Programmer Manual

Tektronix

2440
Digital Oscilloscope

070-6601-00

Please check for change information at the rear of this manual.

First Printing: September 1987
Revised Printing: February 1991

Printed in the U.S.A.

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

WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of three (3) years from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

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

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

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

Contents

A GPIB Commands

B Discussion on Waveforms

C Event Tables

J Example Programs

Index

Change Information

List of Illustrations

List of Tables

Operators Safety Summary

The general safety information in this part of the summary is for both operating and servicing personnel. Specific warnings and cautions will be found throughout the manual where they apply and do not appear in this summary.

Terms in Manuals

CAUTION statements in manuals identify conditions or practices that could result in damage to the equipment or other property.

WARNING statements in manuals identify conditions or practices that could result in personal injury or loss of life.

Terms on Equipment

CAUTION on equipment means a personal injury hazard not immediately accessible as one reads the marking, or a hazard to property including the equipment itself.

DANGER on equipment means a personal injury hazard immediately accessible as one reads the marking.

Symbols in Manuals

⚠

This symbol indicates where applicable cautionary or other information is to be found.

Symbols on Equipment

⚡	⏚	⚠
DANGER *High Voltage*	*Protective* *ground (earth)* *terminal*	*ATTENTION* *Refer to* *manual*

Power Source

This product is intended to operate from a power source that will not apply more than 250 V rms between the supply conductors or between either supply conductor and ground. A protective ground connection, by way of the grounding conductor in the power cord, is essential for safe operation.

Grounding the Product

The 2440 is grounded through the grounding conductor in the power cord. To avoid electric shock, plug the power cord into a properly wired receptacle before making connections to the 2440 input or output terminals. A protective-ground connection, by way of the grounding conductor in the power cord, is essential for safe operation.

Danger Arising from Loss of Ground

Upon loss of the protective-ground connection, all accessible conductive parts (including knobs and controls that may appear to be insulating) can render an electrical shock.

Use the Proper Power Cord

Use only the power cord and connector specified for your product. Use only a power cord that is in good condition. For detailed information on power cords and connectors see Table 1-1 in the Operators Manual.

Use the Proper Fuse

To avoid fire hazard, use only a fuse of the correct type, voltage rating and current rating as specified in the parts list for your product.

Do Not Operate in Explosive Atmospheres

To avoid explosion, do not operate this product in an explosive atmosphere unless it has been specifically certified for such operation.

Do Not Remove Covers or Panels

To avoid personal injury, do not remove the product covers or panels. Do not operate the product without the covers and panels properly installed.

1
Introduction

Introduction

The Tektronix 2440 Digitial Oscilloscope is a portable, dual-channel instrument with a maximum digitizing rate of 500 megasamples per second. The programmable features of the 2440 let you perform a wide variety of automatic tests and measurements. All of the testing capabilities of the 2440 can be controlled remotely via the GPIB.

How to Use This Guide

This guide is designed to help test-system programmers interface the Tektronix 2440 Digitial Oscilloscope with GPIB systems. It also is the reference for interfacing the 2440, via the GPIB, for non-controller directed operation (i. e., for outputting waveforms to printers without controller supervision). For a more general overview of 2440 operation, please refer to the 2440 Operators manual. Here are a few of the major topics this manual discusses:

- **Establishing GPIB Communication**—Section 2 shows how to set up the 2440 for GPIB operation and how to communicate with a controller.
- **How to Write Programs for the 2440**—Section 3 explains basic command syntax, types of commands, and various programming features of the 2440.
- **Debugging Your Programs**—Section 4 gives hints for fixing your 2440 programs if they have bugs and explains how to use the 2440 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 2440.
- **Waveform Transfers**—Appendix B explains how to transfer waveform data to or from the 2440's internal storage areas and what that data actually means.
- **Events Tables**—Appendix C lists the event codes you can get from the 2440. Event codes are numbers that correspond to status, error, and system messages. The 2440 outputs the code and you look up the message in this appendix.
- **GPIB Introduction**—Appendix D gives a brief overview of basic GPIB theory.

- **Other Appendices**—These provide various information useful in 2440 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 in 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 2440 and debugging the programs you've written. Also, use the HELP feature of the 2440 to learn about any areas that you might have questions about. HELP is described in the Section 5 of the Operators Manual.

2 Establishing GPIB Communication

Establishing GPIB Communication

The GPIB is the communications link between the 2440 and your controller. The first thing to do is make sure a functional GPIB cable is connected from the 2440 to the controller. Then set the 2440 GPIB parameters so they configure the 2440 for your GPIB system.

The GPIB parameters include the mode, terminator, and address. All are accessed from the front panel of the 2440. This section discusses how to set up these various GPIB parameters.

Setting the GPIB Mode

The 2440 can be set up to be a talker, a listener, or both. It can also be taken “off bus”. In addition, the 2440 can output waveform data to printers and plotters. Any of these different functions are selected by changing the GPIB mode of the 2440.

In general, you will set the mode to be talker/listener. This setting allows the 2440 to accept commands from a controller over the GPIB and to send answers (in response to queries) back to the controller. The 2440'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, then pressing 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 now look at each of these options.

Talk-Only

Pushing the T/ONLY selection puts the 2440 into the Talk-Only GPIB mode. In this mode, the 2440 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 2440) connected to one or more Listen-Only devices (such as a printer, another oscilloscope, or another 2440).

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 2440 will send when it talks. All talk-only options require at least one listener on the bus. If there is no listener present on the bus at the time the instrument is ready to transmit, the 2440 will display an error message on its CRT and refuse to transmit. The three options available in this menu are:

CURVE—With this selection, the 2440 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 pressing the TRANSMIT menu button. The 2440 will then send the CURVE portion of all displayed waveforms to all listeners on the bus.

WAVFRM—With this selection, the 2440 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 pressing the TRANSMIT menu button. The 2440 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 2440 will transmit an AutoStep Sequencer program. The 2440 will send whichever Sequencer program is currently selected (underlined) in the RECALL menu for the Sequencer (press the **PRGM** front-panel button followed by the RECALL menu button). To transmit the Sequencer program, simply press the SEND PRGM button (from the T/ONLY submenu), followed by pressing the **OUTPUT** front-panel button, followed by pressing the SENDPRGM menu button.

Listen-Only

Pushing the L/ONLY menu selection button places the 2440 into the Listen-Only mode. The front-panel ADDR indicator should light. When in Listen-Only mode, the 2440 acts only as a listener (LADS). It can only accept commands over the GPIB; it cannot send any messages or assert SRQ.

While in this mode, the controller may send setup data (previously queried from the 2440 or constructed independently) to the 2440 to implement a standard test setup. This mode can also be used in transfers of waveform data between two 2440's where a controller is not involved. In this application, the 2440 that is in listen only mode will be the one to which data is sent.

Talk-Listen

Pushing the T/L menu selection button places the 2440 into the Talk-Listen mode. In this mode, the 2440 acts as both a Talker and a Listener. This is the normal GPIB mode which most people will use for day-to-day testing activities. In this mode, the 2440 can accept data and commands and reply to queries.

Devices

Pushing the **DEVICES** button brings up a menu which allows you to tell the 2440 what type of hard copy device you are using. The 2440 can format waveform information for several types of output devices. The devices supported are: some devices that accept HPGL (Hewlett-Packard Graphics Language), such as the HP7470A Plotter, HP7475A Plotter, HC100 (TEK) Plotter, and the HP Thinkjet® printer (model HP-2225A). (HP, HPGL, and Thinkjet are registered trademarks of Hewlett-Packard). You press the menu button HPGL PLOTTER when selecting one of the plotters; press THINKJET PRINTER for selecting the HP Thinkjet® printer.

To make a printout, first connect the device to the 2440 and make sure the device is in Listen Always mode. Make sure all other GPIB instruments are disconnected since we will be operating without a controller. To start the printout, just press the 2440 **OUTPUT** front-panel button, followed by the PRINT menu button. The menu button will be labeled PLOT if HPGL PLOTTER is chosen in the DEVICE menu.

While the 2440 is printing, the PRINT button becomes an ABORT button. Press this button at any time to terminate the printout. To disable the printing mode of the 2440, you must return to the MODE menu and select another mode besides DEVICES. The section Making Printer or Plotter Copies on page 3-26 talks about setting up for different plotters and printers.

Off Bus

Pushing the OFF BUS selection makes the 2440 bus-transparent. When the 2440 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 pressing 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 2440 always recognizes EOI as a message terminator, no matter what byte it is asserted with. You may also program the 2440 to recognize a line-feed as a terminator. Recognizing the line-feed 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 2440 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 2440 will send a carriage-return, followed by a line-feed with EOI asserted, as the last bytes of each message. The 2440 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 is a valid number in binary waveform and LLSET strings. If the controller receives the line-feed character as such a number, it will prematurely assume 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 the SET? query.

A similar problem occurs when transferring AutoStep sequences. Sending and receiving sequences is discussed on page 3-24. Line-feed characters are output for each line when transferring from one 2440 to another 2440 or to a printer. For printers, the line-feed formats the output so it is readable. For transfers to a scope, however, the receiving scope terminates prematurely on the first line-feed it receives if its terminator character is set to LF/EOI. You must set TERM to EOI 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 2440'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.

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 by one. The selected GPIB address establishes both primary talk and listen addresses for the 2440. It can be set to any value from 0 to 30, inclusive.

Example Controller Program

This short program allows you to send commands or queries to the 2440 and prints responses from the 2440. The program is written in BASIC using the National Drivers and illustrates basic principles that should carry over to other languages. Lines 100-230 are required for the National GPIB drivers.

```
100 CLS110 REM decl.bas
120 REM
130 REM GURU initialization code; declarations
140 REM
150 CLEAR ,58900!           'IBM BASICA Declarations;
                             =BYTES FREE -size(bib.m);
160 IBINIT1 = 58900!        'a smaller-than-calculated # is OK
170 IBINIT2 = IBINIT1 + 3   'these lines (thru CALL statements
below)
180 BLOAD "bib.m",IBINIT1   'MUST be included in your program
190 CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IB-
SIC,IBLOC,IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IB-
IST,IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
200 CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWR-
TA,IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IB-
DIAG,IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IBERR%,IBCNT%)
210 DEV$ = "DEV1"
220 CALL IBFIND (DEV$,DEV%)
230 IF DEV% < 0 THEN PRINT "IBFIND ERROR":END
300 PRINT "ENTER YOUR COMMAND OR QUERY (USE E TO EXIT):";
310 LINE INPUT MSG$
320 REM *****************************************************
330 REM   Keep inputting commands until e or E is entered.
340 REM   These are printed to the 2440 and the response read
350 REM   back. If the response is empty it is not printed.
360 REM *****************************************************
370 WHILE MSG$<>"e" AND MSG$<>"E"
380 RD$=SPACE$(240)
390 CALL IBWRT(DEV%,MSG$)
400 CALL IBRD(DEV%,RD$)
410 B=IBCNT%
420 WHILE IBCNT%=240
430 PRINT RD$;
440 CALL IBRD(DEV%,RD$)
450 B=B+IBCNT%
460 WEND
470 RD$=MID$(RD$,1,IBCNT%)
480 PRINT RD$;
490 PRINT
500 PRINT B;"BYTES READ";
510 PRINT
```

```
520 GOSUB 600
530 PRINT " COMMAND? ";
540 LINE INPUT MSG$
550 WEND
560 END
600 REM ******************************************************
610 REM  This part reads the status byte and prints both it
620 REM  and the event code.
630 REM ******************************************************
640 STA%=1
650 WHILE STA%<>0
660 PRINT " STATUS BYTE :";
670 CALL IBRSP(DEV%,STA%)
680 PRINT STA%;
690 MSG$= "PATH ON;EVENT?"
700 C$= SPACE$(75)
710 CALL IBWRT(DEV%,MSG$)
720 CALL IBRD(DEV%,C$)
730 PRINT ", ";MID$(C$,1,IBCNT%)
740 PRINT
750 WEND
760 RETURN
```

Debugging the Interface when It Doesn't Work

This subsection helps you debug the interface when you can't talk to the 2440 with your controller. First, make sure the GPIB cable is securely connected to both the 2440 and the controller. If there is any doubt about the cable, get another one to make sure it is not causing the problem.

Next, go look at the GPIB status menu. To get there simply press 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 2440 is set up. Make sure the mode is set to T/L (both talk and listen) and the terminator and address are set up correctly.

Also, make sure that the Debug mode is not paused. When Debug is paused, the 2440 will not listen to any bytes from the bus, so this could be the problem. 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 not the problem. If Debug is paused, turn it off by pressing the **OUTPUT** front-panel button, then the DEBUG menu button to display the DEBUG menu. Then turn Debug off by pressing the DEBUG ON|OFF menu button.

GPIB Status Menu

Now a word about the GPIB status menu. This menu is very useful to refer to when you have questions about how the GPIB is set up. The underlined listings are GPIB-related 2440 functions and the adjacent listing gives the status for the function.

Most of the listings are self-explanatory (all are covered in this guide), but there are a few functions on this menu screen that need explanation. This menu is the only place where you can find out what the Fast Transmit status is. This is because as soon as the Fast Transmit mode is turned on, the 2440 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 2440 will interpret binary data sent to it. The DATA ENCDG status (just above BINWFM) shows the format of binary data the 2440 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 2440'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 SRQ in the listing indicates that 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 2440, the event code pointed to will be returned.

What Do the GPIB Status LED's Mean?

The 3 lights over the 2440 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 2440 will not respond to any front-panel controls. The 2440 locks out its front panel while doing a self-calibration, an Auto Setup, and during certain portions of AutoStep 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 2440. Or it can send the LOCK LLO command to the 2440, and follow that command with the GPIB universal command LLO. The LLO universal command is an IEEE-488-1978 command while LOCK ON is a 2440-specific command. The LOCK command is explained further on page A-40.

SRQ—This LED is lit while the 2440 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 2440 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

This section describes how to send commands to and get responses from the 2440. A few guidelines can help make your programming task easier:

1. If possible, use full command and argument names. This makes them more readable in program listings and makes upgrading easier. Also, new releases of firmware may have slightly different abbreviations which are usually longer since there are more commands.

2. Try to write instrument-specific software in a modular fashion using subroutines when possible. This technique makes it easier to debug and modify your program when something changes.

Sending a Command to the 2440

To change the 2440 operating state, you must send it a command. For example, to change the Volt/Division setting for Channel 1 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 2440 when receiving a message. In this section, you will learn what a command is made up 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 2440 what to do. Here are three examples of 2440 commands for you to refer to as we discuss headers and arguments.

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 2440. For instance, by using the CURSOR header (see example 3), the 2440 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 2440 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, sending the header RUN cannot tell the 2440 whether to actively acquire or go to save mode. Therefore the argument SAVE is added to the header to indicate that the 2440 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 2440 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 and are called NR1, NR2, and NR3 respectively. They will be denoted as <NR1>, <NR2>, and <NR3> in this manual. 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 shown earlier into one long message by inserting a semicolon between them. 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 ties up 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 linefeed character as a terminator. You need to determine which type of controller you are using and make sure that the 2440's terminator is set appropriately.

If you select EOI as terminator, the 2440 will interpret any data byte received with EOI asserted as the end of the input message. The 2440 will assert EOI simultaneously with the last data byte of an output message.

If you select the LF character as terminator, the 2440 will interpret the LF character without EOI asserted (or any data byte received with EOI asserted) as the end of an input message. The 2440 will transmit a carriage return followed by a line-feed (LF) with EOI asserted to terminate output messages. See page 2-4 for further information on selecting message terminators.

Asking the 2440 a Question

There are many times when a controller needs to obtain information about the operating state of the 2440. To do this, the controller sends a query or question to the 2440. Queries consist of a header followed by a question mark. For example, to find out whether the 2440 is currently acquiring or in save mode, simply send the query RUN? to the 2440. If the response message is RUN ACQUIRE, then the 2440 is acquiring. If the response is RUN SAVE, then the 2440 is in save mode.

You may query the 2440 for any header 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 returns 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 the 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 arrive at the number you're after, you must strip off the path portion. You can eliminate this extra processing by setting PATH to OFF before sending the query to the 2440. When PATH is OFF, the 2440 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 2440 to obtain a setup which you wish to send back to the 2440 (to recreate that setup), be sure to set PATH to ON. Without the path portion, the 2440 will be unable to interpret the command. In the above example, returning the string CH1 POSITION:2 to the 2440 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 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 2440 will send the full representation for each header and argument contained in a query response. If LONG is OFF, only the essential characters are returned, resulting in a shorter response string. LONG does not affect commands being sent to the 2440. The addition of new features to the 2440 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 2440 firmware.

How to Use Service Requests (SRQs)

The 2440 can issue a Service Request (SRQ) to interrupt the controller and let it know that something interesting has happened. To issue an SRQ the 2440 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 2440 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 2440 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 2440 encountered a command error while interpreting the last command string sent by the controller. An EVENT? query sent to the 2440 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, indicating that one of the symbols (headers and/or arguments) making up the command was mis-typed and the 2440 could not make sense of it.

When the 2440 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 2440 only saves 2 events which are capable of asserting SRQ at any one time. When these 2 slots are full, up to 8 subsequent events are placed into an event buffer. If the event buffer becomes full, 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 2440 is currently asserting SRQ on the bus and will insist that the controller poll the 2440 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 2440 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 2440.

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 in the program example on page 2-6.

Next you need to determine what kind of information you will need for your application. The 2440 allows the programmer to select the types of occurrences that will assert an SRQ. For example, sending the command CER ON to the 2440 will result in command errors asserting SRQs. If you wish to keep the 2440 from asserting any SRQs, send the command RQS OFF. For a complete list of commands of this type, refer to the Service Request Commands on page A-39.

One common use for SRQs is to have the 2440 assert SRQ when it has finished a task. For example, you might have the 2440 assert SRQ when it completes a single-sequence operation. See the Operation Complete Event Codes in Appendix C for a complete list of tasks that the 2440 can report with an SRQ. Let's walk through some of the steps a controller program would take in order to wait until the 2440 has finished an acquisition.

1. Turn this type of SRQ on by sending the OPC ON command to the 2440.
2. Clear out all previous events and SRQs by sending the command INIT SRQ.
3. Set the 2440 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 2440 to start acquiring by sending the command RUN ACQUIRE.

At this point the controller will wait for the SRQ from the 2440. 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 2440 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 2440. Actually, programming without SRQs is simpler to understand and to implement, but it does not allow the controller to perform other tasks while the 2440 is completing its task. Before you begin programming without SRQs, you must first send the RQS OFF command to make sure the 2440 will not generate any SRQs.

When SRQs are not used, the status of the 2440 is determined by sending the EVENT? query. First clear out all the current events and SRQs that might be present by sending an INIT SRQ command to the 2440. Then have the controller set the 2440 to the desired operational state and wait for the 2440 to finish. During this waiting period, the controller repeatedly sends an EVENT? query to the 2440 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 2440 has nothing to report, it will return EVENT 0 in response to the EVENT? query. An EVENT query must be used because the 2440 will not update the status byte if RQS is OFF. The 2440 does not know when a status byte is read by a controller unless SRQ is asserted with the status byte. The 2440 never changes the current status byte because it does not know when the controller is ready for a new status byte. This situation is similar to the one discussed earlier when event number 459 is returned.

There are other ways to determine that the 2440 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 2440 is acquiring, the response will be RUN ACQUIRE. When the single sequence operation is complete, the 2440 enters Save mode and the response will change to RUN SAVE.

Making Measurements with the 2440

The 2440 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 2440 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 2440 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 2440 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. Commands are listed starting on page A-5.

