requesting service. The SRQ line is deactivated when the instrument requesting service is polled. The 2432 asserts SRQ whenever there is a change in its status to report to the controller. SRQs are only asserted if the corresponding SRQ mask is enabled.

Handshake Bus

Three lines (NRFD, DAV, and NDAC) comprise the handshake bus. These three lines control the sequence of operations each time a byte is transferred on the data bus. This sequence is not under user control, but information about the three transfer lines is presented here for completeness.

NOT READY FOR DATA (NRFD). An active NFRD line indicates that one or more of the listeners is not ready to receive the next data byte. When the NRFD line goes inactive (indicating all listeners are ready to receive data), the talker places the next data byte on the data bus and activates the DAV signal line. The 2432 uses the NRFD line to hold off new data when its input buffer is full. As characters are removed from the buffer, the 2432 will again be ready to accept new data.

DATA VALID (DAV). The DAV line is activated by the talker shortly after it places a valid data byte on the data bus. This tells each listener to capture the data byte currently on the bus.

NOT DATA ACCEPTED (NDAC). The NDAC line is held active by each listener until the listener captures the data byte on the data bus. When all listeners have captured the data byte, NDAC goes inactive. This tells the talker to take the byte off the data bus.

GPIB Communication Protocol

Each instrument on the bus at any given time may be either a controller, a talker, or a listener. Some instruments have two or even three of these capabilities. For example, some instruments are talk-only, others are listen-only, others can talk and listen, others can talk, listen, and control.

Controllers

Controllers are instruments that assign talk and listen status to other instruments on the bus. Since only one instrument can talk at a time, and since it is seldom desirable for every instrument to listen, a controller is needed to designate which instrument is to talk and which instruments are to listen during any data transfer. There are two kinds of controllers on the GPIB: the system controller and the controller-in-charge.

A GPIB network can have at most one instrument acting as the system controller and one instrument acting as the controller-in-charge. The system controller and the controller-in-charge may be (and often are) the same instrument.

A GPIB configuration may include any number of instruments capable of acting as the system controller or controller-in-charge, subject only to the limitations on the total number of instruments allowed on the bus.

Once a GPIB network has been set up, system control cannot be passed from instrument to instrument; controller-in-charge status can. Once an instrument is the system controller, no other instrument on the bus can assume that role without your reconfiguring the GPIB network.

Only the system controller can affect the status of the Interface Clear (IFC) and Remote Enable (REN) management lines. Asserting the IFC line makes the system controller the controller-in-charge, and untalks, unlistens, and disables serial polling for all instruments.

Asserting REN enables the remote operation of instruments by the controller. Asserting REN does not automatically put all instruments on the bus into the remote state, but simply allows the controller to put them into this state. The LLO Universal Command must be issued to get the 2432 to go to the remote with local lockout state and lock the front panel.

Any instrument acting as controller-in-charge may pass control to any other instrument on the bus capable of assuming control.

Talkers

A talker is an instrument that has sensed its talk address on the data lines with ATN asserted. When a talker is addressed to talk, it sends data via the data lines when ATN is unasserted.

Only one instrument on the bus may be addressed to talk at a time. When an instrument addressed to talk recognizes another instrument's talk address on the data lines with ATN asserted, the first instrument automatically untalks itself. (When an instrument is untalked, it does not place data on the data lines when ATN is unasserted.)

An instrument may also untalk itself when it recognizes its listen address on the data lines with ATN asserted. In this case, the instrument becomes a listener when ATN is unasserted.

Listeners

A listener is an instrument that has sensed its listen address being sent on the data lines with ATN asserted. After an instrument is addressed to listen, it accepts information on the data lines when ATN is unasserted.

Any number of instruments on the bus may be addressed to listen at the same time. When an instrument previously addressed to listen senses its talk address being sent on the data lines with ATN asserted, the instrument may unlisten itself and become a talker when ATN is unasserted.

Universal Commands

Universal commands are commands that are obeyed by all instruments on the bus that have the appropriate subsets of the IEEE 488 interface functions implemented. The controller sends universal commands by placing certain values on the data lines with ATN asserted. The universal commands include: Device Clear (DCL), Local Lockout (LLO), Serial Poll Disable (SPD), and Serial Poll Enable (SPE). Also included in this description are Unlisten (UNL), and Untalk (UNT). Although not commands in the strict sense, the values of UNL and UNT act like universal commands when sent on the data lines with ATN asserted.

Device Clear (DCL)

The DCL command clears (initializes) all instruments on the bus that have a DC1 or DC2 subset of the DC interface function. The DCL message reinitializes communication between the 2432 and the controller. In response to DCL, the 2432 clears any input and output messages as well as any unexecuted control settings. Any errors and events waiting to be reported, except power-on, are also cleared. If the SRQ line is asserted for any reason other than power-on, it becomes unasserted when DCL is received. Fastxmit mode cannot be aborted with DCL. To send the DCL command, the controller places the value 20 on the data lines with ATN asserted.

Local Lockout (LLO)

The LLO command locks out the front panels of all instruments on the bus that have an RL1 subset of the RL interface function. (Devices with RL0 or RL2 subsets of the RL interface function ignore LLO.) After receiving the LLO command and its listen address, an instrument ignores any subsequent inputs from front panel control switches with corresponding remote controls, and only obeys commands coming over the GPIB interface. To lock out the 2432 front panel, LLO must be sent with REN asserted. To keep the 2432 front panel locked out, REN must remain asserted. To unlock the 2432 front panel, unassert REN. To send the LLO command, the controller places the value 17 on the data lines with ATN asserted.

Serial Poll Disable (SPD)

The SPD command returns all instruments on the bus from the serial poll enabled state. To send the SPD command, the controller places the value 25 on the data lines with ATN asserted.

Serial Poll Enable (SPE)

The SPE command puts all instruments that are on the bus with an SR1 subset of the SR interface function into the serial poll enabled state. In this state, each instrument sends the controller its status byte, instead of the its normal output, after the instrument receives its talk address on the data lines with ATN asserted. To send the SPE command, the controller places the value 24 on the data lines with ATN asserted.

Unlisten (UNL)

The UNL command takes all listen-addressed instruments on the bus out of the listen-addressed state. To send the UNL command, the controller places the value 63 on the data lines with ATN asserted.

Untalk (UNT)

The UNT command takes any talk-addressed instrument on the bus out of the talk-addressed state. To send the UNT command, the controller places the value 95 on the data lines with ATN asserted.

Addressed Commands

Addressed commands are commands that are sent to specific instruments on the bus. The controller sends addressed commands by placing certain values on the data lines with ATN asserted. Addressed commands include Group Execute Trigger (GET), Go to Local (GTL), Parallel Poll Configure (PPC), Selected Device Clear (SDC), and Take Control (TCT). All of the addressed commands, except TCT, require that the instrument receiving the command be listen-addressed. The TCT command requires that the instrument be talk-addressed.

