Programmer Manual

Tektronix

2714 & 2715
Spectrum Analyzer

070-8533-04

This document supports Firmware
Version 2.14.97.

Copyright © Tektronix, Inc. 1992, 1993, 1994. All rights reserved.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supercedes that in all previously published material. Specifications and price change privileges reserved.

Printed in the U.S.A.

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

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

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

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

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

Table of Contents

Introduction

Message Structure

Functional Groups

Command/Query

Status Reporting

Programming

Appendices

Index

List of Figures

List of Tables

Introduction

Introduction to Programming

The Tektronix 2714 or 2715 Spectrum Analyzer allows remote control of its functions with one of two communication port options. Option 08 provides an RS-232 data communications interface; Option 03 provides an IEEE Standard 488.1 General Purpose Interface Bus (GPIB) communications interface.

With a desktop computer and an appropriate control program, you can configure front panel settings (except those intended for local use only, such as INTENSITY) and acquire, transfer, process, and analyze data remotely.

The command set and message structure for the RS-232 and GPIB interfaces are almost identical. However, a few interface-specific considerations, such as communications parameters and protocols, are different. The setup for each interface is described separately in this section.

NOTE*. If your instrument is equipped with the RS-232 interface then continue with the next subsection,* RS-232 Operation (Option 08)*. Otherwise, turn to the* GPIB Operation (Option 03) *subsection and follow the instructions there.*

RS-232 Operation (Option 08)

NOTE*. If your spectrum analyzer is equipped with a GPIB instrument bus, you can skip this subsection.*

The 2714 or 2715 Spectrum Analyzer follows EIA Standard RS-232 when equipped with the RS-232 interface. This standard establishes electrical levels, connector configuration, and signal protocols for communication between two devices called the DCE (data circuit-terminating equipment) and the DTE (data terminal equipment). The 2714 or 2715 implements the DTE end of the interface.

Note that the RS-232 interface is NOT a bus. Only one device can be connected to the instrument's RS-232 interface. Unlike a GPIB interface, RS-232 does not support device addresses or serial polling.

For example, if a computer is connected to the spectrum analyzer's RS-232 interface, a printer or plotter could not be connected to the spectrum analyzer without first disconnecting the computer. To plot screen data directly from the spectrum analyzer, you would first have to disconnect the computer and then connect your printer or plotter.

The 2714 or 2715's RS-232 interface requires a minimum of three signal lines for operation:

- Transmit data (TXD)
- Receive data (RXD)
- Ground (GND)

If hardware handshake is required, additional lines must be supplied in the cable. Refer to *Appendix A: RS–232 Concepts* for cabling diagrams.

The section titled *Selecting a Data Flow Control Method* on page 1–7 describes the use of the additional lines for hardware flow control.

EIA Standard RS-232 defines other lines typically used for modem control and handshaking. The 2714 or 2715 can operate using the minimum wiring configuration. If the appropriate handshake lines are provided, a printer or plotter that expects handshaking over the RS-232 interface may be used.

Data bits are transferred serially, one bit at a time, over the RS-232 interface. Data consists of instrument commands and queries, control settings, parameter values, or display information.

If a computer is connected to the spectrum analyzer by the RS-232 interface, the computer's serial interface (called a COM port if the controller is an MS-DOS computer) must be correctly configured beforehand. Programmed commands and data can then be transmitted over the interface to the instrument.

If a query such as `FREQ?` is transmitted, the spectrum analyzer formats its response immediately and sends it back to the computer. The control program must be ready to receive the incoming data. In the following subsections you will learn how to set up your 2714 or 2715 for RS-232 operation. *Appendix A: RS-232 Concepts* provides additional information concerning RS-232 implementation for the 2714 or 2715 including wiring for connectors and null-modem adapters.

Operation Over the RS-232 Interface

The following equipment is required to operate the 2714 or 2715 Spectrum Analyzer over the RS-232 interface:

- System controller or terminal
- Software device driver
- 2714 or 2715 equipped with an RS-232 interface (Option 08)
- Interconnecting cable
- Application software
- Printer or plotter (optional)

Figure 1–1 shows two typical RS-232 system configurations. The top illustration shows a computer (PC) controlling the spectrum analyzer over the RS-232 interface; a plotter is connected to the computer over a Centronics interface. The lower illustration shows the spectrum analyzer connected directly to a plotter by the RS-232 interface.