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 on page 3-5 and 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 parameter selected 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 2440:

```
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 the point 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 determination 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, the 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 Appendix C in the Operators 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 are initially set to 10, 50, and 90 percent, respectively and can be adjusted. Each level can be individually set to a percentage of the difference between the base and top levels, or each can be set to an absolute voltage level. For example, to set the mesial level to 45% up from the base level, send the following command to the 2440:

```
MEASUREMENT MESIAL:UNITS:PERCENT,MESIAL:PLEVEL:45
```

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 on page A-5 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 2440:

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

MEASUREMENT DISPLAY—Up to four parameters may be selected for display on-screen. Visual voltage threshold crossing indicators or 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 2440:

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

ERROR AND WARNING CONDITIONS—When using WPE, the 2440 defines an error condition as one that prevents it from getting the information it needs to calculate a requested parameter value. It defines a warning condition as one that makes accuracy of the value calculated for the parameter questionable. In both cases, the commands you sent to setup the 2440 for the measurement were not adequate.

Let's take errors first. Assume that you have input an approximately 10 kHz waveform into CH 1 of the oscilloscope, and you want to measure its frequency. You send the command ASECDIV 500E-9, which sets the oscilloscope to a 500 ns/div acquisition rate. Then, you send the command DATASOURCE:CH1:VALUE?PERIOD to return the period for the waveform. An error condition is created, since a 10 μs acquisition record (500 ns/div × 20 div) is not long enough to let the 2440 acquire the entire period of a 10 kHz waveform (100 μs).

When an error occurs, the 2440 always returns 99e99, and sends an Execution Error event code. The error message that corresponds to this event code can be found in the Execution Error Events table in Appendix C. In this example the event code returned would be 264.

Now let's consider warnings. Assume you put an approximately 10 V pk-pk square wave into CH 1, and you want to measure its pk-pk voltage. You send the command CH1 VOLT:5E-1 which sets CH1 to 0.5 V/D. Then, you send DATA SOURCE:CH1:VALUE?PK2PK to return the amplitude in V pk-pk for the waveform. A warning condition is created, since the vertical window for acquisitions is 10.24 divisions. At 0.5 V/D, a 10 V pk-pk waveform requires 20 divisions for a vertical window.

NOTE

The actual vertical window size is 10.24 divisions for acquisition rates of 100 μs/Div and slower. At acquisition rates of 50 μs/Div and faster, the window decreases to a minimum of 9 divisions at some acquisition rate settings. See page 3-16.

When a warning condition is created, the 2440 will return the value of the parameter, and will send an Execution Warning event code. The warning message that corresponds to this event code can be found in the Execution Warning Events table in Appendix C.

In general, to determine if the parameter value returned is valid you can use EVENT? after each VALUE? query to get the warning message.

If you're programming with SRQ's, you can get the error message SRQ's by enabling ERW mask (default is ON) and the warning message SRQ's by enabling EXW (default is ON). Then you use EVENT? as before to get the error code, and you look it up in Appendix C of this guide.

NOTE

As you've seen from this discussion on error and warning conditions, there are considerations for setting up the 2440 for performing WPE's. We suggest that you consult Section 3, Section 5, and Appendix C in the 2440 Operators Manual for more information.

EXAMPLE MEASUREMENT PROGRAM—The following BASIC program using the National GPIB Drivers will automatically set the 2440 to acquire and display an unknown signal, optimize it for a rise time measurement, and return the rise time value. A EVENT query loop is used to wait for the acquisition to complete before the measurement is made. Lines 100 - 230 are required for the National GPIB Drivers.

```
100 CLS
110 REM decl.bas
120 REM
130 REM GURU initialization code; declarations
140 REM
150 CLEAR ,58900!           'IBM BASICA Declarations;
                            =BYTES FREE -size(bib.m);
160 IBINIT1 = 58900!        'a smaller-than-calculated #
                            is OK
170 IBINIT2 = IBINIT1 + 3   'these lines (thru CALL
                            statements below)
180 BLOAD "bib.m",IBINIT1   'MUST be included in your
                            program
190 CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IBSIC,IBLOC,
    IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IBIST,
    IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
200 CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWRTA,
    IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,
    IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IBERR%,
    IBCNT%)
210 DEV$ = "DEV1"
220 CALL IBFIND (DEV$,DEV%)
230 IF DEV% < 0 THEN PRINT "IBFIND ERROR":END
300 EV$=SPACE$(50)
310 RD$=SPACE$(50)
320 CMD$="RQS OFF;INIT SRQ;PATH OFF"
330 CALL IBWRT(DEV%,CMD$)
340 REM Set to fastest time/div for risetime measurement.
350 CMD$="ACQ REP:ON;HORIZ ASEC:2E-9"
360 CALL IBWRT(DEV%,CMD$)
370 CMD$="MEASUREMENT METHOD:HISTOGRAM;DATA SOURCE:CH1"
380 CALL IBWRT(DEV%,CMD$)
390 CMD$="ATRIGGER MODE:SGL;RUN ACQUIRE"
400 PRINT "ACQUIRING DATA FOR RISE TIME MEASUREMENT"
410 CALL IBWRT(DEV%,CMD$)
420 CMD$="EVENT?"
430 CALL IBWRT(DEV%,CMD$)
440 CALL IBRD(DEV%,EV$)
450 WHILE VAL(EV$) <> 461   'Waiting for single sequence
                            to complete.
460 CALL IBWRT(DEV%,CMD$)
470 CALL IBRD(DEV%,EV$)
480 WEND
490 CMD$="VALUE? RISE"
500 CALL IBWRT(DEV%,CMD$)
510 CALL IBRD(DEV%,RD$)
520 PRINT "RISE="RD$
530 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. Such extractions are useful when the automatic measurements just described 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 2440.

The segment of the targeted waveform that is analyzed when making measurements is designated by the START and STOP commands. The results obtained are similar those obtained by using 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, send the command VMINIMUM? to the 2440. 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 across 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 2440 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 2440 will proceed 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 the crossing level to -15 and the hysteresis to 10, the 2440 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 2440 will return a value of 0 in response to PCROSS?.

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

The Vertical Window for Making Measurements

Although the 2440's screen displays 8 vertical divisions, WPE and custom measurements can be made on waveforms fitting within at the least a 9-division vertical window at every acquisition rate setting and up to a 10.24-division vertical window at some acquisition rate settings. The maximum vertical window available for any acquisition rate (Sec/Div setting) is specified as Dynamic Range in Table 6-1 of Section 6 of the Operators manual. For convenience, Table 3-1 gives the window size in vertical divisions and range in both signed and positive-integer representation. For maximum resolution of amplitude measurements or time-related measurements using time-reference points tied to amplitude, choose a Volts/Div setting that sizes the waveform to occupy as much of that vertical window as possible.

Table 3-1: Vertical Window Size

	Range		
Acquisition Rate (Sec/Div)	**Signed Integer**	**Positive Integer**	**Size Divisions**
5 sec to 100 μs	-128 to +127	0 to 255	10.24
50 μs to 500 ns	-124 to +123	+4 to 251	9.92
200 ns	-121 to +120	+7 to 248	9.76
100 ns	-113 to +112	+15 to 240	9.04
50 ns to 2 ns			
REPET Off	-113 to +112	+15 to 240	9.04
REPET On	-121 to +120	+7 to 248	9.68

At acquisition rates of 100 μs/Div and slower, the vertical window is 10.24 divisions. However, at settings of 50 μs/Div and higher, the vertical window decreases. An example will illustrate why it is important to keep in mind the variability of vertical window size: a 10 V signal displayed at a Volts/Div setting of 1 V and an acquisition rate of less than 50 μs/Div would fit within the 10.24-division vertical window available at that acquisition rate setting. However, if the acquisition rate is increased to 100 ns/Div, the vertical window decreases to about 9 divisions; therefore, a setting of 1 Volt/Div would cause the signal to be clipped. The Volts/Div setting for this acquisition rate should be 2V.

To avoid clipping waveforms, either determine the vertical window for the acquisition rate used and select the Volts/Div setting accordingly, or, if high-resolution measurement isn't needed, select Volts/Div setting based on the the worst case vertical window. The 9-division window just calculated is the smallest window available for any acquisition rate settings; therefore, selecting the Volts/Div setting to keep the waveform within this 9-division window ensures the waveform won't clip for all acquisition rate settings.

RESOLUTION—Although the window size varies with different acquisition rates, the per division vertical resolution for each Volts/Div setting is constant at 1/25 of 1 division, since there are always 25 values or digitizing levels (DL's) per division. Therefore, given that the waveform to be measured is sized to fit within the vertical window, you can determine the resolution you will obtain for a given measurement. Let's use the previously-discussed 10 volt waveform as an example: for the 10.24-division window, at the 1 V/DIV setting, the resolution is 1/25 of 1V and 1/25 of 2V for the 9-division window (2 V/DIV setting). This gives 40 mv or 0.4% resolution for the 10.24-division window versus 80 mv or 0.8% resolution for the 9-division window. In general, vertical resolution for a measurement is expressed as follows:

R(volts) = 1/25 × V/DIV setting,

R(%) = 100/(25 × Vsig),

where

R = resolution

Vsig = signal amplitude in divisions

Also, see the YMUlt argument to the WFMpre query on page A-57.

Regardless of the window size, if increased resolution is needed, AVG mode can be used when measuring repetitive signals. By averaging a waveform over a user-specified number of averages, the obtainable resolution is improved by 1 divided by the user-specified number of averages. In other words, for 2 averages specified, the resolution is 2 times better or 1/50 of a division; for 4, 1/100 of a division, etc. The accuracy of the measurement is not improved, only the resolution.

How to Use the 2440 AutoStep Sequencer

The 2440 AutoStep Sequencer (hereafter referred to as sequencer) 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 2440, sent back to the controller, or sent to another 2440. 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 2440 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 2440 should be set to the desired operating mode including all measurements, Auto Setup 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 2440 is functioning optimally. The actions Auto Setup, 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 desired actions, add up their assigned numbers and send the sum to the 2440 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, the command to send is SETUP ACTION:160.

Here are definitions 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 2440 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 are performed on the 2440 prior to loading the programmed front-panel settings. If self-diagnostics fail, the sequence is aborted and the extended diagnostics menu is displayed.

Auto Setup (8) An Auto Setup will be performed immediately after setting up the saved front panel. The type of Auto Setup will depend upon the mode selected in the saved front panel. Auto Setup makes many changes to the 2440 front-panel setup loaded before the Auto Setup is performed. See Table B-16 in Appendix B of the 2440 Operators manual to see which controls are affected and how they are affected.

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) When Bell is on for a step, the internal 2440 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, 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 codes.

Pause (128) After the completion of all other actions, the 2440 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 SEQUENCE IN connector. 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 for the first step of a sequence, the sequence is protected from accidental deletion. Setting protect on for other steps has no effect. To delete a protected sequence, the first step must be edited to turn protect off for that step, or you can override the protection by sending a SETUP FORCE:ON GPIB command.

The 2440 automatically controls the order of each selected action within any sequence step. For instance, Self-Cal and Self-Test actions are always completed before loading the instrument setup. The Auto Setup 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 2440 is in the desired operating mode, you may save the sequence step by sending the SETUP SAVE:"seq name" command. This command 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 2440. 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, the 2440 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", and "FIVE" are reserved for saving only single-step sequences with no actions. By reserving these names, compatibility is maintained with previous 2440 instruments. Saving a step while specifying a reserved name replaces the previous step stored under the reserved name rather than appending the new step to the old (remember, reserved names are for one front-panel setup only). These names may be used without quotes just as on previous 2440 instruments. For example, SETUP SAVE:"ONE" is equivalent to SETUP SAVE:ONE.

The selection menu limits the number of sequences to 40 because only 40 names can be displayed on the 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 200 and as few as 25 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 2440, 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 2440 will find the desired sequence, load the first step, and begin executing the actions associated with the first step. See What Are Actions on page 3-18 for more information on what each action does. If the 2440 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 unprotected sequences. An individual sequence can be deleted by sending the SETUP DELETE:"seq name" command to the 2440. This will remove the sequence called "seq name". All sequences currently saved in the 2440 may be deleted by sending the SETUP CLEAR command. Remember, deleted sequences are gone forever.

To delete protected sequences, you must override the protection before using the SETUP DELETE or SETUP CLEAR command. You can override the protection by sending the SETUP FORCE:ON command to the 2440. As long as this override is in effect, any sequence can be removed by the SETUP DELETE or SETUP CLEAR commands. 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.

Using the Sequencer

The sequencer in the 2440 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 2440 becomes much less significant.

CONTROLLING THE SEQUENCER—Several methods of maintaining synchronization between the 2440's sequencer and the controller are provided. In addition to letting the controller command the sequencer, these methods let the controller know what state the sequencer is at any time so that the total test can be managed.

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 finished, the technician 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. If paused, 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?—With the sequencer, the 2440 can provide semi-automatic, stand-alone testing. For many test applications, test setups can be entered from a controller or from the front panel as a sequence and the controller removed from the test system. The 2440 can then be used as a stand-alone portable tester for field or in-house work. Testing speed is increased, since you no longer need to change 2440 settings—you just select the next step!

In applications for which you still need the controller, the sequencer still enhances the 2440's usefulness in test-systems. By programming the sequencer with your test-setup sequence, you enter the front-panel setups once, as a sequence, and then run the sequence as often as desired. For repetitive testing tasks, running a test sequence (as opposed to sending a series of test setups again and again) increases test-system throughput by reducing bus traffic between the controller and the 2440. Once you create and send a test sequence to the 2440 sequencer, you change to a new test setup by merely proceeding to the next sequence step (just sending the SAVE RECALL command to the 2440 is enough to invoke a completely new setup).

The sequencer also helps in controller-related applications by managing each acquisition cycle. The sequencer monitors the acquisition cycle and proceeds to the next action only when the oscilloscope is ready. Bus traffic is further reduced, since the controller is relieved of continuously monitoring the 2440 for these actions.

Sending and Receiving Sequences

The 2440 can send stored sequences to a controller, to a printer, or to another 2440. The controller can request the 2440 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 2440, but the sequence must be sent in ASCII. This feature is useful for making hard copies of a sequence or updating another 2440 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 2440, send the LLPRGM? "seq name" query to the 2440. This query 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 2440 will be returned.

To send a sequence or sequences to the 2440, send the data block returned by the LLPRGM query back to the 2440. 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 2440 memory.

The binary format used by the 2440 in compressing its sequences depends on the firmware version of the instrument from which the sequence originally came. 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 and archive an ASCII version of the sequence 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 will be readable by, and are upwardly compatible with, future instruments in the 2400 series. 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 2440. The 2440 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 about the same length response as is returned by a SET? query. To send this sequence back to the 2440, 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 2440. 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 2440—This operation is very similar to that used to make copies of 2440 acquired waveforms without using a controller (see page 3-26). Both operations use the talk-only mode of the 2440 to allow output without the controller. Here, however, instead of getting hardcopy of waveforms, a formatted ASCII sequence is sent. The device to which this information is sent may be either a printer or another 2440.

To configure the 2440 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 2440, first set the "To" 2440 to use only EOI as the terminator. 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" 2440 to listen always mode. Use the same method used for configuring the "FROM" 2440 to T/ONLY, 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 below for help.

After setting up the receiving 2440 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.

Now make sure there is a GPIB cable between the "from" 2440 and the "to" device (printer or 2440). To begin the transfer, push the **OUTPUT** front-panel button and then the SENDPRGM menu button.

Making Printer or Plotter Copies

The 2440 can print hard copies of its waveforms on a Hewlett-Packard Thinkjet® printer and some HPGL plotters such as the Tektronix HC100 and the Hewlett-Packard HP7470A and HP7475A Plotters. 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 2440 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.

When the 2440 acquires waveforms, 20 divisions (1024 data points) are acquired but only the 10 divisions (512 data points) seen on screen are printed. All 20 divisions can be printed using HC100 mode, and is discussed on page 3-32.

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 in 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 2440 to make the copy, how to connect the two together, and how to start the copy process.

SETTING UP THE PRINTER/PLOTTER—Here, the idea is to make the hard copy device a listener. This task is normally the controller's responsibility, but, since we are operating without a controller, it must be done by the user. Different devices may have different methods for 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 five 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 2440—There are two steps involved in setting the 2440 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. Next, push the MODE menu button, followed by the DEVICES menu button. Pushing the DEVICES menu button should turn on the ADDR indicator LED. This lit ADDR indicator shows that the 2440 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 2440 AND THE PRINTER—The 2440 and the printer/plotter should be the only instruments connected to the GPIB when a controller is not being used. Connect a GPIB cable from the 2440 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 right-most menu button. This button will be labeled PRINT, if you selected the Thinkjet® printer, or PLOT, if you selected the HPGL plotter mode. 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.

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.

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. The controller can then make the printer or plotter a listener so that it can receive and print the hard copy from the 2440. The address is assigned in the same way as when 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 2440—If you are already using a controller to talk to the 2440, the GPIB setup of the 2440 will not change for printing/plotting. If you need to start from scratch, see page 2-1 on how to set the 2440 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 2440 should send. The format is selected by sending DEVICE TYPE:HPGL or DEVICE TYPE:THINKJET to the 2440. The DEVICE command also allows you to compose the print in much the same way as the DEVICES menu button.

CONNECTING THE 2440 AND THE PRINTER—Make sure there is a GPIB path between the 2440 and the printer/plotter. If both instruments can communicate with the controller, this path exists.

STARTING THE COPY—There are three things the controller must do to start a hard copy output from the 2440. First, the PRINT command must be sent to the 2440. Second, the printer/plotter must be listen addressed. Third, the 2440 must be talk addressed. As the 2440 sends data, the printer/plotter should receive that data and begin printing/plotting.

Here is a sample program written using BASIC and the National GPIB Drivers. It is using board level commands to initiate the printer by sending the PRINT command, turning the Thinkjet® printer into a listener and the 2440 into a talker. Lines 380-390 tells the 2440 to send data for the graticule and waveform, but not to send data for plot text or settings. Lines 100 - 230 are required for the National GPIB Drivers.

```
100 CLS
110 REM decl.bas
120 REM
130 REM GURU initialization code; declarations
140 REM
150 CLEAR ,58900!           ´IBM BASICA Declarations;
                            =BYTES FREE -size(bib.m);
160 IBINIT1 = 58900!        ´a smaller-than-calculated #
                            is OK
170 IBINIT2 = IBINIT1 + 3   ´these lines (thru CALL
                            statements below)
180 BLOAD "bib.m",IBINIT1   ´MUST be included in your
                            program
190 CALL IBINIT1(IBFIND,IBTRG,IBCLR,IBPCT,IBSIC,IBLOC,
    IBPPC,IBBNA,IBONL,IBRSC,IBSRE,IBRSV,IBPAD,IBSAD,IBIST,
    IBDMA,IBEOS,IBTMO,IBEOT,IBRDF,IBWRTF,IBTRAP)
200 CALL IBINIT2(IBGTS,IBCAC,IBWAIT,IBPOKE,IBWRT,IBWRTA,
    IBCMD,IBCMDA,IBRD,IBRDA,IBSTOP,IBRPP,IBRSP,IBDIAG,
    IBXTRC,IBRDI,IBWRTI,IBRDIA,IBWRTIA,IBSTA%,IBERR%,
    IBCNT%)
210 DEV$ = "DEV1"
220 CALL IBFIND (DEV$,DEV%)
230 IF DEV% < 0 THEN PRINT "IBFIND ERROR":END
300 REM THIS PROGRAM ASSUMES SCOPE ADDRESS IS SET TO 1
    AND PRINTER ADDRESS 29
310 V% = 1320 CALL IBONL (DEV%,V%)´PLACE DEVICE ONLINE
330 CALL IBSIC (DEV%)       ´Send interface clear
340 CMD$="_?"               ´UNTALK UNLISTEN
350 CALL IBCMD(DEV%,CMD$)
360 CMD$="!^"               ´LISTEN SCOPE TALK PC
370 CALL IBCMD(DEV%,CMD$)
380 CMD1$="DEVICE TYPE:THINKJET,SET-
TINGS:ON,GRAT:ON,TEXT:OFF,WAVFRM:ON;PRINT"
390 CALL IBWRT(DEV%,CMD1$)  ´SEND COMMAND TO SCOPE
400 CMD$="=A"               ´LISTEN PRINTER TALK SCOPE
410 CALL IBCMD(DEV%,CMD$)
420 PRINT " PRINTING WAVEFORM FROM SCOPE IN PROGRESS"
430 X% = 0                  ´DELAY FOR COMMANDS
440 WHILE X% < 5000
450 X% = X% + 1
460 WEND
470 CALL IBONL (DEV%,V%)    ´PLACE PC OFFLINE
480 END
```

KNOWING WHEN A COPY IS FINISHED – There are three methods for determining whether the print or plot has completed. First, the 2440 can be programmed to signal the controller by issuing an Operation Complete SRQ, with an associated event code of 463, at the conclusion of the hard copy. Operation Complete, or OPC, SRQ's are enabled with the command OPC ON. Second, the controller can do a serial poll of the 2440 and check the BUSY bit in the status byte, since the 2440 sets this bit while it is doing the hard copy and clears it when it is done. (See Using SRQs and Events on page 3-5 for more information on the status byte and Operation Complete SRQs.) The third 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 2440.