Group Execute Trigger (GET)

The GET command causes all listen-addressed instruments incorporating a DT1 subset of the DT interface function to begin operation (for example, for measurement instruments to make their measurements, output devices to output their signals, and so on). To send the GET command, the controller places the value 8 on the data lines with ATN asserted. The 2432 implements GET. See the DT command in appendix A for more information on what the 2432 will do upon a receipt of GET.

Go To Local (GTL)

The GTL command causes all listen-addressed instruments to obey incoming commands from their front panel control switches. These instruments may store, but not respond to, commands coming through the GPIB interface until the instrument is once again listen-addressed. To send the GTL command, the controller places the value 1 on the data lines with ATN asserted. The 2432 must first be returned to the local state by unasserting the REN line. Sending the GTL command, without unasserting REN will not unlock the 2432 front panel.

Selected Device Clear (SDC)

The SDC command clears or initializes all listen-addressed instruments. To send the SDC command, the controller sends a value of 4 on the data lines with ATN asserted.

Take Control (TCT)

The TCT command passes controller-in-charge status to a talk-addressed instrument. To send the TCT command, the controller places the value 9 on the data lines with ATN asserted. The 2432 cannot become a controller, so does not understand TCT.

Serial Polling

Serial polling is a means of reading serially the individual status messages of all instruments configured and enabled to respond to a serial poll.

Status Bytes

Each instrument on the GPIB incorporating the SR1 subset of the SR interface function has an eight-bit status byte. The status byte's contents describe (by means of a device-dependent code) the instrument's status. The 2432 incorporates the SR1 subset of the SR interface function.

Requesting Service

The coding of an instrument's status byte is almost entirely up to the instrument's designer, with one restriction: bit 7 (the second-most significant bit) is reserved to indicate whether or not an instrument is requesting service from the controller-in-charge.

Bit 7 of the status byte is known as the RQS or requesting service bit. A value of 1 indicates that an instrument is requesting service from the controller-in-charge. A value of 0 indicates that the instrument is not requesting service. To request service, an instrument must set the RQS bit of the status byte to 1 and then assert SRQ.

Conducting Serial Polls

The controller can be programmed to generate an interrupt and to serially poll each instrument on the bus, to determine which instrument is requesting service, when the controller recognizes an instrument on the bus asserting SRQ. The controller can be programmed to conduct a serial poll any time; it does not have to receive a request for service from an instrument on the bus.

The controller conducts the poll by sending the SPE (serial poll enable) command, followed by a sequence of listen addresses. As each listen address is received, the instrument that is listen-addressed sends its status byte to the controller. The controller can then check the status byte to see if the RQS bit is set. Receiving the status byte from the instrument requesting service clears the RQS bit of that instrument (sets the bit back to 0).

When the instrument asserting SRQ is discovered, the controller usually terminates the serial poll (by sending the SPD command), and transfers control to a user-defined SRQ handler routine for the instrument requesting service. Reading the instrument's status byte clears the instrument's RQS bit. However, the factors that caused the instrument to request service in the first place must be handled, or the instrument may simply request service again and again.

Appendix E

Init Settings and Power Up States

The INIT command of the 2432 is used to return the oscilloscope to a known operating state. Initializing the 2432 provides a basic setup from which to begin programming the 2432. See the INIT command description in the "Miscellaneous Commands" section of Appendix A for more information.

The INIT command has several arguments which determine what type of initialization is performed. The PANEL argument causes a normal initialization of the 2432. For a full definition of the normal reset status, refer to the Operators Manual. The GPIB argument causes an initialization of command parameters unique to the GPIB. For a list of these GPIB parameters and there reset status, refer to the "INIT Command" in Appendix A.

When the 2432 first powers up, several instrument states are initialized. These are listed below in the form of commands that a controller would use to select the same settings.

```
DEBUG OFF;  USER OFF;  PID OFF;  OPC ON;  CER ON;
EXW ON;  EXR ON;  INR ON;  DEVDEP ON;  LOCK LLO;
LONG ON;  PATH ON;  FASTXMIT OFF,1,ENCDG:RIBINARY;
WFMPRE BN.FMT:RI;  SETUP FORCE:ON,ACTION:0;
DT OFF;  START 256;  STOP 512;  HYSTERESIS 5;
DIRECTION PLUS;  FORMAT OFF;  LEVEL 0
```

F

Appendix F

ASCII and 2432 Character Charts

The following table shows the character set the 2432 will display. The large characters in the middle of the boxes are the characters that show up on the 2432 CRT. The smaller characters and numbers in the corners are the octal value (in the top left), the hex value (in the bottom left), the decimal value (in the bottom right), and the GPIB equivalent code (in the top right).

The GPIB equivalent code designates what character string is sent by the 2432 (or should be sent by the controller) to designate the 2432 display character. You will notice that GPIB equivalent codes are shown for the 2432 characters that do not correspond to their ASCII counterparts. The reason for the GPIB equivalent code is that most of the 2432 characters are not standard ASCII characters and thus cannot be displayed on conventional terminals.

As an example, suppose you wish to send the string HELLO to line 10 on the 2432. You would send the command 'MESSAGE 10:" HELLO" '. If you wanted to send the same message but have the letters underlined, you would send the command 'MESSAGE 10:" \h\e\l\l\o" ' to the 2432. It is important to remember to use the GPIB equivalent code if it is shown.

B7 B6 B5	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
BITS B4 B3 B2 B1	SPECIAL		NUMBERS SYMBOLS		UPPER CASE		UNDERLINED	
Ø Ø Ø Ø	0 \0 — 0 0	20 \@ Ø 10 16	40 SP 20 32	60 Ø 30 48	100 @ 40 64	120 P 50 80	140 \` @ 60 96	160 \p P 70 112
Ø Ø Ø 1	1 \1 ⌖ 1 1	21 \A 1 11 17	41 \! 2 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 \a A 61 97	161 \q Q 71 113
Ø Ø 1 Ø	2 \2 ▯ 2 2	22 \B 2 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 \b B 62 98	162 \r R 72 114
Ø Ø 1 1	3 \3 d 3 3	23 \C 3 13 19	43 \# d 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 \c C 63 99	163 \s S 73 115
Ø 1 Ø Ø	4 \4 k 4 4	24 \D 4 14 20	44 \$ (blank) 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 \d D 64 100	164 \t T 74 116
Ø 1 Ø 1	5 \5 % 5 5	25 \E 5 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 \e E 65 101	165 \u U 75 117
Ø 1 1 Ø	6 \6 z 6 6	26 \F 6 16 22	46 \& z 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 \f F 66 102	166 \v V 76 118
Ø 1 1 1	7 \7 s 7 7	27 \G 7 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 \g G 67 103	167 \w W 77 119
1 Ø Ø Ø	10 \8 Δ 8 8	30 \H 8 18 24	50 \(Δ 28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 \h H 68 104	170 \x X 78 120
1 Ø Ø 1	11 \9 ÷ 9 9	31 \I 9 19 25	51 \) ÷ 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 \i I 69 105	171 \y Y 79 121
1 Ø 1 Ø	12 \: * A 10	32 \J Ω 1A 26	52 * 2A 42	72 : 3A 58	112 J 4A 74	132 Z 5A 90	152 \j J 6A 106	172 \z Z 7A 122
1 Ø 1 1	13 \; m B 11	33 \K ⏚ 1B 27	53 + 2B 43	73 ; 3B 59	113 K 4B 75	133 \[[5B 91	153 \k K 6B 107	173 \{ [7B 123
1 1 Ø Ø	14 \< μ C 12	34 \L B LW 1C 28	54 , 2C 44	74 < 3C 60	114 L 4C 76	134 \\ \ 5C 92	154 \l L 6C 108	174 \\| | 7C 124
1 1 Ø 1	15 \= n D 13	35 \M ° 1D 29	55 - 2D 45	75 = 3D 61	115 M 4D 77	135 \]] 5D 93	155 \m M 6D 109	175 \}] 7D 125
1 1 1 Ø	16 \> p E 14	36 \N → 1E 30	56 . 2E 46	76 > 3E 62	116 N 4E 78	136 \^ ↑ 5E 94	156 \n N 6E 110	176 \~ ~ 7E 126
1 1 1 1	17 \? / F 15	37 \O ← 1F 31	57 / 2F 47	77 ? 3F 63	117 O 4F 79	137 _ ↓ 5F 95	157 \o O 6F 111	177 \DEL[a] H O 7F 127