System Controller

The system controller can be any general purpose computer or terminal equipped with an RS-232 interface (also called a COM port or serial interface). Specially built controllers can be used, but are beyond the scope of this manual. The techniques and programs discussed in this manual are appropriate to the IBM PC family of computers and their function-alike counterparts that support the MS-DOS, PC-DOS, or OS/2 environments.

Software Device Driver

The device driver is a program that handles input and output to the RS-232 interface on your computer. The driver for your system depends on the operating system and the programming language you are using. For example, if you are operating a PC, the RS-232 driver configuration may be set with the MS-DOS `MODE` command. If your control program is written in the BASIC or QuickBASIC language, optional arguments in the `OPEN` statement can supply RS-232 configuration settings.

2714 or 2715 Equipped with RS-232 Interface (Option 08)

Your 2714 or 2715 Spectrum Analyzer must be equipped with an RS-232 port to communicate over the RS-232 interface. If your 2714 or 2715 is equipped with the GPIB interface (Option 03), refer to *GPIB Operation* later in this section. Press the key sequence **[UTIL] [4] [9]** to see a list of the installed options and capabilities.

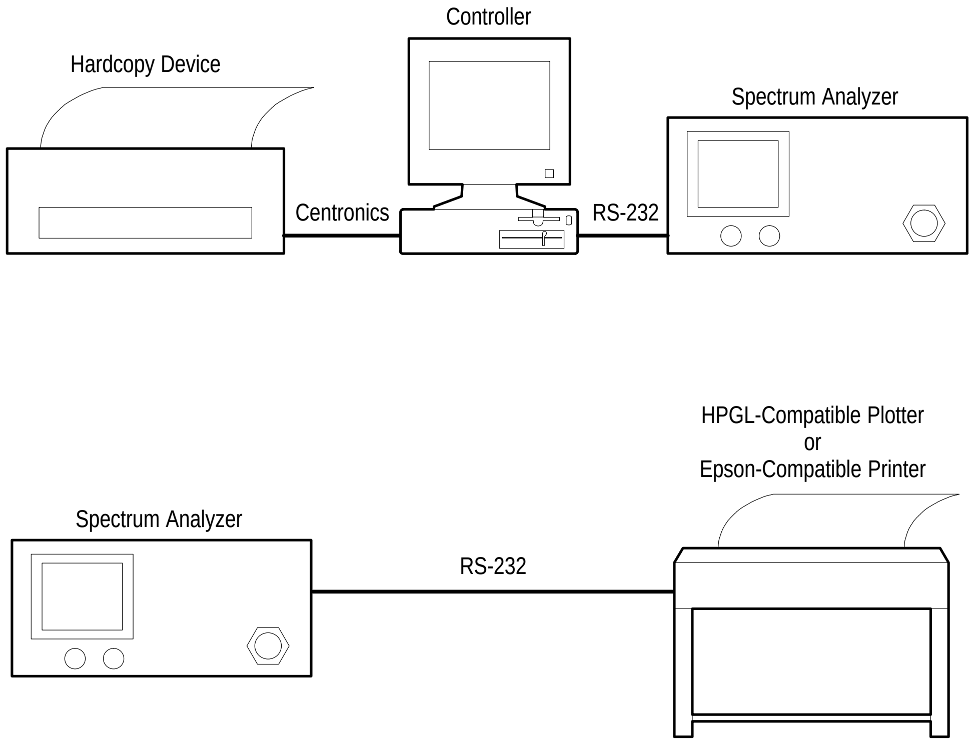


Figure 1–1: Two RS-232 System Configurations

Interconnecting Cable

An appropriate cable is required to connect between the controller and the spectrum analyzer. The pinout and connector type on the 2714 or 2715 are identical to the 9-pin connector used for PC/AT type RS-232 interfaces. Such cables are available in most computer stores. For some RS-232 devices, null-modem adapters will be needed. Refer to Appendix A for further information on connectors and adapters.

Application Software

Application software is the program or programs that control and acquire data from the spectrum analyzer. You can write your own programs using the information in this manual. Off-the-shelf software is also available.

Printer or Plotter (Optional)

A printer or plotter (not both simultaneously) can be connected to the RS-232 interface to provide hard-copy output. A printer is the preferred instrument for character-based data such as parameter values or instrument settings. Plotters provide superior results when displaying graphical data.