Save-On-Delta Hard Copies

When used without a controller, the 2440 can be set up in a babysitting mode. In this mode, the Save-On-Delta is used in tandem with Envelope mode to create a limit envelope for monitoring an event. (See Storage Applications, Save-On-Delta mode, for information on creating an envelope from the GPIB. These titles are found in Section 3 of the 2440 Operators Manual.)

When events occur that move outside this limit window, the 2440 goes to save mode and sends a hard copy of the event to the printer. The out-of-limit event is hard copied, time-stamped, and printed. The 2440 then rearms the Save-On-Delta for further monitoring of events.

The Save-On-Delta feature highlights the out-of-limit event by centering it on the screen. Since PRINT outputs the 10 divisions on screen, you always get a hard copy with the event centered for easy identification.

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 2440.

NOTE

The positive integer format mode for the HC100 cannot be used with Save-On-Delta waveforms (either with or without controller direction). Use the HPGL mode for the HC100 plotter when using Save-On-Delta.

The special HC100 mode has two advantages. First, you get a hard copy of all 1024 points (20 divisions) of a waveform instead of just the 512 points (10 divisions) contained on the screen. Second, it can be faster because the plotter does its own formatting. The plotter doing the formatting reduces bus traffic considerably since fewer bytes are sent over the bus and the HC100's input buffer can hold most of the bytes. The 2440 is freed for other tasks.

To illustrate the time savings possible by use of the special HC100 mode, consider that it takes over 2 minutes to plot one waveform using the HC100 in HPGL mode. During this time, the 2440 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 2440 only sends data to the HC100 for about 35 seconds. Therefore, the 2440 is free for other tasks after only 35 seconds, instead of 2 minutes.

To use this special HC100 plotter mode, make sure the plotter is working correctly by following the procedures for setting up 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 up the 2440 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.

How to Save Instrument Settings

This section explains various methods for saving and restoring 2440 operational settings, or obtaining default settings. First we'll look at the two user-selectable front-panel setups the 2440 can automatically restore at power-up. Second, the two methods for saving and restoring settings when using a controller are discussed. Finally, for use without a controller, we look at how the 2440's internal AutoStep Sequencer can be used to save and restore front-panel setups.

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 telling the 2440 to send a data string representing its front-panel setup, and then storing that string in controller memory. The 2440 can send this string in either ASCII or binary format.

Because future versions of firmware for the 2440 may contain different feature sets, the abbreviations for some symbols may vary between versions. For this reason, we recommend that you store an ASCII format archive of each front-panel state sent to the 2440 during the course of your controller program. To do this, simply send LONG ON;SET? to the 2440, then query the 2440 and save its response.

ASCII FRONT PANEL SETUP—The SET? query is used to obtain the ASCII form of the 2440's front-panel state. The string returned by the 2440 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 2440. 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 2440 and restoring them to another 2440 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 message terminator.

BINARY FRONT PANEL SETUP—You use the LLSET? query to extract a binary version of the 2440's front panel. LLSET stands for Low-Level-SET. The response to an 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 2440. Because this block is directly translated into a 2440 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 2440 attaches an internal checksum byte to the block which is checked when the block is returned to the 2440. 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 2440.

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 2440 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 2440. 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 2440 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 or from the controller. From the controller, set up the 2440 with the settings you wish to save and then issue the command SETUP SAVE:"name" to the 2440. The 2440 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 2440's sequencer memory, simply send the ASCII string back to the 2440. It takes around 8 seconds to reload the sequence into the 2440 sequencer. Once the sequence is in the 2440'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 Sequencer Commands section of Appendix A.

Using Group Execute Trigger

When the 2440 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 2440 is that GET speeds the execution time of the selected task. This is because the 2440 does not have to read and interpret the command before performing 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 2440 to do upon receipt of the GET, then issue the appropriate DT command. For example, let's say you want the 2440 to set up for a Save-On-Delta operation and then start acquiring. First, send the command DT SODRUN to the 2440. Then, when you want the scope to enter the Save-On-Delta mode and begin acquiring, issue the GET command. Refer to the manuals for your controller to learn how to make your controller to send a GET. See page A-26 for more information on the DT command.

4
Debugging Your Programs

Debugging Your Programs

The 2440 has a DEBUG mode to help users make their GPIB programs run. You can use this feature to display all messages sent over the GPIB or only messages sent to the 2440.

Displayed messages are scrolled horizontally across the 2440 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

To become familiar with the various Debug modes, we encourage you to write a very short program which sends just one or two commands to the 2440 over and over again. This short program will give you something with which to experiment.

After writing the program, push the **OUTPUT** front-panel button, followed by the DEBUG menu button. You will then see the DEBUG menu displayed on the bottom 3 lines of the screen. Push the menu button labeled DEBUG ON|OFF so that ON is underlined. DEBUG mode is now active.

Interacting with the Display

To monitor messages sent to the 2440, push the menu button labeled BUS|SCOPE 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 located.

When an error occurs, the 2440 will automatically pause the message to allow you to see it. 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. The PAUSE button label is underlined when the display is paused.

You can observe both incoming and outgoing messages. The type of message is selected by pushing the menu button labeled IN|OUT (this button toggles between the IN and OUT settings). While IN is underlined, only messages being sent to the 2440 are displayed. While OUT is underlined, both incoming and outgoing messages will be displayed. Outgoing messages (query responses) appear underlined as they scroll across the screen.

The characters appearing on screen are the 2440'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 menu button labeled SLOW. While SLOW is underlined, the update rate is reduced to allow you to read the text. Push the PAUSE button to freeze the message on screen. You may then take as long 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 2440 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 2440 as a Bus Monitor

You can use the 2440 to monitor all GPIB traffic by pushing the menu button labeled BUS|SCOPE so that BUS is underlined. The 2440 must be in Listen Only mode before BUS can be selected. If you get an error message while attempting to make the BUS selection, go to the section on Setting the GPIB Mode on page 2-1 and change the mode to Listen Only.

In BUS monitor mode, the 2440 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 2440!

To use the 2440 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.

A
GPIB Commands

GPIB Commands

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 (don't send the brackets as part of the argument). In any command that has a default, omitting the link argument sets the default. For example, RUN ACQUIRE and RUN are equivalent.

Table A-1: Acquisition Commands

Header	Argument	Argument	Description
ACQuire	ERAse		COMMAND ONLY. Causes the acquisition sequence currently running to restart. This is similar to a single-sequence reset.
	MODe	**AVG** **ENV** **NOR**mal	Selects one of the three acquisition modes listed. **AVG** selects average mode, **ENV** selects envelope mode, and **NOR**mal selects normal mode. If set to **AVG** with delay time and **DELT**a set to **ON**, only Delay 1 will be displayed, and a warning SRQ will be issued if **EXW** is **ON**.
	NUMACq		QUERY ONLY. Returns the number of acquisitions that took place before save was activated. Turning **SAVD**el to **ON** will reset the **NUMAC**q number to zero. An intermediate number will be returned if save has not been entered yet.
	NUMAVg	<NR1>	<NR1> is the number of acquisitions averaged using the Stable Average mode (see Section 5, Storage System, AVG (Average) in the 2440 Operators Manual for more information). Valid <NR1> values are: 2, 4, 8, 16, 32, 64, 128, or 256. Any other value will be set to the closest valid value and an SRQ will be issued if **EXW** is **ON**.

EXAMPLES

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

Table A-1: Acquisition Commands (Cont.)

Header	Argument	Argument	Description
ACQuire (cont)	**NUME**nv	<NR1> **CON**t	<NR1> is the number of envelope sweeps done before resetting. Valid <NR1> values are: 2, 4, 8, 16, 32, 64, 128, 256, or **CON**t. When **NUME**nv is set to **CON**t acquisition restarts when a control setting that affects the data being acquired is changed or the **ACQUIRE** button is pressed. Any other value will be set to the closest valid value and an SRQ will be issued if **EXW** is **ON**. The number of envelopes is ignored in Roll mode.
	REPet	[**ON**] **OFF**	When **ON**, enables equivalent time sampling at sweep speeds of 50 ns and faster. When **OFF** at these same speeds, points between samples will be interpolated. **REP**et mode has no effect at sweep speeds of 100 ns and slower.
	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**. Termination of **SAVD**el can be determined by checking for the following: **OPC** SRQ, **BUS**y? is **OFF**, or **ATR**igger? **STAT**e is **SAV**e.

EXAMPLES

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

Table A-1: Acquisition Commands (Cont.)

Header	Argument	Argument	Description
RUN	[ACQuire] **SAVe**		The **ACQuire** argument will cause the scope to start an acquisition. **SAVe** will cause an immediate transition to Save mode.
SMOoth	[**ON**] **OFF**		Turns smoothing **ON** or **OFF**. If smoothing is **ON** a 5 point auto regressive moving average is performed on any displayed waveforms except references. Waveforms sent using a CURVe? are affected by smoothing.

EXAMPLES

Query	Response
RUN?	RUN ACQUIRE
SMOOTH?	SMOOTH OFF

Table A-2: Automatic Feature Commands

Header	Argument	Argument	Argument	Description
AUTOSetup	**EXE**cute			COMMAND ONLY. Causes one Auto Setup cycle to take place. In **AUTOS**etup the scope automatically sets itself up to get a good "picture" of an incoming waveform. The scope will not act on any new commands until this cycle is complete.
	MODe	**FALl** **PERI**od **PUL**se **RIS**e **VIE**w		The **MOD**e parameter tells the scope what type of a display or measurement the user desires. **FALl** focuses on the falling slope of the waveform, **PERI**od displays one cycle, **PUL**se focuses on the minimum pulse width, **RIS**e focuses on the rising slope, and **VIE**w sets up the display for best overall viewing.

EXAMPLES

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

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
AUTOSetup (cont)	RESolution	HI LO		When **RES**olution is set to **HI**, the Auto Setup 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 Auto Setup cycle will optimize the horizontal resolution to fit the waveform on the screen. **RES**olution is ignored in **VIE**w mode.

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
MEASurement	**ONE**	**DSO**urce **SOU**rce **TYP**e	<data src> <data src> <type>	Specifies the measurement type and data source of the value in the first measurement display line. Four measurement types may be displayed on the screen at one time. **DSO**urce specifies the delay source, **SOU**rce specifies the data source, and **TYP**e specifies the type of measurement. <type> can be one of: **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. <data src> can be one of: **CH1**, **CH2**, **ADD**, **MUL**t, **REF1**, **REF2**, **REF3**, **REF4**, **CH1D**el, **CH2D**el, **ADDD**el, and **MULTD**el.

EXAMPLES

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

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
MEASure-ment (cont)	**TWO**	**DSO**urce **SOU**rce **TYP**e	<data src> <data src> <type>	Specifies the measurement type and data source of the value in the second measurement display line. **DSO**urce specifies the delay source, **SOU**rce specifies the data source, and **TYP**e specifies the type of measurement. Refer to **MEAS**urement **ONE** for valid <type> and <data src> values.
	THRee	**DSO**urce **SOU**rce **TYP**e	<data src> <data src> <type>	Specifies the measurement type and data source of the value in the third measurement display line. **DSO**urce specifies the delay source, **SOU**rce specifies the data source, and **TYP**e specifies the type of measurement. Refer to **MEAS**urement **ONE** for valid <type> and <data src> values.

EXAMPLES

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

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
MEASurement (cont)	**FOU**r	**DSO**urce **SOU**rce **TYP**e	<data src> <data src> <type>	Specifies the measurement type and data source of the value in the fourth measurement display line. **DSO**urce specifies the delay source, **SOU**rce specifies the data source, and **TYP**e specifies the type of measurement. Refer to **MEAS**urement **ONE** for valid <type> and <data src> values.
	DISPlay	[**ON**] **OFF**		Controls the display of measurements on the screen.
	DISTal	**PLE**vel **UNI**ts **VLE**vel	<NR3> **PERC**ent **VOL**ts <NR3>	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%.

EXAMPLES

Query	**Response**
MEASUREMENT? DISPLAY	MEASUREMENT DISPLAY:OFF
MEASUREMENT? DISTAL	MEASUREMENT? DISTAL:UNITS: PERCENT,DISTAL:PLEVEL:9.00E+1, DISTAL:VLEVEL:2.400

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
MEASurement (cont)	**DME**sial	**PLE**vel **UNI**ts **VLE**vel	< NR3 > **PERC**ent **VOL**ts < NR3 >	Changes the mesial value used in calculating where on the delay waveform to measure to. **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.
	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.
	MESIal	**PLE**vel **UNI**ts **VLE**vel	< NR3 > **PERC**ent **VOL**ts < NR3 >	Changes the mesial value used in calculating measurement results. The mesial value is mid way between top and base. **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. A typical mesial value is 50%.

EXAMPLES

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

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
MEASure-ment (cont)	**METh**od	**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.
	PROXimal	**PLE**vel	<NR3>	Changes the proximal value used in calculating measurement results. The proximal value is in closest proximity to the base. **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. A typical proximal value is 10%.
		UNIts	**PERC**ent **VOL**ts	
		VLEvel	<NR3>	
	WINdow	[**ON**] **OFF**		Turns windowing on or off. Windowing limits measurement calculations to within the time cursor area if Time Cursors are on.

EXAMPLES

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

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
UNIts?	<type>			QUERY ONLY. Returns the units associated with the specified measurement <type> on the selected waveform. The selected waveform is specified with the **DAT**a **SOU**rce or **DAT**a **DSOU**rce command. A new parameter calculation is done each time the **UNI**ts? query is received. See the **VAL**ue command for valid <type> values.
VALue?	<type>			QUERY ONLY. Returns the measurement value for the specified <type> on the selected waveform. The selected waveform is specified with the **DAT**a **SOU**rce or **DAT**a **DSOU**rce command, with **DAT**a **DSOU**rce used only when **VAL**ue? uses the **DELA**y argument. A new parameter calculation is done each time the **VAL**ue? query is received.

EXAMPLES

Query	Response
VALUE? RISE	VALUE RISE:2.000E-7

Table A-2: Automatic Feature Commands (Cont.)

Header	Argument	Argument	Argument	Description
VALue? (cont)				**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, or the "Delay To" target, while **DAT**a **SOU**rce selects the first source, or the "Delay From" target. The **DAT**a **SOU**rce and **DAT**a **DSOU**rce commands are on page A-54. <type> can be one of: **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, **FALl**, **RMS**, **ARE**a, **DELA**y, and **DME**sial. If no type is specified, all types are returned. If the scope returns a value of 99E99, then an error has occurred. Check the event code to determine which one (see **EVE**nt?).

Table A-3: Calibration and Diagnostic Commands

NOTE

Refer to the Operators manual Appendix A for additional information on the calibration and diagnostic functions.

Header	Argument	Description
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.
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 commands to the scope.
HALt		COMMAND ONLY. Stops any test being executed; specifically used to stop a looping test. Will also stop any sequence that is currently running.
LOOp	**CON**t **FAI**l **ONE** **PAS**s	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.
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.

Table A-3: Calibration and Diagnostic Commands (Cont.)

Header	Argument	Description
TESTNum	<NR1>	<NR1> indicates the test number within the **TESTT**ype that is to be run when **TESTT**ype is set to **EXTCA**l or **EXTD**iag. 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. Refer to Appendix A Table A-1 in the Operators Manual for test numbers and hierarchies.
TESTType	**SELFC**al **SELFD**iag **EXTCA**l **EXTD**iag	Indicates which type of test will run upon receipt of an **EXE**cute command. **SELFC**al automatically calibrates the scope's analog system, **SELFD**iag tests the internal system microprocessor components, **EXTD**iag allows you to run individual **SELFD**iag tests and uses internal feedback.

Table A-4: 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	**DISP**lay	**UNI**ts **VAL**ue		QUERY ONLY. **VAL**ue will return the <NR3> value 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**.
	FUNction	**OFF** **ONE/T**ime **SLO**pe **TIM**e **VO**Lts **V.T**		Selects the cursor type. **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**.

Table A-4: Cursor Commands (Cont.)

Header	Argument	Argument	Argument	Description
CURSor (cont)	**MOD**e	**ABSO**lute **DELT**a		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.
	NEWref			COMMAND ONLY. Use the current position of the cursors to set up the new reference for making percentage, dB, and degree measurements.
	REFSlope	**VAL**ue	<NR3>	Sets the reference slope level.
		XUNit	**CLK**s **DIV** **SEC** **V** **VV**	Sets the units in the x axis for the reference slope level. **CLK**s specifies the sample clock, **DIV** specifies graticule divisions, **SEC** specifies seconds, **V** specifies volts, and **VV** specifies volts squared.
		YUNit	**DIV** **V** **VV**	Sets the units in the y axis for the reference slope level. **DIV** specifies graticule divisions, **V** specifies volts, and **VV** specifies volts squared.

Table A-4: Cursor Commands (Cont.)

Header	Argument	Argument	Argument	Description
CURSor (cont)	**REFT**ime	**UNI**ts	CLKs **SEC**	Sets the units of the reference time level. **CLKs** specifies the sample clock, and **SEC** specifies seconds.
		VALue	<NR3>	Sets the reference time level.
	REFVolts	**UNI**ts	**DIV** **V** **VV**	Sets the units of the reference volts level. **DIV** specifies graticule divisions, **V** specifies volts, and **VV** specifies volts squared.
		VALue	<NR3>	Sets the reference volts level.
	SELect	**ONE** **TWO**		Selects which cursor is the active one.
	TARget	**ADD** **ADDD**el **CH1** **CH2** **CH1D**el **CH2D**el **MUL**t **MULTD**el **REF1** **REF2** **REF3** **REF4**		This command selects which waveform the user would like the cursors 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.

Table A-4: Cursor Commands (Cont.)

Header	Argument	Argument	Argument	Description
CURSor (cont)	**TPO**s	**ONE** **TWO**	<NR3> <NR3>	<NR3> sets the vertical time cursor position to the waveform location selected. The range for <NR3> is 0 to 1023 with a resolution of 0.01 (see **HOR**izontal **POS**ition on page A-27). <NR3> values outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON**.
	UNIts	**SLO**pe	**BAS**e **DB** **PERC**ent	Selects the units for the slope cursors. **BAS**e units are volts/ second.
		TIMe	**BAS**e **DEG**rees **PERC**ent	Selects the units for the time cursors. **BAS**e units are seconds.
		VOLts	**BAS**e **DB** **PERC**ent	Selects the units for the volts cursors. **BAS**e units are volts.