[a] \DEL = \RUBOUT

KEY TO CHART

octal — 32 \J — GPIB code

Ω — 2430 character

hex — 1A 26 — decimal

(4918-35) 6338-03

ASCII & GPIB CODE CHART

B7 B6 B5 BITS	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
Ø Ø Ø Ø	0 NUL 0 0	20 DLE 10 16	40 0 SP 20 32	60 16 0 30 48	100 0 @ 40 64	120 16 P 50 80	140 0 ` 60 96	160 16 p 70 112
Ø Ø Ø 1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 1 ! 21 33	61 17 1 31 49	101 1 A 41 65	121 17 Q 51 81	141 1 a 61 97	161 17 q 71 113
Ø Ø 1 Ø	2 STX 2 2	22 DC2 12 18	42 2 " 22 34	62 18 2 32 50	102 2 B 42 66	122 18 R 52 82	142 2 b 62 98	162 18 r 72 114
Ø Ø 1 1	3 ETX 3 3	23 DC3 13 19	43 3 # 23 35	63 19 3 33 51	103 3 C 43 67	123 19 S 53 83	143 3 c 63 99	163 19 s 73 115
Ø 1 Ø Ø	4 SDC EOT 4 4	24 DCL DC4 14 20	44 4 $ 24 36	64 20 4 34 52	104 4 D 44 68	124 20 T 54 84	144 4 d 64 100	164 20 t 74 116
Ø 1 Ø 1	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 5 % 25 37	65 21 5 35 53	105 5 E 45 69	125 21 U 55 85	145 5 e 65 101	165 21 u 75 117
Ø 1 1 Ø	6 ACK 6 6	26 SYN 16 22	46 6 & 26 38	66 22 6 36 54	106 6 F 46 70	126 22 V 56 86	146 6 f 66 102	166 22 v 76 118
Ø 1 1 1	7 BEL 7 7	27 ETB 17 23	47 7 ' 27 39	67 23 7 37 55	107 7 G 47 71	127 23 W 57 87	147 7 g 67 103	167 23 w 77 119
1 Ø Ø Ø	10 GET BS 8 8	30 SPE CAN 18 24	50 8 (28 40	70 24 8 38 56	110 8 H 48 72	130 24 X 58 88	150 8 h 68 104	170 24 x 78 120
1 Ø Ø 1	11 TCT HT 9 9	31 SPD EM 19 25	51 9) 29 41	71 25 9 39 57	111 9 I 49 73	131 25 Y 59 89	151 9 i 69 105	171 25 y 79 121
1 Ø 1 Ø	12 LF A 10	32 SUB 1A 26	52 10 * 2A 42	72 26 : 3A 58	112 10 J 4A 74	132 26 Z 5A 90	152 10 j 6A 106	172 26 z 7A 122
1 Ø 1 1	13 VT B 11	33 ESC 1B 27	53 11 + 2B 43	73 27 ; 3B 59	113 11 K 4B 75	133 27 [5B 91	153 11 k 6B 107	173 27 { 7B 123
1 1 Ø Ø	14 FF C 12	34 FS 1C 28	54 12 , 2C 44	74 28 < 3C 60	114 12 L 4C 76	134 28 \ 5C 92	154 12 l 6C 108	174 28 | 7C 124
1 1 Ø 1	15 CR D 13	35 GS 1D 29	55 13 - 2D 45	75 29 = 3D 61	115 13 M 4D 77	135 29] 5D 93	155 13 m 6D 109	175 29 } 7D 125
1 1 1 Ø	16 SO E 14	36 RS 1E 30	56 14 . 2E 46	76 30 > 3E 62	116 14 N 4E 78	136 30 ^ 5E 94	156 14 n 6E 110	176 30 ~ 7E 126
1 1 1 1	17 SI F 15	37 US 1F 31	57 15 / 2F 47	77 UNL ? 3F 63	117 15 O 4F 79	137 UNT _ 5F 95	157 15 o 6F 111	177 DEL (RUBOUT) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

octal	25 PPU	GPIB code
	NAK	ASCII character
hex	15 21	decimal

Tektronix®
COMMITTED TO EXCELLENCE

REF: ANSI STD X3. 4-1977
IEEE STD 488-1978
ISO STD 646-1973

TEKTRONIX STD 062-5435-00 4 SEP 80

6338-04

Appendix G

Answers to Common Questions

Who Can I Call for Help?

Ask your local Tektronix Application Engineer or Sales Engineer for help. If you don't know who they are, you can call this toll free number for the location and phone number of the nearest field office. The number is (800) 547-1512 or if you are calling from Oregon then (800) 452-1877.

How Do I Unlock the Front Panel?

If the response to a 'LOCK?' query is 'LLO' then you need to strobe the remote enable line (REN) from HI to LO. Another way is to send the command 'LOCK OFF' to the 2432. Another way is to turn the 2432 off and then back on. It is also possible that the 2432 has locked its own front panel while doing some special operations. See the section on "What Do The GPIB Status LED's Mean".

What Does a 255 (FFh) in My Data Stream Mean?

It means the 2432 was talked with nothing to say. The most common cause is sending a query that is mis-typed and then asking for a response from the 2432. Because the 2432 didn't understand the question it can't send an answer. But instead of hanging the bus it sends the FFh. See Appendix H for more information.

How Do I Get Just the Answer From a 2432 Query?