A printer or plotter cannot be connected to the spectrum analyzer’s interface when a computer is connected. For this reason you must choose between computer control or hard-copy output when working directly from the 2714 or 2715’s RS-232 interface. An alternate approach connects the computer to the spectrum analyzer interface while using a control program to acquire data from the spectrum analyzer. A second RS-232 port, a GPIB port, or a Centronics port on the computer is then used to produce output on a printer or plotter.

Setting Up for RS-232 Operation

Your equipment must be correctly configured before performing RS-232 operations. The following tasks must be completed:

- Installation of cables between the system components
- Configuration of the spectrum analyzer and device driver
- Installation of the device driver into controller memory
- Configuration of the (optional) printer or plotter

This section describes each task in detail.

Connecting the Equipment

Only one device (computer, plotter, or printer) can be attached to the spectrum analyzer's RS-232 interface. For systems consisting of a controller and the spectrum analyzer, simply connect one end of the interconnecting cable to each device. Figure 1–1 shows two possible configurations. See *Appendix A: RS-232 Concepts* for the cable configuration appropriate for your system.

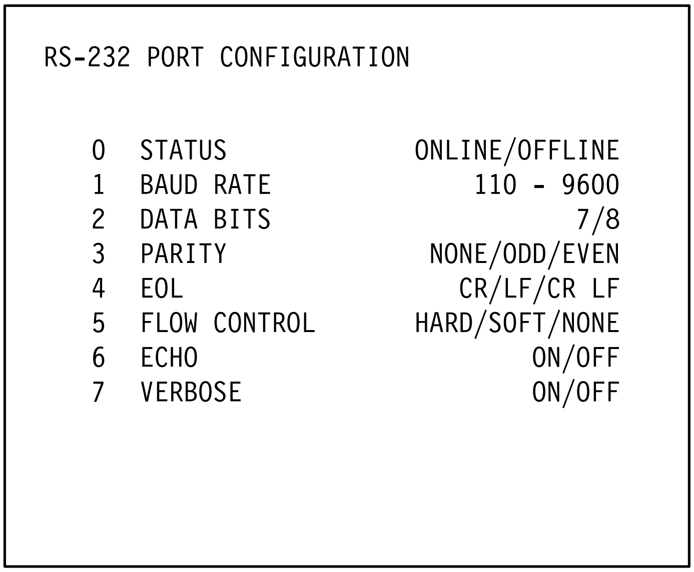
Configuring the Spectrum Analyzer

Both devices (the computer and spectrum analyzer) in an RS-232 system must be configured the same way. Before setting up the spectrum analyzer, be sure to check the configuration settings for the device with which you expect to communicate.

To set the spectrum analyzer configuration settings, turn on the power to the 2714 or 2715 and press the key sequence

[UTIL] [4] [0] [2]

on the spectrum analyzer KEYPAD. An RS-232 PORT CONFIGURATION Menu appears that is similar to the one shown in Figure 1–2. This menu allows for configuration of the spectrum analyzer's RS-232 parameters. Following are detailed descriptions of each parameter in the RS-232 PORT CONFIGURATION Menu.



```
RS-232 PORT CONFIGURATION


    0  STATUS             ONLINE/OFFLINE
    1  BAUD RATE              110 - 9600
    2  DATA BITS                     7/8
    3  PARITY              NONE/ODD/EVEN
    4  EOL                   CR/LF/CR LF
    5  FLOW CONTROL       HARD/SOFT/NONE
    6  ECHO                       ON/OFF
    7  VERBOSE                    ON/OFF
```

Figure 1–2: The RS-232 Port Configuration Menu

Placing the 2714 or 2715 Online. Item 0 of the RS-232 PORT CONFIGURATION Menu, STATUS, controls the RS-232 online/offline status. When the status is set to OFFLINE, the RS-232 interface is ignored; data is neither received nor transmitted. After all preparations have been completed and RS-232 operations are ready to begin, press **[0]** on the KEYPAD to toggle item 0 until the STATUS indicates ONLINE. The spectrum analyzer is then ready to exchange information over the RS-232 interface.