EXAMPLES

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

Table A-4: Cursor Commands (Cont.)

Header	Argument	Argument	Argument	Description
	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 outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON**.
	YPOs	**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 outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON**.

Table A-5: Display Commands

Header	Argument	Argument	Description
INTENSIty	**DISP**lay	<NR3>	**DISP**lay sets the intensity of the display. The range for <NR3> is a number from 0 to 100 with a resolution of 0.25; 0 being off and 100 being the brightest. <NR3> values outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON.**
	GRAt	<NR3>	**GRA**t sets the intensity of the graticules. The range for <NR3> is a number from 0 to 100 with a resolution of 0.25. <NR3> values outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON.**
	INTENS	<NR3>	**INTENS** sets the intensity of the intensified zone in A intensified Horizontal Mode. The range for <NR3> is a number from 0 to 100 with a resolution of 0.25. <NR3> values outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON.**
	REAdout	<NR3>	**REA**dout sets the intensity of the readout. The range for <NR3> is a number from 0 to 100 with a resolution of 0.25. <NR3> values outside this range will be set to the closest valid number and a warning SRQ will be sent if **EXW** is **ON.**

EXAMPLES

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

Table A-5: Display Commands (Cont.)

Header	Argument	Argument	Description
INTENSIty (cont)	**VEC**tors	[**ON**] **OFF**	Determines whether the display is drawn using vectors (**ON**) or dots (**OFF**).
MENuoff			COMMAND ONLY. Clears any menus occupying the three bottom display lines that label the menu buttons. Any full-screen menus/displays, such as the SNAPSHOT display, Sequencer sub-menus, status (instrument, OUTPUT, or TRIG) are also cleared. Useful for creating custom menus: **MEN**uoff, followed by a **USE**r **ON** command, must be used to clear the menu and enable SRQ's to be issued when menu buttons are pressed. The programmer then sends his own custom menu to the 2440. (See **MESS**age command for creating the menu.)
MESSage	<NR1>	"string"	Displays a message, specified by "string", on the screen at line <NR1>. The range of <NR1> is 1 to 16 with line 1 at the bottom of the screen and line 16 at the top. <NR1> values outside this range will be set to the closest valid number and a SRQ will be sent if **EXW** is **ON**. The string will be left justified in the display and will be blank filled or truncated to 40 characters long.

Table A-5: Display Commands (Cont.)

Header	Argument	Argument	Description
	CLRstate		COMMAND ONLY. Clears out all 16 lines of readout from the instrument display. *NOTE* *The top two and bottom three lines of the 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 these lines (see MENuoff 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.*
REAdout	[**ON**] **OFF**		Turns all readout display **ON** or **OFF**.

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"

Table A-6: Horizontal Commands

Header	Argument	Argument	Description
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. <NR1> values outside this range will be set to the closest valid number and if **EXW** is **ON** a warning SRQ will be sent.
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.

EXAMPLES

Query	Response
DLYEVTS?	DLYEVTS MODE:OFF,VALUE:124
DLYTIME?	DLYTIME DELTA:OFF,DLY1:1.8400E-5,DLY2:0

Table A-6: Horizontal Commands (Cont.)

Header	Argument	Argument	Description
HORizontal	**ASE**cdiv	<NR3>	<NR3> can range from 2E-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 and if **EXW** is **ON**, an SRQ will be issued. If the A sweep is set faster than the B sweep, B sweep is set or locked to match A sweep and an SRQ is sent if **EXW** is **ON**. Changing the A sweep setting causes the B sweep setting to change until the sweeps are unlocked (see **BSE**cdiv).
	BSEcdiv	<NR3>	<NR3> range is the same as for **ASE**cdiv. The B Sec/Div setting can be set whether **BSW**eep mode is being used or not. B sweep can be set to any available Sec/Div setting faster than the A Sec/Div setting (this is how you unlock the A and B sweeps). As long as changes to the A and B Sec/div settings keep A slower than B, the sweeps are independently setable or unlocked. If A sweep is set faster the B sweep the sweeps again lock and a SRQ is issued if **EXW** is **ON**. Subsequent changes to the A Sec/Div sets both sweeps to the same Sec/Div setting until the sweeps are unlocked as just described.

EXAMPLE

Query	Response
HORIZONTAL?	HORIZONTAL MODE:ASWEEP, POSITION:5.12000E+2,ASECDIV:1E-5, BSECDIV:1E-5,EXTEXP:1

Table A-6: Horizontal Commands (Cont.)

Header	Argument	Argument	Description
HORizontal (cont)	**EXTE**xp	<NR1>	Sets the expansion factor that is applied to an externally clocked A waveform in Save mode. Valid <NR1> values are: 1, 2, 5, 10, 20, 50, or 100. The expansion factor is reset to 1 if not in Save mode or not working with externally clocked waveforms. If <NR1> does not match a valid setting, it will be set to the closest valid setting and if **EXW** is **ON**, an SRQ will be issued.
	MODe	**AIN**tb **ASW**eep **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.

Table A-6: Horizontal Commands (Cont.)

Header	Argument	Argument	Description
HORizontal (cont)	**POS**ition	< NR3 >	**POS**ition will move point < NR3 > of the waveform(s) to the center of the display. The range for < NR3 > is 0 to 1023 with a resolution of up to 0.01. The resolution for unexpanded live waveforms is 1; the 0.01 resolution is to accommodate expanded saved displays. If the display is expanded 2X, points are named 0.0, 0.5, 1.0,...1022.5 (expressed in scientific notation) and the resolution is 0.5. If the display is expanded 10X, points are named 0.0, 0.1,...1022.9, 1023.0, and so on up to 0.01 resolution at 100X expansion. If < NR3 > requires rounding, such as when the resolution specified is greater than available for the display, or limiting, such as when < NR3 > value is outside the 0-1023 range, a warning SRQ will be issued if **EXW** is **ON**.

EXAMPLE

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

Table A-7: Miscellaneous Commands

Header	Argument	Description
BELl		COMMAND ONLY. Rings the bell.
DEBug	[**ON**] **OFF**	Command controls the debug option. See the description of Debug mode for further information.
DT	**OFF**	The **DT** command tells the scope what to do with any Group Execute Trigger (GET) command it receives over the GPIB. The **DT OFF** command causes the scope to ignore any GET command sent to it. **DT OFF** is the power-up state.
	RUN	Does the same as a **RUN ACQ**uire command when a GET is received.
	SODRUN	Executes the following commands when a GET is received: "**ACQ**uire **SAVD**el:**ON**;**RUN ACQ**uire".
	STEp	When a GET command is received, the Sequencer will go on to the next step if it is currently pausing.
	"ascii string"	When a GET command is received, the scope will load the named sequence into its Sequencer memory and start the execution of the sequence.
HELp?		QUERY ONLY. Will return a list of all valid command headers available to the user.

EXAMPLE

Query	**Response**
DT?	DT SODRUN
HELP?	HELP AUTOSETUP,ACQUIRE,RUN,CH1,CH2, VMODE,EXTGAIN,BTRIGGER,HORIZONTAL, ATRIGGER,DLYTIME,CURSOR,SETWORD, 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 MESSAGE,PANS,PRINT,REM,SAVEREF,SNAP STEP,TESTNUM,TESTTYPE,TIME,WFMPRE

Table A-7: Miscellaneous Commands (Cont.)

Header	Argument	Description
ID?		QUERY ONLY. Returns the message "ID TEK/2440, 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>: "01-OCT-90 V2.40/2.5"
INIT	[**BOT**h]	Does both the **PAN**el and **GPI**b initializations.
	[**ERA**se]	Irretrievably erases any saved waveforms, whether on screen or stored in REF memory locations, as well as any sequences stored in sequencer memory. Also performs the **PAN**el and **GPI**b initializations. This feature, called TEKSECURE Erase Memory, is also executable from the front panel; see the Operators manual for more information.
	GPIb	COMMAND ONLY. Causes all bus unique commands to be initialized to known states as follows: **PAT**h **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.
	PANel	COMMAND ONLY. Will cause the scope to go to a factory preset front-panel setup. This command can help in creating new test setups; the scope is initialized to a known setup and the desired changes made to this setup.

EXAMPLE

Query	Response
ID?	ID TEK/2440,V81.1,"01-OCT-90 V2.40/2.5"

Table A-7: Miscellaneous Commands (Cont.)

Header	Argument	Description
INIT (cont)	**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.
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.
LONg	[**ON**] **OFF**	Determines whether a response to a query is given with unabbreviated symbols, **ON**, or not.
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, allowing the programmer to read just the scope setup part of the response into her program without having to strip off unwanted parts.
REM	"ascii string"	COMMAND ONLY. Does nothing but throw away the ascii string. Useful for introducing comments into user programs.
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.
TIMe?		QUERY ONLY. Returns the internal clock time in an ascii string. The number returned 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.

EXAMPLE

Query	**Response**
TIME	TIME″19:54″

Table A-8: Output Commands

Header	Argument	Argument	Description
DEVIce	**GRA**t	[**ON**] **OFF**	Determines whether the graticule is printed or not.
	PAGesize	**A4** **US**	Determines whether the page size is **US** (8.5 × 11) or the European **A4** size.
	SETTIngs	[**ON**] **OFF**	Determines whether instrument settings are 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.
	TYPe	**THI**nkjet **HPGl**	Shows which device the scope will format its output for when a **PRI**nt command is received.
	WAVfrm	[**ON**] **OFF**	Determines whether waveforms are printed or not.
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 **PRI**nt 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

Table A-9: Save Reference Commands

Header	Argument	Argument	Description
REFDisp	**REF1**	**EMP**ty	The **EMP**ty command will erase the contents of the selected reference.
		[**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 a SRQ is issued if **EXR** is **ON**.
	REF2	[**ON**] **OFF** **EMP**ty	
	REF3	[**ON**] **OFF** **EMP**ty	
	REF4	[**ON**] **OFF** **EMP**ty	
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

Table A-9: Save Reference Commands (Cont.)

Header	Argument	Argument	Description
REFPos	**MOD**e	**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.
REFPos	**REF1** **REF2** **REF3** **REF**	<NR3> <NR1> <NR1> <NR1>	<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).
SAVERef	**REF1** **REF2** **REF3** **REF4** [**STAC**k]		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**.

EXAMPLES

Query	**Response**
REFPOS?	REFPOS MODE:INDEPENDENT,REF1:5.12000E+2, REF2:4.60000E+2,REF3:5.59000E+2, REF4:4.57000E+2

Table A-10: Sequencer Commands

Header	Argument	Argument	Description
FORMat	[**ON**] **OFF**		If **FORM**at is **ON**, formatting characters including (CR,LF, and spaces) will be inserted into the text returned for a **PRG**m? query. Also, the sequence is compressed; that is, after the first step, only the sequence step settings that change from the previous step settings will be printed. 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

Table A-10: Sequencer Commands (Cont.)

Header	Argument	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, the incoming sequence is ignored. If no "ascii string" name is sent, all sequences will be returned. If the named sequence is not present, an error SRQ will be returned if **EXR** is **ON**. Any 2440 or controller receiving binary block transfers should be set up to not recognize the line feed as the terminator character. See Section 2 for information on the terminator character. *NOTE* *The LLPRGM binary block form will speed up sequence transfers in a production environment but an 'archive' taken using the PRGm? query is recommended for upward compatibility with future firmware releases. The binary images of the scope setup may change with future releases.*

Table A-10: Sequencer Commands (Cont.)

Header	Argument	Argument	Description
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**, formatting characters will be inserted into the text to make a very readable listing of the sequence and the sequence steps will be compressed (see **FORM**at command). The complete path (see **PAT**h) will always be printed regardless of whether **PAT**h is **ON** or **OFF**, and response will be in the unabbreviated or **LON**g form, regardless of whether **LON**g is **ON** or **OFF** (see **PAT**h and **LON**g under "Miscellaneous Commands"). If **FORM**at is **ON**, any 2440 or controller receiving ascii transfers should be set up to not recognize the line feed as the terminator character. See Section 2 for information on the terminator character.

Table A-10: Sequencer Commands (Cont.)

Header	Argument	Argument	Description
SETUp	**ACT**ion	<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 **ACT**ion number is associated with the next step or sequence. Only change the **ACT**ion number when new actions are needed. The <NR1> number sent with the **ACT**ion command may be calculated as shown in this example: For the actions Selfcal, SRQ, Pause, and Bell the **ACT**ion number would be 2 + 64 + 128 + 32 = 226. At power-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 Auto Setup. 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.
	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

Table A-10: Sequencer Commands (Cont.)

Header	Argument	Argument	Description
SETUp (cont)	**DELE**te	"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**.
	FORCe	[**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.
	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"

EXAMPLES

Query	Response
SETUP? MEMORY	SETUP MEMORY:12062
SETUP? NAMES	SETUP NAMES:"TEST",NAMES:"TEST2"

Table A-10: Sequencer Commands (Cont.)

Header	Argument	Argument	Description
SETUp (cont)	**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.

Table A-10: 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.

Table A-11: Service Request Commands

Header	Argument	Description
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.
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.
DEVDep	[ON] **OFF**	Enables/disables a device dependent SRQ such as the one generated when a Transmit or Abort button is 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.
EXR	[ON] **OFF**	If **ON**, this mask will allow the scope to assert an SRQ whenever an execution error is detected. An example of an execution error is requesting a waveform from an empty save reference waveform.
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.

EXAMPLES

Query	**Response**
BUSY?	BUSY OFF
CER?	CER OFF
EVENT?	EVENT 0;EVENT 467
EXW?	EXW ON
INR?	INR OFF

Table A-11: Service Request Commands (Cont.)

Header	Argument	Description
LOCk	**ON** **OFF** **LLO**	Controls the front-panel lock state. If **ON**, the front-panel controls are locked out. **LLO** is the power-up default and indicates that the scope will lock out the front panel whenever the universal GPIB command, **LLO**, is received. **OFF** unlocks the front panel (certain commands will cause the scope to lock its panel anyway: Self Cal, Auto Setup, Print/Plot commands, etc).
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.
PID	[ON] **OFF**	Enables/disables an SRQ generated by a probe identify button being pushed.
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.
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
LOCK?	LOCK LLO;LOCK ON
PID?	PID OFF
RQS?	RQS ON
USER?	USER OFF

Table A-12: Trigger Commands

Header	Argument	Argument	Description
ATRigger	**ABSE**lect	**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.
	CLRstate		COMMAND ONLY. Resets the trigger-state variable to ARMED.
	COUpling	**AC** **DC** **HFR**rej **LFR**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**.
	HOLdoff	<NR3>	<NR3> is setable 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.

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?COUPLING	ATRIGGER COUPLING:AC

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
ATRigger (cont.)	**LEV**el	<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.
	LOGsrc	**A.B** **OFF** **WOR**d	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.
	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).
	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.

EXAMPLES

Query	Response
ATRIGGER? MINIMUM	ATRIGGER MINIMUM:-9.01E+1

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
ATRigger (cont.)	**MOD**e	**AUTO**	**AUTO** switches to Roll at Sec/Div settings of 100 ms and slower. There is no **AVG** Acquire Mode during Roll.
		AUTOLevel	Sending an **AUTOL**evel command while **AUTOL**evel is selected forces a recalculation of the trigger level. See **INITA**t50 command also.
		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.
	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.
	SLOpe	**MINU**s **PLU**s	

EXAMPLES

Query	Response
ATRIGGER? MODE	ATRIGGER MODE:SGLSEQ

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
ATRigger (cont.)	**SOU**rce	**CH1** **CH2** **EXT1** **EXT2** **LIN**e **VER**tical	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**.
	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.

EXAMPLE

Query	Response
ATRIGGER? STATE	ATRIGGER STATE:RTRIG

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
BTRigger	**COU**pling	**AC** **DC** **LFR**ej **HFR**ej **NOI**serej	See **ATR**igger **COU**pling comments.
	EXTCLk	[**ON**] **OFF**	When **ON**, the B trigger source becomes the time base for the scope. The rate is one sample/ cycle.
	LEVel	<NR3>	See **ATR**igger **LEV**el comments.
	MODe	**RUNS**aft **TRI**gaft	Selects whether B sweep free runs (**RUNS**aft) or must be triggered to run (**TRI**gaft).
	SLOpe	**PLU**s **MINU**s	See **ATR**igger **SLO**pe comments.
	SOUrce	**CH1** **CH2** **WOR**d **VER**tical **EXT1** **EXT2**	See **ATR**igger **SOU**rce comments.
EXTGain	**EXT1**	**DIV1** **DIV5**	
	EXT2	**DIV1** **DIV5**	
	POSition	<NR1>	See **ATR**igger **POS**ition comments.

EXAMPLE

Query	Response
BTRIGGER?	BTRIGGER MODE:TRIGAFT,EXTCLK:OFF, SOURCE:CH1,COUPLING:DC,LEVEL:-9.53E-2, SLOPE:PLUS,POSITION:16

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
INITAt50			COMMAND ONLY. Will perform an autolevel cycle which will set the selected (A or B) trigger level halfway between the maximum of the signal and the minimum.
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 **ATR**igger? **STAT**e query.
SETTV	**ICO**upling	**ALT** **FLD1** **FLD2** **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.
	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.
	LCNTReset	**BOT**h **F1O**nly	Reset the line count only on field 1. Reset the line count on both field 1 and field 2.
	LCNTStart	**PRE**fld	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.
	NICoupling	**FLD1** **TVL**ine	Shows the type of display when the video signal applied is a noninterlaced signal.
	SYNc	**MINU**s **PLU**s	Show direction of the sync pulse relative to the baseline.

Table A-12: Trigger Commands (Cont.)

Header	Argument	Argument	Description
SETTV (cont.)	**TVC**lamp	[**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.
		ATFld	Line count begins at the sync pulse (non-SYSTEM M).
SETWord	**CLO**ck	**ASY**nc	**ASY**nc allows the probe to "free-run" at approximately 10 KHz.
	PROBe		QUERY ONLY. Returns **ON** if the word probe is attached or **OFF** if the probe is not connected.
	RADix	**HEX** **OCT**	Shows which numbering scheme will be used in the display.
		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 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**.

EXAMPLE

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

Table A-13: Vertical Commands

Header	Argument	Argument	Description
BWLimit	**TWE**nty **HUN**dred **FULl**		
CH1 (or **CH2**)	**COU**pling	**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**	
	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.

EXAMPLES

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

Table A-13: 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.
	VOLts	<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 Volts/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 mV and 200 mV 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

Table A-13: 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.
VMOde	ADD	[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.
	CH1	[ON] OFF	**ADD** and **MUL**t are mutually exclusive. The selection of one causes the other to be turned off.
	CH2	[ON] OFF	
	DISPlay	XY YT	
	MULt	[ON] OFF	

EXAMPLES

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

Table A-14: 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
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****, the data returned will be smoothed.*

Table A-14: Waveform Commands (Cont.)

Header	Argument	Argument	Description
DATa	DSOUrce SOUrce	< data src > < data src >	**DAT**a **SOU**rce specifies the source (**CH1**, **CH2**, 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 the parameter will be extracted from. (See **VAL**ue on page A-12.) 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 command). If secondary source is an empty reference memory, an error will be returned when the information is requested if **EXR** is **ON**. < data src > available are: **CH1**, **CH2**, **ADD**, **MUL**t, **REF1**, **REF2**, **REF3**, **REF4**, **CH1D**el, **CH2D**el, **ADDD**el, and **MULTD**el.

Table A-14: Waveform Commands (Cont.)