Issue the command 'PATH OFF' to the 2432. With PATH set to ON the response to the 'CH1? POSITION' query might be 'CH1 POSITION:1'. With PATH set to OFF the response would be '1'. See "Asking the 2432 a Question" for more information.

How Do I Clear All SRQ's and Event Codes?

By sending the command 'INIT SRQ' to the 2432. This is usually done before waiting for the 2432 to complete a task. See "How to Use Service Requests" for more information. From the front panel, changing the GPIB address of the 2432 will also clear all SRQ's and event codes.

Can I Run Self-Cal From a Controller?

Yes. Send the following command 'TESTTYPE SELFCAL;EXECUTE' to the 2432. See the "Calibration and Diagnostics Commands" table in Appendix A for more information.

How Can I Get the Most Recent GPIB Related Status?

Use the GPIB status screen. Display it by pushing the OUTPUT front-panel button followed by the STATUS menu button.

How Do I Get All the Measurements at Once?

Send the 'VALUE?' query to the 2432. If no arguments follow the header then all measurement results will be returned. This is the GPIB's equivalent of the Snapshot. If you just wanted the risetime then send the 'VALUE? RISE' query to the 2432. Make sure that path mode is on ('PATH ON') before getting the measurements if you want to make sense of the information, otherwise you will get a lot of numbers with no measurement types associated with them.

How Do I Clear the Screen?

To clear all 16 lines of text send the 'MESSAGE CLRSTATE' command to the 2432. Use the 'MENUOFF' command to clear just the bottom 3 lines of readout.

How Do I Clear One Line of Text?

Send the command 'MESSAGE line#:" "' to the 2432. The "line#" field in this command indicates which line you want to clear. Remember, the lines are numbered from 1 to 16 with line number 1 at the BOTTOM of the display.

How Do I Write Text to the Screen?

Send the command 'MESSAGE line:" message" ' to the 2432. The line number of the line you want to clear is 'line' and '" message" ' is the actual text you want to appear on the screen.

How Do I Write My Own Menus and Handle Them?

First send the 'MENUOFF;USER ON' command to the 2432. This lets the scope know that there are custom text lines on the display and enables the issuing of User Request SRQ's when a menu button is pushed. Then write your text to the screen using the MESSAGE command. Whenever you push a menu button, the 2432 will send an SRQ. The corresponding event code will indicate which menu button was pushed. See "How to Use Service Requests" for more information. There is also a sample program in Appendix J that uses some menu text.

Why Do I Keep Getting Event 459, SRQ Pending?

You have SRQ's turned on (RQS ON) but are not handling them by serially polling the instrument. See "How to Use Service Requests" for more information.

How Do I Send a Sequence From One 2432 to Another?

Set the "from" oscilloscope to Talk Only mode with the SEND PRGM selection. Then set the "to" oscilloscope to Listen Only mode. Select the sequence to send by underlining it in the RECALL menu found by pushing the PRGM button. Then push the OUTPUT front-panel button followed by the SENDPRGM menu button to send the sequence. See "Establishing Communications" for more information on how to set up these modes.

How Do I Print Out a Status, Help, or Snapshot Screen?

You need to initiate the print or plot using a controller. See "Making Printer and Plotter Copies" for more information.

Why Does the Acquisition Restart When I Send Certain Commands?

Acquisitions restart because the 2432 needs to clean up when a mode changes. The interactions between various modes are numerous and often the only solution is to restart the acquisition between each mode change. Any volts/div, time/div and trigger system change along with the MENUOFF command will cause a restart. The MENUOFF command causes a restart because pushing that button on the front panel is the exit from calibration and diagnostics and all of the hardware registers in the 2432 are reset to a "normal" state before leaving.

Why Doesn't the Waveform I Sent to the 2432 Look Like the One I Took Out?

You probably took the waveform out with a different encoding than the 2432 was expecting when you sent it back. This happens when sending binary data back and forth. See the section of Appendix B that deals with waveform data being sent to the scope for more information.

Why are the Abbreviations of Symbols on the 2432 Not the Same as Other Tektronix Oscilloscopes?

The command sets are different between instruments. Commands are abbreviated so they are unique within a certain command set. When the command set is different, the abbreviations may also be different. To alleviate some of this problem, use full command and argument names instead of abbreviations.

Why Would I Use the INIT Command?

When setting up a test sequence it is often much faster to reset the 2432 to a known state and then change a few parameters than it is to find out what state the instrument is in and then change to a new one. The INIT command is used to set the 2432 to the known state. See the Operators Manual for a description of that state.

Do Sequences Go Away When I Turn Power Off?

No. They are stored in non-volatile memory.

Can I Access a Step in the Middle of a Sequence?

No. You must execute all steps preceding the step you want. You can only access by sequence name, not step number.

Why Do I Only Get Part of the Response to a SET? Query?

The 2432 always sends the entire response back to your controller. Since the SET? query response is very long (approximately 2500 bytes) it is possible that your controller's message timeout value is set too short. This can also cause a problem when using DEBUG mode in either SLOW or PAUSE. The controller will timeout and stop listening, but all you will see is an incomplete string.

How Can I Tell When a Command Has Been Executed?

You can insert a "dummy" query at the end of the command. The 2432 will process the message in the order it was received so the last thing the 2432 will do is respond to the query. After sending the commands put some type of "input" statement in your program to wait for the 2432 to respond to the query. When you get that response you know the command is done. This is the way IEEE suggests in the new release of GPIB standards. For example, if you send the command string 'CH1 VOLTS:5;CURSOR TPOS:ONE:50;EVENT?' to the 2432, you know that the 2432 has executed the commands when the response to the EVENT query is returned. You also know that the command string executed correctly if the event number returned is 0.

H

Appendix H

2432 Specific Information

Talked With Nothing to Say

The 2432 always says something when it is made a talker. If it has nothing to say, it sends a byte of all ones (ASCII RUBOUT) with the EOI signal. This lets the listening device know that no meaningful data is forthcoming, and prevents tying up the GPIB while one device waits for another to talk.

This can occur if the 2432 is sent an incorrect query and then talk addressed to read the response. Since the 2432 didn't understand the query it doesn't know what to say when talked and will respond with the RUBOUT character.

Another way for this to occur is to ask the 2432 for a response to a query that requires the 2432 to return a very long string (SET? or CURVE?) which may vary in total length. If your controller cannot handle long strings very well, the incoming string may be broken (while coming in) into a series of smaller strings. If the 2432 is finished sending the response string but the controller still needs to fill some of its substrings, then the 2432 will send the RUBOUT character for the remaining substrings.

Accepting Numbers

The 2432 always sends numbers in the number format specified in the command tables for that particular command. But it will accept any number format the controller sends. If the 2432 receives a number whose precision is greater than the instrument can handle internally, the number will be rounded to enhance accuracy.

Accepting Characters

The 2432 receives characters in both uppercase and lowercase and then maps them all into an uppercase set before evaluating them (for example: a = A, b = B). When using DEBUG mode to observe characters being sent to the 2432, only the uppercase characters will be shown since that is what the 2432 is working with. An exception is when using the MESSAGE command. The 2432 then interprets uppercase and lowercase letters differently when a backslash precedes the character.