Setting the Baud Rate. Item 1 of the RS-232 PORT CONFIGURATION Menu, BAUD RATE, sets the baud rate of the spectrum analyzer. Baud rate represents how fast data is transmitted across the interface. To select a baud rate, repeatedly press **[1]** on the KEYPAD until the baud rate you desire is displayed. Baud rates ranging between 110 and 9600 are available.

The number of stop bits used is automatically selected by the spectrum analyzer when you change baud rates. If the baud rate is 110, two stop bits are selected. One stop bit is selected for all other baud rates.

***NOTE**. The spectrum analyzer baud rate must equal the baud rate of the other device connected to the RS-232 interface.*

Setting the Number of Data Bits. Item 2 of the RS-232 PORT CONFIGURATION Menu, DATA BITS, selects the number of data bits sent per character. This is either seven or eight. Eight bits must be selected for binary transfers. Press **[2]** on the KEYPAD to choose between seven or eight data bits.

Setting Parity. Item 3 of the RS-232 PORT CONFIGURATION Menu, PARITY, determines whether odd or even parity is used for data checking, or it selects no parity checking. The default setting is NONE. To change the PARITY selection, repeatedly press **[3]** on the KEYPAD until ODD, EVEN, or NONE is displayed.

Setting the Message Terminator. Item 4 of the RS-232 PORT CONFIGURATION Menu, EOL, selects the EOL (end-of-line) indicator used to terminate messages sent over the spectrum analyzer's RS-232 interface. The terminator can be CR (carriage return, ASCII 13), LF (line feed, ASCII 10), or CR LF (carriage return followed by line feed). To change the EOL status selection, repeatedly press **[4]** on the KEYPAD until CR, LF, or CRLF is displayed.

When a controller sends data, the spectrum analyzer interprets either CR or LF as a terminator, independent of the setting.

Selecting a Data Flow Control Method. Item 5 of the RS-232 PORT CONFIGURATION Menu, FLOW CONTROL, selects between three flow control methods: SOFT, HARD, or NONE. An explanation of each selection follows.

SOFT: When the spectrum analyzer sends data through the interface and SOFT flow control is enabled, CTRL-S (ASCII 19, same as pressing **[CTRL]** and **[S]** simultaneously) halts the data stream until CTRL-Q (ASCII 17) is received. Any other character received in the interim is ignored. This type of flow control can be used with a 3-wire setup because additional handshake lines are not needed.

When SOFT control is selected, the spectrum analyzer sends CTRL-S when its input data buffer is within 200 characters of being full. It sends CTRL-Q when the buffer empties to the point at which additional characters can be safely accepted (less than 200 characters remain in the buffer). If the input buffer is allowed to overflow, the spectrum analyzer discards the incoming data and signals an error (Event 372).

HARD: When HARD flow control is selected, the instrument sends data as long as the CTS (Clear-To-Send) line is TRUE and stops sending data if CTS goes FALSE. Additional handshake lines (more than a 3-wire RS-232 implementation) are required to support HARD flow control.

When receiving data and HARD flow control is selected, the spectrum analyzer asserts RTS (Request-To-Send) TRUE until the input buffer is within 200 characters of being full. It then sets RTS FALSE. Data is received while RTS is FALSE until the buffer overflows. If the buffer is allowed to overflow, the spectrum analyzer signals an error (Event 372), and incoming data is discarded.

NONE: No flow control is used.

Follow these general rules when selecting a flow control method:

- Do not use SOFT flow control when transmitting file or waveform data (binary transfers) because there is no guarantee that the ASCII-decimal values corresponding to CTRL-S and CTRL-Q do not appear in the input stream. Instead, specify HARD flow control or NONE for files and waveform data.
- If NONE is specified, you must ensure that buffers do not overflow. This can be done by allocating enough buffer space to handle most contingencies. A buffer size of 1200 is sufficient for most purposes. The 2714 or 2715 uses a 1200-byte, internal input buffer.

Selecting the Echo Feature. Item 6 of the RS-232 PORT CONFIGURATION Menu, ECHO, chooses ECHO modes of ON or OFF. ECHO mode is intended primarily as a means of interacting with the 2714 or 2715 from a “dumb” terminal, or for testing purposes. Press **[6]** on the KEYPAD to choose between ON or OFF.