Header	Argument	Argument	Description
DATa	**ENC**dg	**ASC**ii	
		RPBinary	**RPB**inary is positive integer.
		RIBinary	**RIB**inary is two's complement.
		RIPartial	**RIP**artial is partial format for **RIB**inary.
		RPPartial	**RPP**artial is partial format for **RPB**inary. *NOTE* *For information on how to use the different data encoding formats and the ranges for each format, see the discussions on Waveform data formats and scaling data points in Appendix B.*
	TARget	**REF1** **REF2** **REF3** **REF4**	Targets which reference memory will receive the next waveform sent to the scope.
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 as user friendly as other types of transfers. 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 .
	DELTa	**BOT**h **CH1** **CH2**	
	ENCdg	**RIB**inary **RPB**inary **RIP**artial **RPP**artial	
	NORmal	**BOT**h **CH1** **CH2**	
	OFF		

EXAMPLE

Query	**Response**
DATA? ENCDG	DATA ENCDG:RPPARTIAL

Table A-14: Waveform Commands (Cont.)

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.
WFMpre	**BN.F**mt	**RI** **RP**	**BN.F**mt tells the scope how to handle binary curve data coming in. **RI** says treat it as a two's 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). These ranges are maximum and assume a vertical window for measurements of 10.24 divisions; some acquisition rates have a smaller vertical window, reducing the range of data available. See Vertical Window on page 3-16. *NOTE* *The scope powers-up expecting RI binary data. If RP is to be sent, first send a BN.Fmt command with RP as the argument to change the scope.*
	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: **RIB**inary, **RPB**inary, **RI-P**artial or **RPP**artial.

EXAMPLE

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? ENCDG	WFMPRE ENCDG:ASCII

Table A-14: Waveform Commands (Cont.)

Header	Argument	Argument	Description
WFMpre (cont)	**NR.P**t	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 remaining missing points 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	**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.
		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).
	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 will be rounded and a number outside the range will be set to the closest valid number. If either of these 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.

Table A-14: Waveform Commands (Cont.)

Header	Argument	Argument	Description
WFMpre (cont)	**WFId**	"ascii string"	The **WFId** field in the **WFM**pre is left undefined by Codes and Formats so individual instruments can 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.
	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 4.0E-11 (2 ns/div). <NR3> values outside this range will be set to the closest valid number and 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.
	XUNit	**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.

Table A-14: Waveform Commands (Cont.)

Header	Argument	Argument	Description
WFMpre (cont)	**YOF**f	<NR3>	**YOF**f locates the ground in digitizing levels relative to data 00. If **YOF**f is positive, then ground is above center screen. <NR3> can range from -2500 to 2500 with a resolution of 0.25. Values outside this range will be set to the closest valid number and a warning SRQ will be issued if **EXW** is **ON**.
	YMUlt	<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.
	YUNit	**DIV** **V** **VV**	Sets the magnitude in volts when associated with **YMU**lt.

EXAMPLE

Query	**Response**
WFMPRE? YOFF	WFMPRE YOFF:5.450E+1
WFMPRE? YMULT	WFMPRE YMULT:4.000E-2

Table A-15: Waveform Data Commands

These commands are used as either measurement querys that fetch waveform data (**MAX**imum, **VMIN**inum, etc.) or to setup, or query the setup of, certain elements that determine how or where on the waveform the measurements are taken (**STAR**t, **LEV**el, etc.). The **DAT**a **SOU**rce variable shows which waveform will be looked at for all waveform data queries.

When calculations are performed on waveforms, points at the positive or negative extremes of the vertical measurement window are considered out-of-bounds. In other words, if the vertical window is 256 digitizing levels (10.24 divisions), any points at the -128 and the +127 levels are ambiguous, since they could represent points that are outside the vertical window or clipped. These points are ignored when responses to measurement queries are calculated and returned; limit values are only returned for measurement queries in order to indicate an out-of-bounds waveform (see **MAX**imum?) or for setup-type queries (see **LEV**el?). Since the size of the vertical window varies with acquisition rate (SEC/DIV setting), the actual digitizing level considered out of bounds also varies. Table 3-1 in Section 3 gives the window limits (range) in digitizing levels and divisions.

Header	Argument	Description
AVG?		QUERY ONLY. Returns an <NR3> number between the negative and positive level limits (inclusive) of the vertical window for the acquisition rate currently in effect. This number represents the average of the points between **STAR**t and **STO**p. If all the waveform points within the requested interval are at or beyond the vertical window boundaries, -128 is returned to indicate the direction of the out of bounds waveform.
DIRection	**PLU**s **MINU**s	Shows which direction to proceed with the search for a **PCR**oss or **NCR**oss. **PLU**s will go from **STAR**t to **STO**p and **MINU**s will go from **STO**p to **STAR**t. The definitions for positive and negative crossings are unaffected by the direction of the search.

EXAMPLES

Query	Response
AVG?	AVG 3.55370E+1
DIRECTION?	DIRECTION PLUS

Table A-15: Waveform Data Commands (Cont.)

Header	Argument	Description
HYSteresis	< NR1 >	< NR1 > may range from 0 to the number of digitizing levels in the vertical window for the acquisition rate in effect. It specifies the number of digitizing levels (increments) that a curve must go below (or above for **NCR**oss) the value specified by **LEV**el before the search is started for the level crossing. < NR1 > is set to 5 at power-up. (See Making Custom Measurements on page 3-14 for more information.)
LEVel	< NR1 >	< NR1 > varies between the negative and positive level limits (inclusive) of the vertical window for the acquisition rate in effect. It sets the vertical level for **PCR**oss and **NCR**oss.
MAXimum?		QUERY ONLY. Returns a < NR1 > number between the negative and positive level limits (inclusive) of the vertical window for the acquisition rate currently in effect. This number represents the maximum value of the acquired waveform between **STAR**t and **STO**p. If all waveform points are at the positive limit level, or if they are all at the negative limit level, that limit level is returned to indicate the direction of the out-of-bounds waveform.
MINImum?		QUERY ONLY. Returns a < NR1 > number between the negative and positive level limits (inclusive) of the vertical window for the acquisition rate currently in effect. This number represents the minimum value of the acquired waveform between **STAR**t and **STO**p. If all waveform points are at the positive limit level, or if they are all at the negative limit level, that limit level is returned to indicate the direction of the out-of-bounds waveform.

EXAMPLES

Query	Response
HYSTERESIS?	HYSTERESIS 5
LEVEL?	LEVEL 23
MAXIMUM?	MAXIMUM 65

Table A-15: Waveform Data Commands (Cont.)

Header	Argument	Description
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. Also, see the **HYS**terisis waveform data command.
PANS		COMMAND ONLY. Positions the time cursors to correspond to the current position numbers for **STAR**t and **STO**p. You can then verify on screen whether or not the portion of the waveform you wish to transfer is bracketed by the **STAR**t and **STO**p values when doing partial waveform transfers (see Appendix B).
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. Also, see the **HYS**terisis waveform data command.
SNAp		COMMAND ONLY. Sets the **STAR**t and **STO**p numbers to correspond to the position numbers for the left and right time cursor respectively. The numbers range from 0-1024. Useful for doing partial waveform transfers (see Appendix B).
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 not, 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**.

EXAMPLES

Query	**Response**
NCROSS?	NCROSS 360
PCROSS?	PCROSS 0,PCROSS 302
START?	START 1

Table A-15: Waveform Data Commands (Cont.)

Header	Argument	Description
STOp	<NR1>	<NR1> sets the end of the interval for the waveform measurement queries (see **STAR**t).
VAVg?		QUERY ONLY. Similar to **AVG**? except **VAV**g returns a value in the same units shown by the volts cursor display. 99e99 is returned if all points are outside the vertical window or if the channel that acquired the waveform has a variable gain setting other than fully calibrated.
VMAximum?		QUERY ONLY. Similar to **MAX**imum? except **VMA**ximum returns an <NR3> number that represents the maximum voltage of the acquired waveform between **STAR**t and **STO**p. Returns 99e99 if all points are outside the vertical window or if the channel that acquired the waveform has a variable gain setting other than fully calibrated.
VMInimum?		QUERY ONLY. Similar to **MINI**mum? except **VMI**nimum returns an <NR2> number that represents the minimum voltage of the acquired waveform between **STAR**t and **STO**p. Returns 99e99 if all points are outside the vertical window or if the channel that acquired the waveform has a variable gain setting other than fully calibrated.

EXAMPLES

Query	Response
STOP?	STOP 1024
VAVG?	VAVG -7.56965E-1
VMAXIMUM?	VMAXIMUM 4.200E-1
VMAXIMUM?	VMAXIMUM 99e99
VMINIMUM?	VMINIMUM -2.180

B
Discussion on Waveforms

Discussion on Waveforms

A waveform transfer is a transmission of a block of waveform data between the 2440 and the controller or between one 2440 and another scope. This block contains data points which trace the waveform shape. You can tell the 2440 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).

We cover several considerations involved in waveform transfers. First, in Doing a Waveform Transfer the DIRECTION (2440-to-controller, controller-to-2440, and 2440-to-2440) is discussed. Second, Waveform Data Formats discusses the five types of ENCODING that can be used for waveform data and how data points are REPRESENTED in those formats. Third, Scaling the Data Points details how the data words corresponding to data point values are SCALED into voltages.

Doing a Waveform Transfer

In this subsection, we describe transferring waveform data between a controller and a 2440, and between two 2440's. During these discussions, we treat the waveforms as strings to be sent back and forth (later subsections describe what the strings mean and how to interpret them).

From the 2440 to a Controller

First, you need to tell the 2440 which waveform you wish to transfer. Use the DATA SOURCE command to specify the source. For example, let's say you want the waveform in CH1. First, you send DATA SOURCE:CH1 to the 2440 to set the source pointer to CH1. Next, you need to transfer the data. Send a CURVe? query to the 2440 to start the transfer, and then read the results. You can read more about the DATA SOURCE command and CURVe? query by looking on pages A-54 and A-53 respectively.

This short sample program, written in BASIC using the National GPIB drivers, transfers a CH1 waveform from the 2440 to the controller (the 2440's address should be set to DEV1's primary address set in IBCONF supplied with the National Drivers). After the program has run, the controller has the CH1 waveform stored in the string array named WAVE$70. Lines 360-390 send the waveform back to the scope's REF1 using the TARGET command described below. Lines 100-230 can be found in the example on page 2-6.

```
300 WAVE$= SPACE$(1040)
310 CMD$ = "DATA SOURCE:CH1,ENCDG:RIB"
320 CALL IBWRT (DEV%,CMD$)
330 CMD$ = "CURVE?"
340 CALL IBWRT (DEV%,CMD$)
350 CALL IBRD (DEV%,WAVE$,1040)
360 CMD$ = "DATA TARGET:REF1"
370 CALL IBWRT (DEV%,CMD$)
380 INPUT "PRESS ENTER TO SEND WAVEFORM TO REF1",A$
390 CALL IBWRT (DEV%,WAVE$)
400 END
```

From the Controller to the 2440

Here, the source of the waveform is the controller and we wish to download the waveform it has stored to the 2440. For this transfer, you need to tell the 2440 where you want the waveform stored. There are four options, REF1 through REF 4, which are the 2440's four reference memories. Only reference memories can accept incoming waveforms.

Use the DATA TARGET command to select the memory where the waveform is to be stored. For example, let's say you wish to store the waveform in the 2440's reference memory 1. First you send DATA TARGET:REF1 to select the target memory, then you transmit the waveform to the 2440. The DATA TARGET command is described on page A-55.

There are a few things to consider when sending waveforms to the 2440. First, if you want to see the waveform on screen, you must display the target reference memory on screen (see the REFDISP command on page A-32). Keep in mind that waveform transfers to the 2440 are slower if the target reference is displayed. Second, if you are sending waveforms in binary format you must tell the 2440 what kind of binary data it is. Read about waveform data formats starting on page B-5 if you don't know what type of binary data you are using. To prepare the 2440 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 2440 expects to receive binary data in signed integer (RI) binary format. See page A-56, for more information on the WFMPRE command.

From 2440 to 2440

You can transfer waveforms from one 2440 to another 2440 over the GPIB without a controller. Waveforms can also be sent from the 2440 to other oscilloscopes that understand the waveform encoding formats the 2440 uses. The Tektronix 2430, 2430A, 2432, and 7D20 can receive 2440 waveforms.

The following steps generally describe how to do a scope to scope transfer. To determine specifically how to set up the 2440's bus and waveform format modes (Step 1), read through Setting the GPIB Mode on page 2-1.

To perform a scope to scope transfer, do the following:

1. Set the "From" scope's bus mode to talk only (T/ONLY) and waveform format mode to Send Waveform Curve and Waveform Preamble (W/WFMPRE).
2. Set the "To" scope to listen only.
3. Display the signal source containing the waveform you want to send on the screen. Do not display any other signal sources (a signal source is CH1, CH2, ADD, MULT, and REF1-REF4) on screen.
4. If you want to watch the transfer, you must put the target REF memory for the "To" scope on screen. Push the front-panel button **OUTPUT**, then the menu button STATUS. Note the REF memory listed for TARGET. Push the front-panel button **DISPLAY REF** and set the noted REF on.
5. Push TRANSMIT in the OUTPUT menu to send the waveform to the the target memory and the screen if the REF was displayed from Step 4.

WAVEFORM FORMAT—The procedure just detailed covers how you should send waveforms between two scopes. By setting the waveform format to send both the curve and the waveform preamble (Step 1), the "To" 2440 determines the waveform encoding (ASCII, RPBINARY, or RIBINARY) the "From" scope is using and uses the same coding. If the preamble isn't sent, the "To" scope isn't set to match the "From" scope's encoding, nor is the Volt/Div and Sec/Div readouts for the target REF updated to match those for the waveform being sent. The result is misleading readouts, and possible offset waveforms because of different encoding for the "To" scope. Remember to make sure that you set the waveform format to W/WFMPRE and such problems are avoided.

WAVEFORM SOURCE – You may have noticed that the procedure does not have you select a source for the "From" scope. The 2440 sends all the waveforms displayed on screen (*for transfers without a controller only*) to the target REF memory of the "To" scope. It ignores the setting for SOURCE that appears in the GPIB Status menu. For this reason, you should only display the waveform you want to send on the "From" scope's screen and no others (Step 3) since the last waveform sent will be the only one stored in the target REF memory.

WAVEFORM TARGET – The target for the "To" scope was not specified in our procedure either. The target for the 2440 can only be changed by a controller, so you must use the REF memory listed for TARGET in the GPIB Status menu. Use the method outlined in Step 4 to determine the target and display it, if desired.

There is another consideration related to the target not being settable for transfers without controllers. If you display more than one waveform on screen ("From" scope) and push TRANSMIT, the waveforms are sent consecutively, each one overwriting the other, and only the last one sent is saved in the target REF memory.

The way to transfer several waveforms is to display each signal source containing a waveform on the screen of the "From", but only one source at a time. Send the displayed waveform, but when the transfer is complete, move the transferred waveform out of the "To" scope's target REF memory into an alternate REF memory. Repeat this process of displaying, sending, and moving the waveforms until each waveform you want to send is stored in a different REF memory. Using this method, you can send up to four transferred waveforms in REF1-REF4 of the "To" scope. See "SAVE" in Section 5 of the Operators Manual for information on moving waveforms between Reference memories.

Waveform Data Formats

Waveform data may be encoded in several different formats. The formats define how a waveform is to be represented in a data block. Waveform formats available on the 2440 include ASCII and four different binary format. The binary formats are RIBINARY, RPBINARY, RIPARTIAL, or RPPARTIAL.

ASCII format uses signed integer data-point representation. The 2440 sends only whole waveforms for ASCII format, although the controller can send partial waveforms to the 2440. 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, and RPPARTIAL sends partial waveforms in positive integer data-point representation.

You can select any one of these waveform encoding formats over the GPIB by using the DATA ENCDG command. For example, to set the 2440 to RPBINARY format, simply send the command DATA ENCDG:RPBINARY.

Data Point Representation

Before we look at the ASCII and binary formats just introduced, we need to understand the different methods for representing the waveform data points or samples. Knowledge of these methods is needed because the different waveform encoding formats use different data representation methods.

The 2440 has an A-D converter that always resolves waveforms to 1/25 of 1 vertical division. This means waveforms are displayed on the instrument screen with a vertical resolution of 25 values (also called digitizing levels) per division. The total number of values available for making waveform measurements is the vertical window size and is different for different ranges of acquisition rate settings. It is always greater than the 8 divisions available on screen, and varies from a minimum of 240 values (9.68 divisions) to a maximum of 256 values (10.24 divisions). (See page 3-16 for a discussion of vertical windows).

Each value within the vertical window can be expressed in positive integer or signed integer.

SIGNED INTEGER REPRESENTATION—In this format, the data value at center screen is zero. Any values below center screen are negative and any above it are positive. The range of the values will depend on the size of the vertical window and, therefore, the acquisition rate. As an example, a 10.24 division window will range from -128 to 0 to 128, with -128 being one division below the bottom of the 2440 screen, 0 being at center screen, and 128 being one division above the top of the screen.

POSITIVE INTEGER REPRESENTATION – In positive integer format, the center screen value is considered to be at the 128 data value. As before, the range varies with the vertical window; using the 10.24 division window again as an example, 0 starts one division below the bottom of the 2440 screen, 128 is at center screen, and 255 is one division above the top of the screen. See Table 3-1 for ranges.

ASCII Format Encoding

In ASCII format, each waveform point is composed of up to four ASCII characters that describe a signed integer data point value. There is one ASCII character for each digit in the signed integer data point value, plus an extra one for the minus sign if the data value is negative (positive signed integers do not have a plus ("+") sign prefixed).

Let's look at an example. Assume a waveform data point is one division below center screen. Its signed integer representation is -25 (remember, 25 digitizing levels per division). In ASCII, that point would be expressed as an ASCII minus sign ("-"), followed by the ASCII number 2, followed by the ASCII number 5 ("-25"). A waveform point one division above center screen would be a positive 25 (remember, center value is zero) signed integer and its ASCII representation would be the ASCII number 2, followed by the ASCII number 5 (note no ASCII plus sign is prefixed).

The entire waveform transfer is made of binary equivalents of the ASCII characters that comprise the data points and the commas separating those data points (data points are always separated by commas). Each ASCII character requires one byte to represent it. Therefore, since each digit in a signed integer data value requires one ASCII character, and each ASCII character requires one byte, data point binary representation can require up to 4 bytes (three digits, plus a minus sign).

ASCII format is selected by sending a DATA ENCDG:ASCII command to the 2440.

Let's look at another example. Consider a DC level consisting of all 1024 data points 5 divisions below center screen. All the data point values for the DC level equal -125. The total number of characters is 5 (for the word "curve" in header) + 1024 × 4 (three digits plus the minus sign) + 1023 commas, or approximately 5100 bytes.

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 (bytes 7 and 8) and the second is 54 (bytes 10 and 11). Each value is separated by a comma. These two points are followed by an additional 1021 points. These points are not shown but are located between the two consecutive commas following byte 11. The last point is 41 and it's followed by the specified message terminator.

Since the 1021 data values between the commas could be any allowed value, the number of bytes needed to express them is not known. Therefore, the byte numbers are given as "XXXX" (unknown). Also, the CURVE part of the header will be absent if PATH is OFF (see PATH command on page A-30).

Table B-1: ASCII Waveform Transfer Example 1

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

There are four binary formats for sending waveforms. RIBINARY and RPBINARY are used to send an entire 1024 point waveform. RIBINARY uses the previously described signed integer method for representing the data points while RPBINARY uses positive integer representation. RIPARTIAL and RPPARTIAL are used to send partial waveforms with RIPARTIAL using signed integer representation and RPPARTIAL using positive integer representation. In all 2440 binary formats, each waveform point is represented by one 8-bit data byte.