Symbols can be abbreviated to the shortest number of characters that guarantee a unique symbol. The minimum string that needs to be sent is shown in bold uppercase letters in the command tables of Appendix A. Instruments with a different set of symbols than the 2432 will have a different way of abbreviating some of those symbols to make them unique. This can also be true of various firmware releases of the 2432. If a new command is added which introduces a symbol conflict, the minimum abbreviation of a current symbol will change. To avoid compatibility problems, use full symbols when programming the 2432. The time saved using the minimum abbreviation is minor because the length of time it takes to send the extra characters for full symbols is fairly short in relationship to the time it takes the 2432 to interpret and act on the command.

I/O Buffer Structure

The 2432 has a 64 character input buffer. Bytes sent from a controller are placed in the input buffer until it fills up. When the input buffer is full, the 2432 will not release the Ready For Data handshake line until the command interpreter has taken a character out of the buffer to process. This allows the 2432 to "keep up" with very fast controllers that are sending large blocks of data. This whole structure is transparent to the controller unless the controller has a very short timeout interval and the string being sent overflows the input buffer.

Power-Up Sequence of the 2432

When the 2432 is first powered up it goes through the following checks before it can be used.

BASIC HARDWARE CHECKS. The ROMs are checksummed and the computed checksums are compared with the checksum stored in each ROM. A series of test patterns are written into RAM to check for stuck bits. Non-volatile memory is checksummed and the results compared against the stored checksum. This last check is the one that determines if oscilloscope calibration has been compromised or if the oscilloscope has "forgotten" a waveform. If all of these checks show no problems then the RUNNING SELF TEST message appears on the display and the 2432 goes on to the next step.

ACQUISITION HARDWARE CHECKS. At this stage the CCDs, the preamplifiers, and the triggers are tested. The tests are the same ones done for the 7000, 8000, and 9000 series of tests found in the self diagnostics section of the operators manual. If any problems are encountered the extended diagnostic menu will be displayed and the next step (the SRQ) will not be done.

POWER-UP SRQ. The last thing done before turning control over to the user is to send a service request (SRQ). This SRQ indicates to the controller that the 2432 has just powered up. This SRQ is sent only if RQS is ON. See the "Service Request" portion of Appendix A for more information.

Appendix I

How to Use Fast Transmit Mode

Fast Transmit (FASTXMIT) mode allows the highest waveform transfer rate available from the 2432. The power of Fast Transmit becomes evident in a repetitive situation, where multiple waveforms are acquired and sent to the controller. Here, the 2432 can acquire a waveform, send it to the controller and rearm itself for the next acquisition at a very high rate. In any measurement situation where the trigger rate is less than the transfer rate (for instance video waveforms) the Fast Transmit mode allows full coverage with an acquisition for every trigger.

How to Set Up Fast Transmit

Let's discuss how to set up the 2432 for Fast Transmit Mode. First, you must select the waveform of interest. The FASTXMIT NORMAL command points at either CH1 or CH2, with the argument specifying the signal (or signals) to send. If you are using the DELTA TIME mode of the 2432 then use the the FASTXMIT DELTA command with the same arguments to send the delta waveforms.

Next specify the number of waveforms you wish to send. This number may vary from 1 to 65535 and specifies how many Acquire-Send sequences will be performed before the Fast Transmit completes.

Next select the type of waveform encoding to be used for the Fast Transmit waveform. Fast Transmit data can be sent in either the Entire Waveform Binary Format or the Partial Waveform Binary Format. For further information on these formats and other questions regarding the 2432's definitions of waveforms, see Appendix B. If PATH is ON then the waveform block will be preceded by a 'CURVE '. If PATH is OFF then just the waveform block of the selected format is sent. The terminator will always be a line feed with EOI.

The next thing to do is tell the 2432 to begin sending Fast Transmit waveforms. Do this by addressing the 2432 to talk. The transition from untalked to addressed-to-talk starts the activity. The Fast Transmit sequence will only begin when the controller issues the talk address for the 2432 and the user has requested some Fast Transmit waveforms. On most controllers, some sort of high level INPUT command causes this talking and untalking of devices automatically.

After reading your waveforms into the controller, make sure to turn Fast Transmit mode off by issuing the command 'FASTXMIT OFF' and then waiting for about 50 milliseconds to make sure the command was executed before issuing any more queries.

Here is a sample 4041 Basic program that will read 30 complete CH1 waveforms in Signed Integer format from a 2432. If the 2432 is in ACQUIRE mode when this program is run then each waveform will be newly acquired before being sent. If the 2432 is in SAVE mode when this program is run then each waveform will be a copy of the one before it because no new acquisitions will take place. In this example the Direct Memory Access (Option 01) feature of the 4041 is being used to increase the speed. Also, the 2432's address needs to be set to 1. The WAIT statement is necessary to allow the 2432 to interpret the command turning Fast Transmit off before continuing on with the program. See "Potential Traps" for what might happen if this wait is omitted.

```
10 Scope=1
20 Open #scope:" gpib1(pri=1,eom=<0>,tra=DMA,tim=2):"
30 Dim wave$ to 32000
40 Print #scope:" FASTXMIT 30,NORMAL:CH1,ENCDG:RIBINARY"
50 Input #scope:wave$
60 Print #scope:" FASTXMIT OFF"
70 Wait .05
80 End
```

Potential Traps

First, a word of caution. The Fast Transmit mode of the 2432 is optimized for speed. To accomplish this, Device Clear and Interface Clear are ignored during a Fast Transmit. The only way to abort a transmission and regain control over the 2432 is to turn the oscilloscope off and then turn it back on. Now, lets look at the two most common problems which occur when using Fast Transmit; either the controller program hangs, or the 2432 hangs.

Controller Program Hangs

The controller program will hang if the 2432 does not transmit a waveform within the timeout period specified in the controller. There are a couple of reasons why this might happen:

1. Fast Transmit mode will not work if the display is being updated in ROLL mode. There is never a true waveform "acquisition".
2. You have requested DELTA waveforms but the Delta Time mode of the 2432 is OFF and the 2432 is in ACQUIRE mode. In this case the requested waveforms will not be acquired so the data won't be sent.
3. You have asked for X number of waveforms from the 2432 but the 2432 has only been triggered X−1 times.

2432 Hangs

This is most often shown by having the 2432's screen completely blank and indicates that the 2432 has been talked but the controller program has not listened to the complete Fast Transmit waveform. The 2432 is now waiting trying to send the rest of the waveform. Here are some reasons that this might happen:

1. You didn't turn the Fast Transmit mode to OFF when it was no longer needed.
2. You omitted the 50 ms delay from the controller program after sending the 'FASTXMIT OFF' command to the 2432. This is necessary to allow the 2432 to interpret the command. This delay is necessary when sending any command to the 2432 while Fast Transmit is active. This includes changing any Fast Transmit parameters (i.e. 'FASTXMIT NORMAL:CH2') as well as changing any oscilloscope setup (i.e. 'CH1 VOLTS:2').