Since we have already explained the difference between signed integer (RI) and positive integer (RP) representation, let's discuss entire and partial waveform transfers.

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. It gives the the total number of all data bytes plus the checksum byte. The byte count is a 2-byte field. Waveforms sent from the 2440 always have the first byte set to 04 hexadecimal and the second byte set to 01 hexadecimal. The whole field is a 16-bit binary number (0401h, 1025 decimal) 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. Waveforms sent from the 2440 contain 1024 data bytes.

c is the 8-bit checksum. To get the checksum, both bytes of the byte count, plus all the data bytes, are summed. Then the two's complement is taken of the least significant 8 bits of the sum to yield the checksum. 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 be 1025 (401h). The CURVE part of the header will not be present if PATH is OFF.

Table B-2: ASCII Waveform Transfer Example 2

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. These two encoding modes can be used by the controller to receive partial waveforms from, or send partial waveforms to, a 2440. Partial waveforms cannot be sent in non-controller directed transfers (i.e. 2440-to-2440 transfers).

The portion of the acquisition sent is determined by the data point numbers for the START and STOP commands. For example, if START is set to 256 and STOP is set to 512, which are the power-up values, a partial binary waveform format 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). See pages A-62 and A-63 for more information on the START and STOP commands.

You may return a partial waveform in binary format to the 2440. Partial waveform transfers only update part of the reference waveform currently in the 2440. To force which bracket of data points of the 1024 available are sent, you can use the START and STOP commands to set the beginning and ending data points for the data string (beginning/ending correspond to earliest/latest).

NOTE

> *You can't specify START and STOP for partial transfer if you use ASCII waveform encoding. ASCII transfers will transfer the partial waveform and then complete the 1024 point waveform record by replicating the last data point sent until the record is full.*

Let's look at an example of a partial waveform and introduce an alternate method for setting the start and stop points for the transfer. Assume a waveform is on the 2440 screen in CH1. 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 2440. The START data point number is now equal to that for the point the left cursor is positioned to; the STOP data point number is now equal to that for the one positioned at the right cursor.

Once the start and stop values are set, you can request a partial waveform from CH1. You should use DAT ENC to set RPPartial or RIPartial, and then use CURV? or WAV? to fetch the partial waveform to the controller. When you send this CH1 partial waveform back, the segment of the waveform stored in the target REF memory that corresponds to the START and STOP values will be replaced by the partial waveform. When the target REF is displayed, the segment between START and STOP will look just like the corresponding segment of CH 1.

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 value 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 is 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 set to 256 and STOP set to 512. The CURVE part of the header will not be present if PATH is OFF.

Table B-3: Binary Waveform Transfer Example

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	
13	SOH	1	01	
14	NUL	0	00	
15	5	53	35	
16	T	84	54	
		...		
		... 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

So far, we have treated waveforms as data strings transferred between controllers and 2440's. All the data values have been 8-bit numbers that correspond to digital levels (abbreviated DL's) in the 2440's digitizing system. Now we will discuss how these digital levels are converted into voltages.

The waveform preamble contains various information fields. These fields are descriptive information about the waveform the preamble is associated with. This information details characteristics about the how the waveform was acquired, such as where the trigger is located in the record, what the waveform encoding is, what the voltage increment is between consecutive digitizing levels, where ground (0 volts) level is relative to the zero data value level, etc. It is this information that lets the 2440 accurately display waveforms sent to it and allows controller programs to process waveform data it uploads. We use part of this information to scale 2440 waveforms.

The waveform preamble and the data values can be fetched from the 2440 using the WAVe? query (recall that CURVe? only returns the data values). Also, the WFMPRE? query can be used to retrieve all, or just certain, fields of the waveform preamble only, and all or part of the preamble can be shipped to the 2440 to scale (and otherwise set up the scope) to display the waveform. The information and data returned by WAVe? and WFMPRE? queries applies to the currently selected DATA SOURCE. Information sent to a 2440 applies to the currently selected DATA TARGET.

For more information about the waveform preamble and the information found in its fields, read the descriptions for the arguments belonging to the WFMPRE query on page A-56. Also, see WAVfrm? and CURVe? on pages A-56 and A-53.

Sample Conversion to Volts

One of the most frequent uses for waveform preamble information is to convert the waveform data point values from either signed integer or positive integer values into voltage values. To correctly convert the waveform, you need to know the voltage between each digitizing level in the vertical window, where ground is positioned in the window, and whether the waveform encoding uses signed integer or positive integer representation.

We get the vertical scale from the YMULT field of the preamble. As mentioned, it is the voltage between two consecutive 2440 digitizing levels. At, say 2 Volts/Div, YMULT would equal 2 Volts/Div divided by 25 DL's/Div or .04 Volts/DL.

The YOFF field in the preamble tells how far ground (for the waveform) is positionally offset from center screen. If, say, ground level was offset 1 division below center screen, YOFF would equal -25, since there are 25 DL's per division. YOFF is always relative to center screen, regardless of signed or positive representation for the waveform data values.

The range of YOFF is from -2500 to +2500 DL's with a resolution of 0.25 of a DL. This large range is needed because the 2440 can vertically position waveforms from 10 divisions above, to 10 divisions below, center screen, yielding ±250 DL's of positioning (±10 division × 25 DL's/Div). Also, the 2440 can vertically expand averaged waveforms up to 10 times, providing ±250 DL's × 10 or ±2500 DL's of positioning range. The 2440 positions waveforms vertically with a resolution of 10 bits, instead of the 8-bit resolution for data values. The extra 2 bits provides 4 times the resolution that the digitizer does.

NOTE

Don't think that you can position a waveform outside the vertical window and read a valid data point value for the out-of-window data points. Data points positioned outside the vertical window will be clamped to the minimum (below the window) or maximum (above the window) data value available, based on the encoding method used, and the vertical window available at the acquisition rate in effect. (See Table 3-1 for the limit values for both encoding methods.)

The BN.FMT field tells whether signed integer or positive integer representation is used for the data values. The field will be RI for those waveform encoding formats using signed integer (ASCII, RIBINARY, and RI PARTIAL) and RP for those using positive integer (RPBINARY and RPPARTIAL).

A simple calculation is used to first remove the position offset (YOFF) from the data point value and then apply the vertical scaling multiplier (YMULT). Since Signed Integer representation considers the zero data value or ground to be at center screen, the YOFF value is subtracted directly from the data point value to remove the offset. Here is the formula to convert a single data point value to a voltage for waveforms using signed integer data representation:

$$\text{Voltage} = (\text{point value} - \text{YOFF}) \times \text{YMULT}$$

Waveforms using positive integer representation consider center screen to be the data value 128. In other words, this method of data representation considers the center screen data value to be 128 digitizing levels above its zero digitization level. Therefore, we calculate the voltage by first subtracting 128 from the data point value (effectively, converting the positive integer to a signed integer), then subtracting YOFF. The result is multiplied by YMULT. The formula for positive integer representation is:

Voltage = (point value - 128 - YOFF) × YMULT

Let's use these formulas on 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. The YOFF field shows the position offset to be 28 DL's below center screen. The YMULT field shows us that the voltage multiplier is 4.00E-2 volts per DL (1 Volt/Div divided by 25 Div/DL, in this case). Inserting the correct values into the first formula gives us:

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

This result corresponds with a waveform point that would be displayed 1 division below center screen (-25 DL's at 25 DL's/Div = -1 division) with no position offset, BUT with the ground reference being positionally offset slightly more than one division above center screen (28 points at 25 DL's/Div = 1.12 divisions). The actual position of the point would be 0.12 divisions above center screen.

Waveform Acquisition and Conversion Programming Example

Here is a simple 2400 Series DSO Waveform to ASCII file transfer program written in C. It produces ASCII output in generic format suitable for importing to Lotus, FrameWork, etc., and acquires a waveform and converts the values to voltage related to time.The program reads RIBINARY data (signed integer representation), converts it and then prints it to the screen. This example uses the National GPIB drivers to interface to the GPIB board.

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <graph.h>
#include <conio.h>
#include <dos.h>
#include "nigpib.h"   /* National Instruments GPIB drivers
                         Function Prototypes */

extern int ibcnt, ibsta, iberr;  /* Global variables used by
                                    NI drivers */
char wfm[1024];       /* raw binary waveform data array *
/
void main( void );    /* main prototype */

void main(){

  int i,              /* looping variable */
  bd;                 /* GPIB board Primary address */
  char cmd[200];      /* command string buffer */
  char resp[200];     /* response string buffer */
  char c,             /* single character buffer */
       hb,            /* high byte of byte-count */
       lb;            /* low byte of byte-count */
  int count;          /* number of bytes in waveform */

                      /* Waveform Preamble values: */
  float ymult,        /* y-multiplier:  volts / DL */
        yoff,         /* DC offset from ground */
        xincr;        /* x-increment: time / pt,
                         (sample interval) */
  int pt_off;         /* point offset: trigger position in
                         waveform */
```

```
/**********************************************************
 **************  Display Version Text  ********************
 **********************************************************/
  _clearscreen(_GCLEARSCREEN);
  printf ("\n\n\n              Waveform Acquisition &
Conversion Example\n");
  printf ("                           Version 1.00 NOV
90\n");
  printf ("                             Tektronix,
Inc\n\n\n");

/**********************************************************
 ********  Open device and find device type  **************
 **********************************************************/
bd=ibfind("DEV1");   /* Open device type */
  if(bd<0){
    printf("Board name GPIB0 no found.\n");
    printf("Check DEVICE=GPIB.COM in your CONFIG.SYS
file.\n");
    exit( 0 );
  }

  ibwrt(bd,"id?",3);  /* check to make sure device is 2400
                          series dso */

  if(ibcnt<3){
    printf("The GPIB is unable to send data to the
scope\n");
    exit( 0 );
  }

  memset(resp,0,50);
  ibrd(bd,resp,49);

  if( strstr( resp, "TEK/24" ) == NULL ){
    printf( "No Tektronix 2400 Series Scope
Found!\n\007\007");
    exit( 0 );
  }

/**********************************************************
 ******  Send CURVE? query and transfer data  *************
 **********************************************************/
  memset( cmd,  0, 100 );
  strcpy( cmd, "path off;data encdg:ribinary,
source:CH1;curve?");
  ibwrt( bd, cmd, strlen(cmd));
  /*
   * curve header = ´CURVE %hblb´
   *      where hblb is the 2-bytes which compose
   *      the byte count.
   * continue to read bytes, one at a time, until the
   * '%' symbol is read.
   */

  do{
    ibrd( bd, &c, 1 );
  }while( c != '%' );

  ibrd( bd, &hb, 1 );       /* read in high byte of byte
count */
  ibrd( bd, &lb, 1 );       /* read in low byte of byte count
*/
  count = hb*256+lb-1;      /* calculate byte-count */
  ibrd( bd, wfm, count);    /* read in raw binary waveform
data */
  ibrd( bd, &c, 1 );        /* read in checksum */

/*********************************************************
 *****  Send WFMPRE? query and transfer data  *************
 *********************************************************/
  memset( cmd,  0, 100 );
  memset( resp, 0, 100 );
  strcpy( cmd, "path off;wfmpre? ymult, yoff, xincr,
pt.off");
  ibwrt( bd, cmd, strlen(cmd));
  ibrd( bd, resp, 99);
  sscanf( resp, "%e,%e,%e,%d",
    &ymult, &yoff, &xincr, &pt_off );

/*********************************************************
 ********  Output scaled data  ***************************
 *********************************************************/
  /*
   * Output scaled x, y values in (Volts, Sec)
   * Volts[i] = (wfm[i] - yoffset) * volts / pt
   * Time[i] = [i - (trigger position)] * (time / pt )
   */
  printf( "VOLTS\t\tSECONDS\n" );
  printf( "ÍÍÍÍÍ\t\tÍÍÍÍÍÍÍ\n\n" );
  for(i=0;i<count;i++){
    printf("%5.2e\t%5.2e\n", (float)(wfm[i] - yoff) *
ymult, (float)(i-pt_off)*(float)xincr );
  }
}
```

C
Event Tables

Event Tables

Table C-1: 2440 Status Bytes

		Decimal				Priority	
Title	**Binary[1,2]**	**RQS Off**		**RQS On**		**RQS**	**RQS**
		Idle	**Busy**	**Idle**	**Busy**	**Off**	**On**
No Status To Report	000X 0000	0	16	0	16	2	1
Power On	010X 0001	1	17	65	81	2	9
Operation Complete	0R0X 0010	2	18	66	82	2	3
User Request	0R0X 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

[1] *"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 2440 is busy at the time the status byte is read. Anytime the 2440 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.

[2] *The first four binary bits are sent in the following order: Device Dependent Bit; RQS Bit; Error Bit; Busy Bit.*

Table C-2: Command Error Events

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 2440 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 in 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.

Table C-3: Execution Error Events

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 2440 will not assert SRQ.

Code	Description
203	I/O buffers full, output dumped.
250	Selected recall memory is unset.
251	Measurement requested on an empty reference memory.
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	No ADD or MULT on previously SAVEd waveforms; ENVELOPE waveform invalid.
260	No cal commands allowed while front panel is doing cal.
261	No sequence by that name to delete.
262	Can't save sequence – out of memory.
263	Can't send a partial waveform to an empty ref.
264	Not enough edges to extract the parameter.
265	Asked for rise time but no rising edge.
266	Asked for fall time but no falling edge.
267	Delay Measurement targets must have matching Sec/Div settings.
268	One or more of the following conditions are not satisfied: BASE $\leq$ PROXIMAL $\leq$ MESIAL $\leq$ DISTAL $\leq$ TOP, BASE $\leq$ MESIAL2 $\leq$ TOP, PROXIMAL > MIN and DISTAL < MAX, MIN < MESIAL2 < MAX
269	Repet waveform not filled when measurement requested.
270	No measurements during live Roll – enter Save mode first.
271	Measurement requested on a Delta Delay target but B Horizontal and Delta Delay modes are not on.

Table C-3: Execution Error Events (Cont.)

Code	Description
272	RMS measurement invalid due to 2440 internal overflow.
275	Sequencer currently active—new sequence commands not accepted.

Table C-4: Internal Errors

Internal Errors are issued when something has happened to the hardware of the 2440 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 2440 will not assert SRQ.

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

Table C-5: System Messages

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 2440 is currently asserting SRQ on the bus and the controller must read the status byte out before reading the event code.

Code	Description
401	2440 was just powered on.
459	There is an SRQ pending.

Table C-6: User Request Events

User Request events are issued when any of the bezel buttons on the 2440 front panel are pushed. The MENUOFF command needs to be issued before these events will be reported. This command allows the user to monitor front panel responses as well as to clear the menu for writing custom text to the screen when desired. Set USER to ON to receive SRQ's when any of these events occurs. If USER is OFF, the 2440 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).

Table C-7: Probe Identify Events

Probe Identify events are reported by the 2440 when the probe identify feature found on certain probes is actuated. You can replicate this action by grounding the outer code ring to the inner shell 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 2440 will not assert SRQ.

Code	Description
455	CH 1 probe identify was used.
456	CH 2 probe identify was used.
457	EXT 1 probe identify was used.
458	EXT 2 probe identify was used.

Table C-8: Operation Complete Events

Operation Complete events are issued when the controller needs to know when the 2440 has completed a task. Set OPC to ON to receive SRQ's when any of these events occurs. If OPC is OFF, the 2440 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.

Table C-9: Execution Warnings

Execution Warnings are issued when the command received has been done, but the result might not be what the user expected to see. Set EXW to ON to receive SRQ's when any of these events occurs. If EXW is OFF, the 2440 will not assert SRQ.

Code	Description
539	50 MHz bandwidth limit not available in 2440. Bandwidth limit set to 100 MHz.
540	RMS measurements need at least 1 period.
541	Amplitude too small to do an accurate timing measurement.
542	Measurement crossing points on Envelope may be misplaced. Turn on Marks to see where measurement was taken.
543	Too few points acquired to guarantee Histogram accuracy for this measurement.
544	Waveform has points off the top of vertical window.
545	Waveform has points off the bottom of vertical window.
546	Waveform has points off the top and bottom of vertical window.
547	Rising/Falling edge has too few points for optimal accuracy.

Table C-9: Execution Warnings (Cont.)

Code	Description
548	Min/Max method should not be used for Overshoot/Undershoot measurements.
549	Not enough samples taken to do accurate Time, Freq, Period, Pulse Width, or Delay measurement. Set Sec/Div faster if possible.
550	Only Delay 1 will be displayed if in Average.
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 or Envelope 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.

Table C-9: Execution Warnings (Cont.)

Code	Description
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.

Table C-10: Device Dependent Messages

Device Dependent messages are issued when the front panel user of the 2440 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 2440 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.

Table C-11: Fatal Error Messages

A Fatal Error is issued when something completely unexpected happens inside the 2440. 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
GPIB Concepts

GPIB Concepts

The IEEE 488-1978 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 488-1978 specifies a standard connector and cable for linking instruments to ensure that GPIB instruments are pin-compatible. The GPIB connector (Fig. 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.

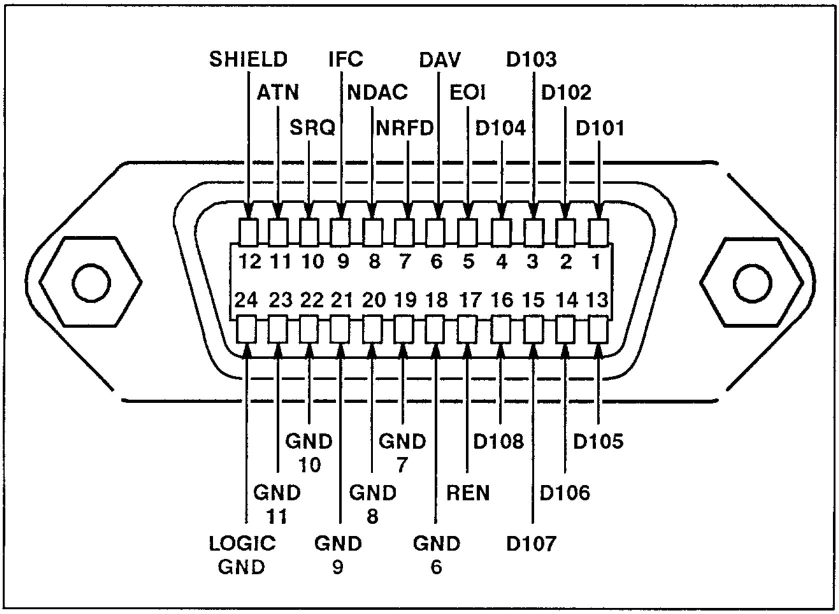


Figure D-1: GPIB connector.

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" logic state corresponds to voltages $\geq$ 2.0 volts and $\leq$ 5.2 volts (HI state). A "1" logic state corresponds to voltages $\geq$ 0 volts and $\leq$ 0.8 volts (LO state).

Messages can be sent either as active-true/passive-false or active-false/passive-true. Active levels must override passive levels wherever any conflict arises (i.e., you can't have active-true and active-false for the same message on the same line). Passive signal levels, whether true or false, correspond to the HI state, and must be carried on a signal line that uses open collector devices.

Functional Elements

Interface functions facilitate sending, processing, and receiving messages for those instruments in which those functions are implemented. The following paragraphs describes the ten different interface functions defined by the IEEE 488-1978. Those not implemented in the 2440 are indicated. The abbreviations for these functions, which are in parenthesis, are taken from the IEEE 488-1978 standard and are commonly used when describing the functions.

Acceptor Handshake (AH)

The AH function works with the SH function to guarantee the listening instrument properly receives data sent by the talking instrument. 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 with the AH function to guarantee the listening instrument properly receives data sent by the talking instrument. 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 allow an instrument 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 2440 implements only the 1-byte address.

Talker (T) and Talker Extended (TE)

The T and TE functions allow an instrument 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 2440 implements only the 1-byte address.

Device Clear (DC)

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

Device Trigger (DT)

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

Remote/Local (RL)

The RL function allows the system controller to select between either the front-panel switches (local) of an instrument or its GPIB interface (remote) as information source. Invoking the function tells the instrument which source the controller wants used.

Service Request (SR)

The SR function allows an instrument to request service from the controller in charge of the interface.

Parallel Poll (PP)

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

Controller (C)

The C function allows an instrument 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. In other words, for an instrument in which the function is implemented, the controller presently controlling the system can invoke the C function to relinquish control to that instrument. The C function is not implemented in the 2440.

Instrument Addresses

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

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. An instrument's listen address is determined by adding 32 to its primary address. For example, an instrument with primary address 19 has a listen address of 19 + 32 = 51.

When the ATN line is asserted and an instrument senses its listen address on the bus, that instrument prepares to receive data sent over the bus. Then when ATN is unasserted, the instrument receives data. 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. 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.

When an instrument senses its talk address on the bus and ATN is asserted, the instrument prepares to send data over the bus. Then when ATN is 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 at the time ATN is asserted, the first instrument automatically untalks itself.

Secondary Address

Some instruments support a special addressing scheme called secondary addressing. Secondary addresses range from 96 to 126. Not all IEEE 488-1978 compatible instruments support secondary addressing. The 2440 does not implement secondary addressing.

GPIB Buses

The 16 GPIB signal lines can be divided into three buses (Fig. 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.

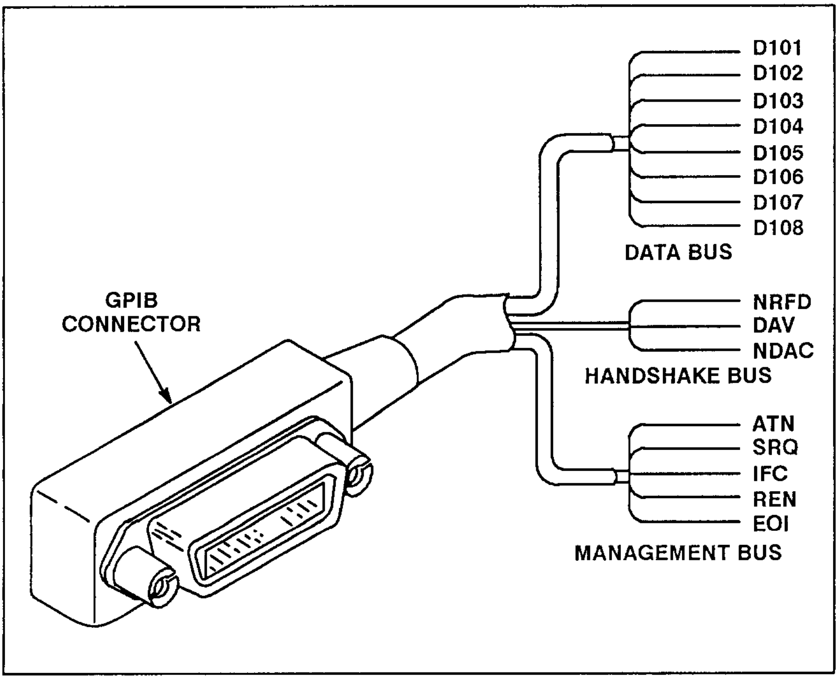


Figure D-2: GPIB Buses.

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. This is the same as sending SPD which is described on page D-14. 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 2440 asserts SRQ whenever there is a change in its status to report to the controller. SRQ's 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 2440 uses the NRFD line to hold off new data when its input buffer is full. As characters are removed from the buffer, the 2440 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, as long as these functions are implemented in the instrument. 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, but controller-in-charge status can. Once an instrument is the system controller, no other instrument on the bus can assume that role unless you reconfigure 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 2440 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 488-1978 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 on universal commands 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 all instruments on the bus that have a DC1 or DC2 subset of the DC interface function. The DCL message reinitializes communication between the 2440 and the controller. To send the DCL command, the controller places the value 20 on the data lines with ATN asserted.

The 2440 responds to DCL by clearing 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.

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 send the LLO command, the controller places the value 17 on the data lines with ATN asserted.

To lock out the 2440 front panel, LLO must be sent with REN asserted. To keep the 2440 front panel locked out, REN must remain asserted. To unlock the 2440 front panel, unassert REN.

Serial Poll Disable (SPD)

The SPD command takes all instruments on the bus out of 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 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 2440 implements GET. See the DT command on page A-28 for more information on what the 2440 does 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.

To use GTL with the 2440, the 2440 must first be returned to the local state by unasserting the REN line. Sending the GTL command, without unasserting REN, will not unlock the 2440 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 2440 cannot become a controller; therefore, it does not understand TCT.

Serial Polling

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

Status Bytes

Each instrument on the GPIB that incorporates 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 2440 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. There is, however, 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 respond to instruments asserting SRQ's by doing a serial poll. If so programmed, it generates an interrupt and serially polls each instrument on the bus to determine which instrument is requesting service. 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.

E
INIT Settings and Power Up States

Init Settings and Power Up States

The INIT command of the 2440 is used to return the oscilloscope to a known operating state. Initializing the 2440 provides a basic setup from which to begin programming the 2440. See the INIT command description on page A-29 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 2440. 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 their reset status, refer to the INIT command in Appendix A. The AUTOSetup command also changes the front-panel setup; see Automatic Feature Commands in Appendix A of this guide and section 5 and Appendix B of the Operators Manual for more information.

When the 2440 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
ASCII and Character Charts

ASCII and Character Charts

This section contains two charts. The first chart shows the character set the 2440 displays. The second chart shows ASCII symbols and their GPIB equivalents.

For the 2440 character chart, each larger character that's shown in the center of each box is a character that can be displayed on the 2440 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 ASCII code chart is the same as the 2440 character chart, except the large center characters are ASCII characters rather than 2440 characters.

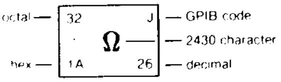
The GPIB equivalent code designates what character string is sent by the 2440 to designate the 2440 display character. Notice that many of the 2440 characters in the 2440 character chart have no equivalent ASCII characters in the ASCII chart. Any character which does not have a corresponding standard ASCII character cannot be displayed on conventional terminals. These nonconventional 2440 characters are, however, given a GPIB equivalent code so they can be sent to and read from the 2440. Any character in the 2440 chart that does have a corresponding ASCII character in the ASCII chart does not have a GPIB equivalent code given, but it can be obtained from the ASCII chart.

Let's look at an example of sending a message consisting of all ASCII characters and one that contains 2440-unique characters. Assume you want to display the string "HELLO" on line 10 of the 2440. Send the command MESSAGE 10:"HELLO". Now assume you want to send the same message only have "HELLO" underlined but these underlined characters have no ASCII equivalents. Send the command MESSAGE 10:"\h\e\l\l\o" to the 2440. Note the \h, \e, etc. are GPIB equivalents of 2440-unique characters. Remember to use the GPIB equivalent code if it appears in the 2440 characters chart.

B7 B6 B5 BITS B4 B3 B2 B1	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
	SPECIAL		NUMBERS SYMBOLS		UPPER CASE		UNDERLINED	
Ø Ø Ø Ø	0 0 — 0 0	20 @ Ø̲ 10 16	40 SP 20 32	60 Ø 30 48	100 @ 40 64	120 P 50 80	140 ` @̲ 60 96	160 p P̲ 70 112
Ø Ø Ø 1	1 1 ⊗ 1 1	21 A 1̲ 11 17	41 ! 2 21 33	61 1 31 49	101 A 41 65	121 Q 51 81	141 a A̲ 61 97	161 q Q̲ 71 113
Ø Ø 1 Ø	2 2 ▯ 2 2	22 B 2̲ 12 18	42 " 22 34	62 2 32 50	102 B 42 66	122 R 52 82	142 b B̲ 62 98	162 r R̲ 72 114
Ø Ø 1 1	3 3 d̲ 3 3	23 C 3̲ 13 19	43 # d 23 35	63 3 33 51	103 C 43 67	123 S 53 83	143 c C̲ 63 99	163 s S̲ 73 115
Ø 1 Ø Ø	4 4 k 4 4	24 D 4̲ 14 20	44 $ (blank) 24 36	64 4 34 52	104 D 44 68	124 T 54 84	144 d D̲ 64 100	164 t T̲ 74 116
Ø 1 Ø 1	5 5 %̲ 5 5	25 E 5̲ 15 21	45 % 25 37	65 5 35 53	105 E 45 69	125 U 55 85	145 e E̲ 65 101	165 u U̲ 75 117
Ø 1 1 Ø	6 6 z̲ 6 6	26 F 6̲ 16 22	46 & z 26 38	66 6 36 54	106 F 46 70	126 V 56 86	146 f F̲ 66 102	166 v V̲ 76 118
Ø 1 1 1	7 7 s 7 7	27 G 7̲ 17 23	47 ' 27 39	67 7 37 55	107 G 47 71	127 W 57 87	147 g G̲ 67 103	167 w W̲ 77 119
1 Ø Ø Ø	10 8 Δ̲ 8 8	30 H 8̲ 18 24	50 (Δ 28 40	70 8 38 56	110 H 48 72	130 X 58 88	150 h H̲ 68 104	170 x X̲ 78 120
1 Ø Ø 1	11 9 ÷̲ 9 9	31 I 9̲ 19 25	51) ÷ 29 41	71 9 39 57	111 I 49 73	131 Y 59 89	151 i I̲ 69 105	171 y Y̲ 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 J̲ 6A 106	172 z Z̲ 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 [ʃ 5B 91	153 k K̲ 6B 107	173 { ʃ̲ 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 L̲ 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] ʅ 5D 93	155 m M̲ 6D 109	175 } ʅ̲ 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 N̲ 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 O̲ 6F 111	177 DEL* H O 7F 127

DEL = RUBOUT

KEY TO CHART



(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

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

TEKTRONIX STD 062-5435-00 4 SEP 80

6338-04

G
Answers to Common Questions

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) 835-9433 (TEK-WIDE).

How Do I Unlock the Front Panel?

If the response to a LOCK? query is LLO, you need to strobe the remote enable line (REN) from HI to LO. Alternately, you can send the command LOCK OFF to the 2440. Powering the 2440 off, then back on, also unlocks the front panel. Be aware that the 2440 can lock its front panel while doing a self calibration, an Auto Setup, and during certain portions of setup sequences.

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

It means the 2440 was talking with nothing to say. The most common cause is sending a query that is mis-typed and then asking for a response from the 2440. Because the 2440 didn't understand the query, it can't send an answer. But instead of hanging the bus, it sends FFh. See Appendix H for more information.

How Do I Get Just the Answer From a 2440 Query?

Send the command PATH OFF to the 2440. 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 page 3-4 for more information.

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

By sending the command INIT SRQ to the 2440. This is usually done before waiting for the 2440 to complete a task. From the front panel, changing the GPIB address of the 2440 also clears all SRQ's and event codes.

Can I Run Self-Cal From a Controller?

Yes. Send the command TESTTYPE SELFCAL;EXECUTE to the 2440. See the Calibration and Diagnostics Commands on page A-14 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 2440. If no arguments follow the header, all measurement results will be returned. The VALUE? query, with no argument, is the GPIB's equivalent of the Snapshot. Note that if PATH is OFF and you request all the measurements, the 2440 returns the all the results but not the measurement types. Make sure PATH is ON, so you get the measurement types also.

How Do I Clear the Screen?

To clear all 16 lines of text send the MESSAGE CLRSTATE command to the 2440. Use the MENUOFF command to clear just the bottom 3 lines of text.

How Do I Clear One Line of Text?

Send the command MESSAGE line#:"" to the 2440. The line# field in this command indicates the number for the 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 2440. 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 2440. 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 2440 will send an SRQ. The corresponding event code will indicate which menu button was pushed. See How to Use Service Requests on page 3-5 for more information. The Pulse Counting program example in Appendix J 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 on page 3-5 for more information.

How Do I Send a Sequence From One 2440 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 page 2-1 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 on page 3-26 for more information.

Why Does the Acquisition Restart When I Send Certain Commands?

Acquisitions restart because the 2440 needs to clean up when certain modes change. The interactions between various modes are numerous, and often the only solution is to restart the acquisition between each mode change. Any commands changing the Volts/Div, Sec/Div, and/or trigger system settings restart the acquisition. The MENUOFF command also causes a restart because that command is used to exit from calibration and diagnostics operations and all of the hardware registers in the 2440 are reset to a normal state before leaving.

Why Doesn't the Waveform I Sent to the 2440 Look Like the One I Took Out?

You probably took the waveform out with different waveform encoding than the 2440 was expecting when you sent it back. This mismatch can happen when the format for encoding binary data is changed in the 2440 between transfers. See the subsections of Appendix B that deal with waveform data transfers and encoding for more information.

Why are the Abbreviations of Symbols on the 2440 Not the Same as Other Tektronix Oscilloscopes?

The sets of commands each instrument executes can differ from other instruments. The abbreviations for a command set are chosen to keep each command unique within its own command set, and the number of letters needed to keep a command unique varies with the command set. For instance in the 2440, DLYEvts and DLYTime require the first four letters (DLYE and DLYT, respectively) be used as the abbreviations to differentiate between the two command headers. If an instrument had only on type of delay, DLY or even DL might be long enough to make the command header unique from others in its command set. The point is that differing command sets may have different abbreviations. 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 set the 2440 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 2440 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 2440 always sends the entire response back to your controller. Since the SET? query response is very long (approximately 2500 bytes), your controller's message timeout value may be set too short. Short timeout values can also cause a problem when using DEBUG mode in either SLOW or PAUSE. The controller timeouts and stops listening, but all you 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. Since the 2440 processes the message in the order it's received, it responds to the dummy query immediately after it has completed the command. After sending the command, put some type of input statement in your program so it waits for the 2440 to respond to the query. When you get that response, you know the command is done. This method is suggested by the IEEE in the new release of GPIB standards.

Let's consider an example of this method. If you send the command string CH1 VOLTS:5;CURSOR TPOS:ONE:50;EVENT? to the 2440, you know that the 2440 has executed the commands when the response to the EVENT? query is returned. You also know the command string executed correctly if the event number returned is 0.

H
2400 Specific Information

Instrument Specific Information

Talked With Nothing to Say

The 2440 always says something when it is made a talker. If it has nothing to say, it sends a byte of all "1's" (ASCII rubout) with the EOI signal. This byte tells the listening device that no data is coming and prevents tying up the GPIB while one device waits for the 2440 to talk.

This ASCII rubout character is also sent if the 2440 is sent an incorrect query (mistyped, etc.) and then talk addressed to read the response. Since the 2440 didn't understand the query, it cannot respond with data when talked.

You may also receive the ASCII rubout character if the 2440 has been sent a query that returns a string of variable length. Very long strings (such as those that might be returned for SET? and CURVE? queries) may not be handled well by your controller. If the controller handles long strings by breaking them down into a series of smaller substrings as they come in, it still might have to finish filling some of these substrings even after the 2440 has sent all of the data in the original string. In this case, the 2440 will send the ASCII rubout character for the remaining strings.

Accepting Numbers

The 2440 always sends numbers in the number format specified in the command tables for that particular command. However, it will accept any number format the controller sends. If the 2440 receives a number whose precision is greater than the instrument can handle internally, the number will be rounded to enhance accuracy.

Accepting Characters

The 2440 accepts both uppercase and lower case alpha characters. If you type commands in lowercase letters, the 2440 converts them to uppercase before evaluating them ("a" becomes "A", "b" becomes "B", etc.). If you use DEBUG mode to see the characters being sent to the 2440, the uppercase letters are shown since the 2440 converts any lowercase letters to uppercase as they come in.

The only exception to the lowercase-to-uppercase conversion rule is when you use the MESSAGE command to create your own on-screen messages or menus. An uppercase or lowercase letter that is immediately preceded by a backslash is a GPIB code equivalent of a 2440-unique character (see Appendix F), and the 2440 converts it to the appropriate 2440 character ("\a" becomes "A", "\J" becomes "W", etc.). Only the converted character (the 2440 character) will be displayed in the menu that's created using MESSAGE. DEBUG mode will show both the converted character and the preceding backslash.

Command and argument names can be abbreviated to the least 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 2440 have there own minimum string abbreviations needed to make each of there commands unique. Some of their abbreviations may conflict with 2440 abbreviations. Also, 2440's with different versions of firmware can have different abbreviations for the same symbol. This will happen when a new release introduces a new command that requires lengthening an existing command's abbreviation to prevent symbol conflict. To avoid both compatibility problems, use full symbols when programming the 2440.

The time it takes to send the extra characters the abbreviation saves is short relative to that taken to interpret and execute the command.

Power-Up Sequence of the 2440

When the 2440 first powers up, it performs the following internal checks before it can be used:

BASIC HARDWARE CHECKS–The ROM'S 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 2440 goes on to the next step.

ACQUISITION HARDWARE CHECKS—At this stage the CCD'S, 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 2440 has just powered up. This SRQ is sent only if RQS is ON. See Service Requests on page A-42 for more information.

I
How to Use Fast Transmit Mode

How to Use Fast Transmit Mode

The Fast Transmit (FASTXMIT) mode is used when it's needed to upload waveforms from the 2440 as fast as the scope can output them. This mode lets you acquire and upload a series of successive acquisitions of a waveform, without sending a CURVE? or WAV? query for each waveform. The 2440 will acquire a waveform, transfer it, and rearm itself for acquiring the next waveform. It does this "acquire-transfer-rearm" sequence at a very high rate (about 20 to 50 times/second; see Fast Transmit Transfer Rates on page I-8), which results in a greatly reduced transfer time for the series of waveforms.

Increasing the transfer rate using FASTXMIT results in less dead time between acquisitions. The user-specified number of acquisitions is output without the need for a corresponding number of consecutive "listen 2440-send command-talk 2440-input waveform" exchanges between 2440 and controller. Instead, the 2440 outputs the specified number of acquisitions as a single string when it is talked by the controller. The result is that the time between waveforms is limited only by the 2440 transfer rate including the time it takes to make the acquisition plus the speed at which the controller can handle the data. If the triggers occur at a rate less that the waveform transfer rate (such as can be true for video waveforms), Fast Transmit can provide an acquisition for every trigger.

How to Set Up Fast Transmit

To set the 2440 up for Fast Transmit mode, you need to determine which channel(s) contain the waveforms you want to upload and the number of waveforms you want to upload. You also must determine which data encoding format (RIBINARY, RIPARTIAL, RPBINARY, RPPARTIAL) you want the data in when sent. You send this information to the 2440 using the FASTXMIT command.

In our example on page I-3, we send the command FASTXMIT 30, NORMAL:CH1,ENCDG:RIBINARY to the 2440. This command tells the 2440 to send 30 acquisitions of the CH1 waveform to the controller. The ENCDG:BINARY argument says to encode the waveform data using RIBINARY format. The NORMAL argument indicates that the 2440 is to send non-delta delayed waveforms.

The argument CH1 can be changed to CH2 if that is the desired waveform source, or BOTH, if waveforms from both CH1 and CH2 are desired. If both CH1 and CH2 waveforms are requested, the number specified refers to the number per channel. In other words, FASTXMIT 30, NORMAL:BOTH, would transfer 60 waveforms, starting with CH1 and alternating between CH1 and CH2.

If NORMAL is replaced with DELTA, the 2440 sends delta delay waveform(s) for the specified signal source(s). Regardless of what waveform is displayed on the screen, the 2440 acquires both a B Delayed and Delta Delay waveform if the Horizontal mode is set to B. If the scope is set to B mode, using NORMAL with FASTXMIT uploads the B Delayed waveform; using DELTA uploads the Delta Delayed waveform. Requesting DELTA waveforms when the Horizontal mode is not set to B mode hangs up the controller program (see Potential Traps on page I-6).

After sending the command string as explained, you must tell the 2440 to begin sending Fast Transmit waveforms. Address the 2440 to talk; the transition from untalked to addressed to talk starts the transfer. The Fast Transmit sequence will only begin when the controller issues the talk address for the 2440 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.

This program was written using the Tektronix DSO Development Libraries included in the DSO Development Software package. It illustrates how to use the Fastxmit commands to rapidly acquire num_waves waveforms to ram. The 2440's address must be set to 1.

NOTE

This program must be compiled with the compact memory model and linked to the compact memory library. If there is an error in execution and the program exits prematurely, you must reset the oscilloscope.