In either of the above two cases, when the controller sends a normal query to the 2432 and talks it to get the response, the 2432 will try to send another waveform because the Fast Transmit mode is still active. Since the controller's INPUT statement isn't expecting such a large response it stops listening before the 2432 is done talking.

3. You asked for X number of waveforms from the 2432 but only read X−1 of them in.

Fast Transmit Transfer Rates

Now for some Fast Transmit timing numbers. We'll give two sets of figures. The first set of figures uses the 4041 sample program shown above. The second set uses a very fast controller so that the only speed limiting factor is the 2432.

These timing measurements were made by attaching a frequency counter to the RTRIG BNC output on the rear panel. This measures the time between record triggers, including both the time to acquire the waveform along with the time it takes to send that waveform to the controller. All the waveform displays are turned off to speed up the transfer rate.

Because waveforms acquired at sweep speeds faster than 100 μs are handled differently than those acquired at 100 μs or slower there are two sets of numbers. At sweep speeds slower that 100 μs the limiting factor is not the data transfer rate but how long it takes to sample the waveform.

By using the Partial Waveform Binary Format to extract just a piece of the waveform you can speed up the transfer rate considerably. The 'ENCDG' portion of the 'FASTXMIT' command is used to select the type of format that Fast Transmit will use. See Appendix B for more information on waveform formats.

Transfer Rates Using the sample program:

		full waveforms/sec
100 μs/div	normal	43
	envelope	35
	average	21
50 μs/div to 500 ns/div	normal	37
	envelope	31
	average	20

Transfer Rates Using a very fast controller:

		full waveforms/sec
100 μs/div	normal	47
	envelope	38
	average	22
50 μs/div to 500 ns/div	normal	39
	envelope	33
	average	20

Some Points to Remember

1. Sending waveforms is an Acquire-Send sequence. The 2432 acquires a new waveform and then sends it.
2. In SAVE mode, sending multiple waveforms just sends multiple copies of the same waveform; no acquisition takes place.
3. Turning the display off increases the transfer rate.
4. The ONLY WAY to abort a transmission and regain control of the 2432 is to turn the power off and then back on.
5. Fast Transmit does not work in ROLL mode.
6. When using Fast Transmit and AVERAGE mode, every intermediate average is sent. To obtain a waveform that has been averaged 8 times, you need to throw the first 7 away and use the 8th waveform sent.

J

Appendix J

Example Programs

This appendix contains sample programs to use as guidelines in programming the 2432.

Talker-Listener Program for HP Basic

This HP Basic program allows you to send commands or queries to the 2432 and prints responses from the 2432.

```
10 !========================
20 ! first enter the 2432's address and then open the
30 ! hardware port to talk to the GPIB. Finally enable
40 ! the srq interrupt and point to the handler routine.
50 !========================
60  print " Enter 2432 address:"
70  input addr
80  if addr>30 then goto 60
90  if addr<0 then goto 60
100 assign @scope to 700+addr
110 dim answr$ [7000]
120 dim comm$ [200]
130 on intr 7 gosub 320
140 enable intr 7;2
150 !========================
160 ! Now get the command or query to send. A query
170 ! has a " ?"  right after the header. If it is a
180 ! query then talk the 2432 and print the response.
190 !========================
200 wait .25
205 comm$=""
210 input " Command? " ,comm$
215 if len(comm$)=0 then goto 210
220 output @scope;comm$,end
230 if  pos(comm$," ?" )=0   then goto 200
240 enter @scope;answr$
250 print " 2432  response is: " &answr$
260 goto 200
270 !========================
280 ! This is the code that is executed when an srq
290 ! happens. We poll the bus and print the status
300 ! byte and the corresponding event query response.
310 !========================
320 stb = spoll(700+addr)
330 output @scope;" path off;event?" ,end
340 enter @scope;event
350 print " srq received: status = " ;stb,"  event = " ;event
```

```
360 output @scope;" path on" ,end
370 enable intr 7;2
380 return
390 end
```

Talker-Listener Program for IBM Basic

This program uses the National GPIB card and the drivers supplied with that card. Lines 1 through 99 of the program need to be supplied with the DECL.BAS file.

```
100 '==========================
110 ' First find the 2432. If no 2432 out there then
120 ' stop. Start by clearing the screen.
130 '==========================
140 dev$=" DEV1"
150 call ibfind(dev$,dev%)
160 cls
170 if dev%<0 then  print " 2432  not found"  : end
180 gosub 480
190 print " Enter  your  command or query (use E to exit):" ;
200 line input msg$
210 '==========================
220 ' keep inputing commands until an e or E. These are
230 ' printed to the 2432 and the response read back. If
240 ' the response is empty it is not printed.
250 '==========================
260 while msg$<>" e"   and msg$<>" E"
270  b$=space$(240)
280  call ibwrt(dev%,msg$)
290  call ibrd(dev%,b$)
300  b=ibcnt%
310  while ibcnt%=240
320   print b$;
330   call ibrd(dev%,b$)
340   b=b+ibcnt%
350  wend
360  b$=mid$(b$,1,ibcnt%)
370  print b$;
380  print
390  print " " ;b;" bytes  were read" ;
400  print
410  gosub 480
414  print " Command? " ;
416  line input msg$
420 wend
430 end
```

```
440 '==========================
450 ' This part reads the status byte and prints both it
460 ' and the event code.
470 '==========================
480  sta%=1
490  while sta%<>0
500   print " status byte :" ;
510   call ibrsp(dev%,sta%)
520   print sta%;
530   msg$=" event?"
540   c$=space$(75)
550   call ibwrt(dev%,msg$)
560   call ibrd(dev%,c$)
570   print " , " ;mid$(c$,1,ibcnt%)
580   print
590  wend
600  return
```