```
#include <stdio.h>
#include <process.h>
#include <malloc.h>
#include <sys\timeb.h>
#include <gpib.h>

#define UNT 0x5f                /* command to 'Un-talk' scope */
#define NUM 100                 /* number of waveforms to transfer */

int SCOPE;                      /* scope address */
int sent;                       /* data bytes received count */
struct timeb t0, t1;            /* times */
char *resp[NUM];                /* pointers to waveforms in mem */
int i, j;                       /* index counters */
char untalk[1] = { UNT };       /* untalk bus message */
float diff_time,                /* total time to transfer NUM
                                   waveforms */
      asec;                     /* seconds / division variable */
long t1milli;                   /* ending time in milliseconds */
long t0milli;                   /* starting time in milliseconds */
int num_waves;

void cdecl main( void );        /* main prototype */

void main(){

  /*
   * Get address of oscilloscope
   */
  settimo( 200L );             /* make timeout 200ms for search */
  finddev("id?", "TEK/24", 0, &SCOPE);  /* search for a 2430 */

  /*
   * If value of scope is less than 0 then no scope was found.
   */
  if( SCOPE < 0 ){
    if( busstat( GPIB0 ) & HWERR )
      fprintf(stderr, "Bad interface board or setup\n\007");
    else
      fprintf(stderr, "No 24xx online!\007\n");
    exit(1);
  }

  settimo( 30000L );          /* change timeout to 30 sec */

  /*
   * Initialize the number of waveforms to transfer.
   */
  num_waves = NUM;
```

```
/*
 * Allocate space for waveform storage.
 */
for(i=0;i<num_waves;i++){
  if(( resp[i] = (char *)malloc((size_t) 1040)) == NULL){
    printf("out of memory");
    exit(0);
  }
}

/*
 * Get seconds / division setting to calculate the time
 * between acquisitions.
 */
gputs("path off;hor? asec",SCOPE);
gscanf(SCOPE, "%f", &asec);

/*
 * Setup scope for fast transmit, exit program if error.
 */
if( gprintf(SCOPE, "path on;fastxmit 1,normal:ch1,encd:ribi")
< 0 )
  exit(0);

/*
 * Delay program to allow scope to setup for fast transmit.
 */
for(i=0;i<10000;i++);
printf("Acquiring %d waveforms....\n", num_waves);
ftime(&t0);             /* get start time */

for(i=0;i<num_waves;i++){
  /*
   * Send an untalk command to scope to begin the fast-
   * transmit process.  NOTE:  If GPIB error occurs during
   * fast transmit, the oscilloscope power must be cycled
   * and the program restarted.
   */
  sendcmd( GPIB0, untalk, 1 );
  if ((sent = receive(SCOPE, resp[i], 1040)) < 0){
    printf("ERROR:  Unable to transfer waveform data\n");
    gprintf(SCOPE, "fastxmit off");
    for(j=0;j<10000;j++);
    exit(0);
  }
}
ftime(&t1);             /* get stop time */
```

```
/***************************************************************/
/* At this point there are num_waves waveform stored in memory.*/
/* You can now process these waveforms and store them to disk. */
/***************************************************************/

/*
 * CLEANUP
 */
  if( gprintf(SCOPE, "fastxmit off") < 0 )
  exit(0);
  t1milli = t1.time*1000 + t1.millitm;  /* end time in millisec-
onds */
  t0milli = t0.time*1000 + t0.millitm;  /* start time in milli-
seconds */
  diff_time = (float)(t1milli - t0milli)/1000; /* differential
time in seconds */

/************** PRINT ON SCREEN TRANSFER RATE ************/
  printf("transfer rate = %.2f waveforms/sec\n", num_waves/
diff_time );

  printf("%d waveforms were xfered in %5.3f sec\n",num_waves,
diff_time );
  printf("time between acqs = %.2e sec\n",
       (diff_time - asec * 20 * num_waves)/num_waves);
  for(i=0;i<10000;i++);         /* delay */
}
```

Potential Traps

For instruments with Firmware Version 2.40 and above:

The Fast Transmit mode of the 2440 is optimized for transfer speed. To maximize speed, Interface Clear messages are ignored during a Fast Transmit. The only way to abort a transmission and regain control over the 2440 during a fast transmit is to turn the scope off and then turn it back on or issue a Device Clear.

For instruments with Firmware Version 2.40 and below:

The Fast Transmit mode of the 2440 is optimized for transfer speed. To maximize speed, Device Clear and Interface Clear messages are ignored during a Fast Transmit. The only way to abort a transmission and regain control over the 2440 during a fast transmit is to turn the scope off and then turn it back on.

Now, let's look at the two most common problems which occur when using Fast Transmit: either the controller program hangs or the 2440 hangs.

Controller Program Hangs

The controller program will hang if the 2440 does not transmit a waveform within the timeout period specified in the controller. Here are a few reasons why the 2440 might time out:

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 2440 is OFF and the 2440 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 2440, but the 2440 has been triggered less than X times.
4. You have not set triggering coupling, mode, or source so the specified number of waveform acquisitions can be triggered.
5. You have not dimensioned your data array(s) large enough to hold the requested number of waveforms. The array must hold the number of waveforms × the bytes/waveform. See Binary Formats in Appendix B for information on determining byte length for waveforms.

2440 Hangs

This condition is most often indicated by a completely blank 2440 screen. The problem is that the 2440 has been talked, but the controller has not listened to the complete Fast Transmit waveform. The 2440 is now waiting trying to send the rest of the waveform. Here are some cases where 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 2440. This delay is necessary to allow the 2440 to interpret the command. It is needed when sending any command to the 2440 while Fast Transmit is active, including those that change any Fast Transmit parameters (i.e., FASTXMIT NORMAL:CH2) or any oscilloscope setup (i.e. CH1 VOLTS:2).

In either of these two cases, when the controller sends a normal query to the 2440 and talks it to get the response, the 2440 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 2440 is done talking.

3. You asked for X number of waveforms from the 2440 but read less than X waveforms in.

Fast Transmit Transfer Rates

Now we look at some some transfer rates for Fast Transmit. The rates were obtained from a very fast controller so that the only significant speed limiting factor was the 2440.

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 and the time it takes to send that waveform to the controller. All the waveform displays are turned off to speed up the transfer rate.

You will note that there are different waveform transfer rates given for acquisitions at 100 µs from those at 50 µs to 100 ns. Because waveforms acquired at sweep speeds faster than 100 µs require special processing time, Fast Transmit for waveforms acquired at 50 µs to 100 ns have a slower transfer rate (waveforms/second) than those acquired at 100 µs. Also, although the rates are not listed, waveforms acquired at rates slower than 100 µs have slower transfer rates as the time to acquire the waveform increases. For example, at 100 µs, the 2 ms required to acquire a waveform is only adding about 10% to the total transfer time; at 1 ms, the 20 ms per waveform acquisition approximately doubles the transfer time; at 10 ms, the 0.2 seconds per waveform acquisition takes up 90% of the transfer time.

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.

Table I-1: Transfer Rates

Acquisition Rate	Acquisition Mode	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 2440 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 2440 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
Example Programs

Example Programs

This appendix contains sample programs to use as guidelines in programming the 2440.

Talker-Listener Program for HP Basic

This HP Basic program allows you to send commands or queries to the 2440 and prints responses from the 2440.

```
10 !===========================
20 ! first enter the 2440´s address and then open the hardware
30 ! port to talk to the GPIB. Finally enable the srq interrupt
40 ! and point to the handler routine.
50 !===========================
60  print "Enter 2440 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 has a "?"
170 ! right after the header. If it is a query, then talk the
180 ! 2440 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 "2440 response is: "&answr$
260 goto 200
270 !=========================
280 ! This is the code that is executed when an srq happens. We
290 ! poll the bus and print the status bye and the corresponding
300 ! 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
```

Pulse Counting Program

This program is written using Microsoft C and the National GPIB Drivers. It will count the number of pulses between time cursors. It demonstrates the use of waveform data commands and queries as well as the way custom text is written to the screen of the 2440.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern  int ibfind(char *name);
extern  int ibpad(int bd,int v);
extern  int ibrd(int bd,char *rd,int cnt);
extern  int ibwrt(int bd,char *wrt,int cnt);
extern  int ibsic(int bd);

void main(){

   short bd;
   int cnt;
   char msg[50];
   char msg2[50];
   char mids[10];
   char maxs[10];
   char mins[10];
   char start[10];
   char *p;
   int evnt,max,min,mid,level,pcross,point1,point2;
   bd=ibfind("GPIB0");

   if(bd<0){
      printf("Board name GPIB0 not found.\n");
      printf("Check DEVICE=GPIB.COM in your CONFIG.SYS file.\n");
      exit(0);
   }
```

```
  ibsic (bd);
  bd=ibfind("DEV1");
  ibpad (bd,1);         /* Set scope address to 1 */

/*****************************************************************
*  Turn cursors on and prompt user to set them for measurement
*  and wait for menu button selection.
*****************************************************************/
  strcpy (msg,"CURSOR FUNCTION:TIME,MODE:DELTA");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE CLRSTATE;MENUOFF;RQS OFF;INIT SRQ");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 10:\"MOVE THE TIME CURSORS TO BRACK-
ET\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 9:\"THE PART OF THE WAVEFORM YOU
WANT\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 8:\"TO COUNT THE PULSES IN\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 6:\"PRESS ANY MENU BUTTON WHEN READY\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"PATH OFF;EVENT?");    /* Check for button push */

  do{
    ibwrt (bd,msg,strlen(msg));
    ibrd  (bd,msg2,10);
    evnt = atoi(msg2);
  }while (evnt < 450 || evnt > 454);

/*****************************************************************
*  The area of interest is now bracketed by the time cursors.
*  Set START and STOP to these values with the SNAP command.
*****************************************************************/
  strcpy (msg,"SNAP");
  ibwrt (bd,msg,strlen(msg));

/*****************************************************************
*  A pulse will be defined as having a positive edge that has at
*  least 50 percent amplitude.  Now find the amplitude of the
*  signal and the 50 percent point.
*****************************************************************/
  strcpy (msg,"DATA SOURCE:CH1;MAXIMUM?");
  ibwrt (bd,msg,strlen(msg));
  ibrd  (bd,maxs,10);
  max = atoi(maxs);
  strcpy (msg,"MINIMUM?");
  ibwrt (bd,msg,strlen(msg));
  ibrd  (bd,mins,10);
  min = atoi(mins);
  mid = min+(max - min)/2;
  p = itoa(mid,mids,10);

/****************************************************************
* Set the crossing level to the calculated 50 percent point. Set
* the HYSTERESIS up to filter noisy signals and set the search
* DIRECTION from left to right.
****************************************************************/
  strcpy (msg,"LEVEL ");
  strcat (msg,mids);
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"HYSTERESIS 20;DIRECTION PLUS;START?");
  ibwrt (bd,msg,strlen(msg));
  ibrd  (bd,msg2,10);
  point1 = atoi(msg2);

/****************************************************************
* Now count the actual pulses. This consists of looking for
* positive crossings of the 50 persent level. If one is found
* the window is adjusted and the count is incremented.
****************************************************************/
  cnt = 0;
  do{
     strcpy (msg,"PCROSS?");
     ibwrt (bd,msg,strlen(msg));
     ibrd  (bd,msg2,10);
     point2 = atoi(msg2);
     if (point2 > point1){
        cnt++;
        point2++;
        p = itoa(point2,start,10);
        strcpy (msg,"START ");
        strcat (msg,start);
        ibwrt (bd,msg,strlen(msg));
     }
  }while(point2 > point1 && point1 != 0);

  printf ("There are %d pulses inside the cursors.\n",cnt);
}
```

Extended Accuracy Timing Measurement Program

This program is written using Microsoft C and the National GPIB Drivers. It makes extended accuracy timing measurements. It uses the time cursorsand pcross function for the time readout and horizontal expansion for increased accuracy in locating the measurement points. The basic algorithm sets the time cursors at the 50 percent points of the waveform. It then expands the waveform 100 times and repositions the time cursors more accurately.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern  int ibfind(char *name);
extern  int ibpad(int bd,int v);
extern  int ibrd(int bd,char *rd,int cnt);
extern  int ibwrt(int bd,char *wrt,int cnt);
extern  int ibsic(int bd);

double cdecl fudge_pos ( int bd );

void main(){

   short bd;
   char msg[100];
   char msg2[100];
   char pt1[20];
   char pt2[20];
   char sec[20];
   char value[20];
   char unit[20];
   int evnt,pcross,point1,point2;
   double secdiv,exsec,cursor;

   bd=ibfind("GPIB0");

   if(bd<0){
      printf("Board name GPIB0 not found.\n");
      printf("Check DEVICE=GPIB.COM in your CONFIG.SYS file.\n");
      exit(0);
   }

   ibsic (bd);
   bd=ibfind("DEV1");
   ibpad (bd,1);        /* Set scope address to 1 */
```

```
/****************************************************************
  Set up the sec/div, start, and stop, and then start the scope
  acquiring. Wait for a single sequence to occur. Path is turned
  off so we can just deal with numbers on query responses.  All
  the time measurements will be done from level 0 or center
  screen.
****************************************************************/
  strcpy (msg,"MESSAGE CLRSTATE;MENUOFF;RQS OFF;INIT SRQ;PATH
OFF");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"HORIZONTAL ASECDIV:100E-6;DATA SOURCE:CH1");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 9:\"        POSITION WAVEFORM TO CENTER
\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 8:\"            SCREEN VERTICALY\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"MESSAGE 2:\"    PRESS ANY MENU BUTTON WHEN
READY\"");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"EVENT?");

  do{
    ibwrt (bd,msg,strlen(msg));
    ibrd  (bd,msg2,10);
    evnt = atoi(msg2);
  }while (evnt < 450 || evnt > 454); /* Check for button push */

  strcpy (msg,"MESSAGE CLRSTATE;ATRIG MODE:SGL;RUN ACQUIRE");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"EVENT?");    /* CHECK FOR ACQ COMPLETE */

  do{
    ibwrt (bd,msg,strlen(msg));
    ibrd  (bd,msg2,10);
    evnt = atoi(msg2);
  }while (evnt != 461);

  strcpy (msg,"START 1;STOP 1024;LEVEL 0");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"HYSTERESIS 5;DIRECTION PLUS");
  ibwrt (bd,msg,strlen(msg));
```

```
/****************************************************************
  Now get the first PCROSS value. This will be the edge we start
  measuring from. If your edge is different substitute a
  different search algorithm here. The START value is moved to
  be one point before the crossing of interest. Then we go
  looking for the next crossing point, so we can set STOP there.
****************************************************************/
  strcpy (msg,"PCROSS?");
  ibwrt (bd,msg,strlen(msg));
  memset (pt1,'\0',20);
  ibrd  (bd,pt1,10);
  point1 = atoi(pt1);
  strcpy (msg,"START ");
  sprintf(msg,"%s%d",msg,point1+1);
  ibwrt (bd,msg,strlen(msg));
  point1--;
  strcpy (msg,"PCROSS?");
  ibwrt (bd,msg,strlen(msg));
  memset (pt2,'\0',20);
  ibrd  (bd,pt2,10);
  point2 = atoi(pt2);
  point2 ++;

/****************************************************************
  Now set the time cusors to the points that we have just found.
  These will be used to give us the actual time values we want.
  Then expand the horizontal by 100X. The cursors will stay at
  the points that we placed them even after the expansion.
****************************************************************/
  strcpy (msg,"cursor function:time,units:time:base");
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"CURSOR TPOS:ONE:");
  sprintf(msg,"%s%d",msg,point1);
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"CURSOR TPOS:TWO:");
  sprintf(msg,"%s%d",msg,point2);
  ibwrt (bd,msg,strlen(msg));
  strcpy (msg,"HORIZONTAL? ASECDIV");
  ibwrt (bd,msg,strlen(msg));
  memset (sec,'\0',20);
  ibrd  (bd,sec,20);
  secdiv = atof(sec);
  exsec = secdiv/100;
  strcpy (msg,"HORIZONTAL ASECDIV:");
  sprintf(msg,"%s%f",msg,exsec);
  ibwrt (bd,msg,strlen(msg));
```

```
/*****************************************************************
   Now that we've expanded everything, we need to go in and find
   the new PCROSS value. Next, we move the time cursors to that
   point and look for the ending point and set the other time
   cursor there. The difference between the time cursors is the
   answer. A slight problem comes in here because horizontal
   position is altered slightly in doing the expansions so that
   points line up correctly. This needs to be compensated for and
   that is what the function FUDGE_POS is doing.
*****************************************************************/
   strcpy (msg,"CURSOR SELECT:ONE");
   ibwrt (bd,msg,strlen(msg));
   cursor = fudge_pos(bd);
   strcpy (msg,"CURSOR TPOS:ONE:");
   sprintf(msg,"%s%f",msg,cursor);
   ibwrt (bd,msg,strlen(msg));
   strcpy (msg,"CURSOR SELECT:TWO");
   ibwrt (bd,msg,strlen(msg));
   cursor = fudge_pos(bd);
   strcpy (msg,"CURSOR TPOS:TWO:");
   sprintf(msg,"%s%f",msg,cursor);
   ibwrt (bd,msg,strlen(msg));
   strcpy (msg,"CURSOR? DISPLAY:UNIT");
   ibwrt (bd,msg,strlen(msg));
   memset (unit,'\0',20);
   ibrd  (bd,unit,20);
   strcpy (msg,"CURSOR? DISPLAY:VALUE");
   ibwrt (bd,msg,strlen(msg));
   memset (value,'\0',20);
   ibrd  (bd,value,20);
   printf("The time difference is %s%s\n",value,unit);}

/*****************************************************************
   This function finds the first PCROSS and then fiddles with the
   horizontal position before setting the cursor value to a
   "corrected" value which is returned to the variable cursor.
*****************************************************************/
double fudge_pos(bd){
   double hpoint1,cursor,exhor,refpos,pospick;
   int point1;
   char msg[100];
   char pt1[20];
   char hpt1[100];
   strcpy (msg,"START 1;PCROSS?");
   ibwrt (bd,msg,strlen(msg));
   ibrd  (bd,pt1,10);
   point1 = atoi(pt1);
   point1--;
   strcpy (msg,"HORIZONTAL? POSITION");
```

```
  ibwrt (bd,msg,strlen(msg));
  memset (hpt1,'\0',20);
  ibrd  (bd,hpt1,20);
  hpoint1 = atof(hpt1);
  exhor = hpoint1 * 100;
  if (exhor < 256.0)
    refpos = exhor;
  else if (exhor >256.0 && exhor < 102044.0)
  {
    pospick = (exhor - 256)/100;
    refpos = 256+(pospick - (pospick))*100;
  }
  else
    refpos = 1023 - (1023 - hpoint1)*100;
  cursor = hpoint1 + (point1 - refpos)/100;
  return (cursor);
}
```

Index