Pulse Counting Program

This program is written in 4041 basic and will count the number of pulses between time cursors. This program demonstrates the use of the waveform data commands and queries as well as using custom text written to the screen of the 2432.

```
10   !==========================
20   !  setup the logical unit and enable srq's
30   !==========================
40  Scope=1
50  Open #scope:" gpib0(pri=1,eom=<0>):"
60  On srq then call srqhand
70  Enable srq
80   !==========================
90   !  unmask the user request srq for monitoring the
100  ! menu keys and then print a prompt message on
110  ! the 2432 screen.
120  !==========================
130 Print #scope:" message clrstate;menuoff;rqs on;user on"
140 Print #scope:" message 10:"" move  the time cursors to bracket"""
150 Print #scope:" message 9:"" the  part of the waveform you want"""
160 Print #scope:" message 8:"" to  count the pulses in"""
170 Print #scope:" message 6:"" press  any menu button when ready"""
180 Print #scope:" cursor  function:time"
190  !==========================
200  ! Wait here until the user presses a menu key.
210  !  Menukey will be reset in the srq routine.
220  !==========================
230 Menukey=0
240 If menukey=0 then goto 240
```

```
250 !==========================
260 ! The area of interest is now bounded with the time
270 ! cursors. Set START and STOP to these values with
280 ! the SNAP command.
290 !==========================
300 Print #scope:" snap"
310 !==========================
320 ! A pulse will be defined as having a positive edge
330 ! that has at least 50 percent amplitude. Now find
340 ! the amplitude of the signal and the 50 percent point
350 !==========================
360 Print #scope:" data source:ch1;path off;maximum?"
370 Input #scope:maxval
380 Print #scope:" minimum?"
390 Input #scope:minval
400 Midval=minval+(maxval-minval)/2
400 !==========================
410 ! Set the crossing level to the calculated 50 percent
420 ! point. Set the hysteresis up to filter noisy signals
430 ! and set the search direction from left to right.
440 !==========================
450 Print #scope:" level " ;midval
460 Print #scope:" hysteresis 20; direction plus"
470 !==========================
480 ! Now count the actual pulses. This consists of
490 ! looking for positive crossings of the 50 percent
500 ! level. If one is found the window is adjusted
510 ! and the count is incremented.
520 !==========================
530 Pulses=0
540 Print #scope:" pcross?"
550 Input #scope:location
560 If location=0 then goto 600
570 Pulses=pulses+1
580 Print #scope:" start " ;location+1
590 Goto 540
600 Print " number of pulses found = " ;pulses
690 Stop
700 End
710 Sub srqhand
720 !==========================
730 ! This routine handles an SRQ interrupt. If the
740 ! event code is 450-454 then a menu button was
750 ! and menykey is set to 1.
760 !==========================
770 Poll stb,saddr
780 Print #scope:" event?"
790 Input #scope:event
800 If (event>449)and(event<455) then  menukey=1
810 Resume
820 End
```

Extended Accuracy Timing Measurement Program

This program (written in 4041 Basic) makes extended accuracy timing measurements. It uses the time cursors and the pcross function for the time readout along with horizontal expansion for increased accuracy in determining the location of measurement points. The basic algorithm sets the time cursors at the 50 percent points of the waveform and then expands the waveform 100 times and repositions the time cursors more accurately.

```
10  !======================
20  ! First set the output channel and initialize some
30  ! variables. Set up the time per div, START and
40  ! STOP and then start the scope acquiring.
50  ! Wait for a period of time for an acquistion to occur.
60  ! You might also use single sequence to take just one
70  ! record. Path is turned off so we can just deal with
80  ! numbers on query responses. All the time measurements
90  ! will be done from level 0 or center screen.
100 !======================
110 delete var all
120 Open #1:" gpib(pri=0):"
130 Dim units$ to 200
140 Dim instr$ to 200
150 Dim outstr$ to 200
160 Integer pcr,pcr2
170 Print #1:" horizontal  asecdiv:100e-6;run acquire"
180 For i=1 to 250
190 Next i
200 Print #1:" run  save;start 1;stop 1024;level 0;path off"
210  !======================
220  ! Now get the first pcross value. This will be the
230  ! edge we start measuring from. If your edge is
240  ! different substitute a different search algorithm
250  ! here. The start value is moved to be one point before
260  ! the crossing of interest. Then we go looking for the
270  ! next crossing point so we can set stop there.
280  !======================
290 Input #1 prompt " pcross?" :instr$
300 Pcr=val(instr$)-1
310 Outstr$=" start " &str$(val(instr$))&" ;pcross?"
320 Input #1 prompt outstr$:instr$
330 Pcr2=val(instr$)-1
340  !======================
350  ! Now set the time cursors to the points that we have
360  ! just found. These will be used to give us the actual
370  ! time values we want. Then expand the horizontal by
380  ! 100x. The cursors will stay at the points that we
390  ! placed them even after the expansion.
400  !======================
410 Print #1:" cursor  function:time,units:time:base"
420 Print #1:" cursor  tpos:one:" &str$(pcr)
```

```
430 Print #1:" cursor tpos:two:" &str$(pcr2)
440 Input #1 prompt " horizontal? asecdiv" :instr$
450 Horex=val(instr$)/100
460 Print #1:" horizontal asecdiv:" &str$(horex)
470  !==========================
480  ! Now that we've expanded everything, we need to go in
490  ! and find the new pcross value. Then we move the time
500  ! cursor to that point and then go looking for the
510  ! ending point and set the other time cursor to that.
520  ! The difference between the time cursors is the answer.
530  ! A slight problem comes in here in that horizontal
540  ! position is altered slightly in doing the expansions
550  ! so that points line up correctly. This needs to be
560  ! compensated for and that is what the subroutine is
570  ! doing.
580  !==========================
590 Print #1:" cursor select:one"
600 Gosub fdghpos
610 Print #1:" cursor tpos:one:" &str$(cursor)
620 Print #1:" cursor select:two"
630 Gosub fdghpos
640 Print #1:" cursor tpos:two:" &str$(cursor)
650 wait 0.5
660 Input #1 prompt " cursor? display:unit" :units$
670 Input #1 prompt " cursor? display:value" :instr$
680 Print " The time difference is " &instr$&units$
690 Stop
700  !==========================
710  ! This subroutine finds the first pcross and then
720  ! fiddles with horizontal position before setting
730  ! the cursor value to a " corrected" value which
740  ! is returned in the variable cursor.
750  !==========================
760  fdghpos: !
770 Input #1 prompt " start 1;pcross?" :instr$
780 Pcrex=val(instr$)-1
790 Input #1 prompt " horizontal? position" :instr$
800 Horpos=val(instr$)
810 Exhorpos=horpos*100
820 If exhorpos<256 then goto left256
830 If exhorpos<102044 then goto middle else goto rt256
840  Left256:  refpos=exhorpos
850     goto calcnum
860  Middle:  pospick=(exhorpos-256)/100
870     refpos=256+(pospick-int(pospick))*100
880     goto calcnum
890  rt256:  refpos=1023-(1023-horpos)*100
900  calcnum: !
910     cursor=horpos+(pcrex-refpos)/100
920     return
930     End
```

Index

MANUAL CHANGE INFORMATION

Tektronix
COMMITTED TO EXCELLENCE

Date: 10-28-87 Change Reference: C1/1087

Product: 2432 Pro. Ref Guide Manual Part No.: 070-6614-00

DESCRIPTION

Product Group 37

PAGE 2-4

Change the following section to read as shown.

Which terminator should you choose?

The 2432 always recognizes EOI as a message terminator, no matter what byte it is asserted with. You may also program the 2432 to recognize a line-feed as a terminator. This is useful when using a controller which sends a line-feed character as the last byte of a message instead of asserting EOI with the last byte of a message.

When the 2432 is set up to recognize the line-feed character as the terminator, it will assume that an incoming message is terminated if it detects either EOI or a line-feed character. On output, the 2432 will send a carriage return followed by a line-feed with EOI asserted as the last bytes of each message. The 2432 always asserts EOI along with the last byte of a message, no matter what terminator has been selected.

A potential problem exists if you select the line-feed terminator for use with either BINARY waveform transfers or LLSET. An ambiguity occurs because the line-feed character may be a valid number in BINARY waveform and LLSET strings. If the controller receives the line-feed character as a valid string number, it will prematurely assume an end of message. When using a controller that recognizes other message terminators in addition to EOI, it is best to use ASCII format for waveform transfers and the SET? query. (FORMAT should be off—see below).

A similar problem occurs when transferring Auto Step sequences with 2432 terminator character (TERM) set to LF/EOI. (See "Sending and Receiving Sequences".) For transfers from one 2432 to another 2432 or to a printer, line-feed characters are output for each line transferred. For printers, the line-feed formats the output so the printer output is readable. For transfers to a scope, however, the receiving scope terminates prematurely on the first line-feed it receives. You should set TERM to EOI (only) to receive the entire sequence.

For controller directed transfers, binary transfers of sequences can have the same ambiguity described for binary waveform transfers. Set controllers or 2432's receiving binary sequences to recognize only EOI as the terminator character. Do the same for ASCII transfers of sequences, if FORMAT is turned on. (SEE PRGm? query and FORMAT command in Appendix A.)

PAGE 3-22

Change the following section to read as shown.

SENDING SEQUENCES TO A PRINTER OR ANOTHER 2432. This is very similar to the "talk-only" mode of the 2432 except that a formatted ASCII sequence is sent instead of a waveform. The device to which this information is sent may be either a printer or another 2432.

To configure the "From" 2432 to send sequences, first push the OUTPUT front panel button followed by the SETUP menu button. Then push the MODE menu button followed by the T/ONLY menu button, and finally make your selection by pushing the SEND PRGM menu button.

If you are sending sequences to another 2432, first set the "To" 2432 to use only EOI as the terminator. To do this, push the OUTPUT front-panel button followed by the SETUP menu button. Then push TERM in the setup menu. Finally, push the menu button labeled EOI to make the terminator selection. Next, configure the "To" 2432 to listen always mode. Use the same method used for configuring the "From" 2432 (above), except select L/ONLY instead of T/ONLY after pushing MODE.

If you are sending sequences to a printer, first configure the printer to listen always mode (see "Making Printer or Plotter Copies" for help).

After setting up the receiving 2432 or printer, select the sequence you want to send. You select it for transfer the same way as you select for recall. First, push the PRGM front-panel button. Then, push the RECALL menu button and use the up and down arrow keys to underline the chosen sequence.

Product: 2432 Pro. Ref Guide Date: 10-28-87 Change Reference: C1/1087

PAGE A-22

Change the description for VALue? to read as shown.

Automatic Feature Commands (cont)

Header	Argument	Argument	Argument	Description
VALue?	<types>			QUERY ONLY. Returns the specified parameter or <type> on the selected waveform. The source containing the waveform to be measured is selected with the **DAT**a **SOU**rce or **DAT**a **DSOU**rce commands, with **DAT**a **BDSOU**rce used only when **VAL**ue? uses the **DELA**y argument (see below). A new parameter calculation is done each time the **VAL**ue? query is received. **DAT**a **SOU**rce is used to select the source for all measurements except for **DELA**y. Since **DELA**y requires two targets, **DAT**a **DSOU**rce is used to select the second source, the "Delay To" target, while **DAT**a **SOU**rce selects the first source, the "Delay From" target. The **DAT**a **SOU**rce and **DAT**a **DSOU**rce commands are in "Waveform Commands" in this appendix. <types> available are: **DIST**al, **PROX**imal, **MESI**al, **MINI**mum, **MAX**imum, **MID**, **TOP**, **BAS**e, **MEAN**, **PK2**pk, **OVE**rshoot, **UND**ershoot, **WID**th, **PERI**od, **FRE**quency, **DUT**y, **RIS**e, **FAL**l, **RMS**, **ARE**a, **DELA**y, and **DME**sial. If the scope returns a value of 99E99 then an error has occurred. Check the event code to determine which one (see **EVE**nt?).

Product: 2432 Pro. Ref Guide Date: 10-28-87 Change Reference: C1/1087

PAGE A-29

Change the descriptions for LLPrgm and PRGm? to read as shown.

Sequencer Commands (cont)

Header	Argument	Description
LLPrgm	"ascii string"	When used as a query the sequence named "ascii string" will be returned in a low level binary block form. When that binary block is returned to the scope it will reconstruct the named sequence. If the incoming sequence has the same name as one currently in memory then the incoming sequence is ignored. If no "ascii string" name is sent then all sequences will be returned. If the named sequence is not present an error SRQ will be returned if **EXR** is **ON**. Available on the 2432 only. Any 2432 or controller receiving binary block transfers should be set up not to recognize the line-feed as the terminator character. See Section 2 for information on the terminator character. *NOTE* *The **LLP**rgm binary block form will speed up front panel transfers in a production environment but an 'archive' taken using the **SET**? query is recommended for upward compatibility with future firmware releases. The binary images of the scope setup may change with future releases.*
PRGm?	"ascii string"	QUERY ONLY. Will return a list of high level commands to reconstruct the named sequence. If "ascii string" is not included, all sequences present will be sent. If **FORM**at is **ON** then formatting characters will be inserted into the text to make a very readable listing of the sequence (see **FORM**at command). The complete path (see **PAT**h) will always be printed but the longform state (see **LON**g) will be observed. Available on the 2432 only. If **FORM**at is **ON**, any 2432 or controller receiving ASCII transfers should be set up not to recognize the line-feed as the terminator character. See Section 2 for information on the terminator character.

PAGE A-35

Change first argument <types>, under DATa, to include BSOUrce and change description to read as shown.

Waveform Commands (cont)

Header	Argument	Argument	Description
	<types>	**CH1** **CH2** **ADD** **MUL**t **REF1** **REF2** **REF3** **REF4** **CH1D**el **CH2D**el **ADDD**el **MULTD**el	**DAT**a **SOU**rce specifies the source (CH 1, CH 2, etc.) of the waveform that a **CURV**e?, **WFM**pre?, **WAV**frm? or **VAL**ue? query will return information on. For **VAL**ue? queries, it specifies the source that parameter will be extracted from. (See **VAL**ue? under "Automatic Feature Commands" in this appendix.) If the source specified is an empty reference memory, an error will be returned when the information is requested if **EXR** is **ON**. **DAT**a **DSOU**rce is only used when a **VAL**ue? query is used to extract the **DELAy** parameter. Then, it specifies the source (CH 1, CH 2, etc.) containing the "DELAY To" target (see **VAL**ue query). If the secondary source is an empty reference memory, an error will be returned when the information is requested if **EXR** is **ON**. <types> available are: **SOU**rce **DSOU**rce