When ECHO is OFF, the spectrum analyzer does not return the characters it receives to the controller. For most cases, ECHO should be OFF. However, set ECHO to ON when using a “dumb” terminal to control the spectrum analyzer.

When ECHO is ON, the spectrum analyzer echoes each character it receives back to the controller. This can cause problems for the control program if it is not expecting the characters. Additional time is required to process each returned character, so it is possible to experience buffer overrun at 9600 baud if the character rate is too high. After each command or query is completed, the spectrum analyzer prompts for further input by returning the string “>” to the controller.

For example, if ECHO mode is ON, “>” appears on the terminal or computer display screen. If the query “VPO?” is entered, the spectrum analyzer returns “VPO?” followed by a normal response to the query, such as “VPOLARITY POSITIVE”. It then appends the “>” to indicate that it is ready to receive additional commands.

ECHO mode is sometimes useful for interactive testing because it lets you see each character received by the spectrum analyzer. Following are some important ECHO mode characteristics:

- If SOFT flow control is enabled, CTRL-S and CTRL-Q are not echoed, but they perform their normal functions.
- If either CR or LF is received by the spectrum analyzer, it is echoed as the currently selected output terminator.

- Any other control character echoes as an up arrow (^) followed by a capital letter. For example, ^X represents pressing the **[CTRL]** and **[X]** keys at the same time.
- ECHO should not be ON with binary transfers.
- When ECHO is ON the prompt character appears on the display under other conditions: 1) when the instrument is powered up or placed on-line with ECHO mode ON, 2) when ECHO is turned on, and 3) after a device clear (break) is received.

Verbose Mode and Error Handling. Item 7 of the RS-232 PORT CONFIGURATION Menu, VERBOSE, turns VERBOSE mode ON and OFF. This feature is provided as an alternative to the GPIB SRQ mechanism. It is generally used when controlling the spectrum analyzer with a "dumb" terminal. Press **[7]** on the KEYPAD to choose between ON or OFF.

When ON, VERBOSE mode forces the spectrum analyzer to respond for each command it receives. The response will be one of the following:

- An event code for an abnormal condition
- A response for a successful query (FREq?)
- The string "OK" for a successful non-query

Refer to *Status Reporting* for additional information on error handling for instruments with the RS-232 interface.

Installing and Configuring the Device Driver

If you are using special applications software or a custom RS-232 driver, follow the detailed instructions for installing and configuring the device driver included with it. However, for PC-type controllers running MS-DOS, the driver is part of the operating system. You can configure a serial communications port with the MODE command by entering a command similar to the following example:

```
MODE COM1:9600,n,8,1
```

This command configures the COM1 interface to run at 9600 baud, no parity, 8 data bits, and 1 stop bit.

***NOTE**. You must use the same setup information for the controller and the spectrum analyzer.*

A program statement, such as OPEN in the BASIC language, is an alternative way to configure the driver. This method of driver configuration is recommended because it sets the driver to a known, and presumably correct, operating state from within the application program and just prior to actual operation. If the

MODE command is used, the last settings applied to the interface must be used. If this data is not used, your program will not work properly.

Configuring the (Optional) Printer or Plotter

A variety of printers and plotters are available for use with your system. The serial or parallel printer of your choice may be connected to the appropriate computer port. For example, the Centronics- or GPIB-compatible, 4-pen Tektronix HC100 plotter is recommended. Its four pens provide a useful complement to the four-trace capability of the 2714 or 2715.

A serial printer or plotter, such as the Tektronix HC100 Option 03, can be attached to the spectrum analyzer's RS-232 interface instead of a computer controller. This arrangement enables data transfer directly from the spectrum analyzer to the printer or plotter with a remote PLOT command when the system controller is unavailable. Of course, the spectrum analyzer must be correctly configured using the SCREEN PLOT CONFIGURATION Menu (key sequence **[UTIL] [4] [1]**).

Figure 1–1, located earlier in this section, shows two alternative configurations using a printer. A plotter could be substituted for the printer in either configuration.

Communicating with the Spectrum Analyzer

The RS-232 interface enables remote or automated control of the 2714 or 2715 Spectrum Analyzer. An application program (often called a test, measurement, or control program) determines 2714 or 2715 operations by exchanging spectrum analyzer-specific messages with the instrument.

The spectrum analyzer-specific messages are also referred to as device-dependent messages. They are generally understood by and meaningful to only the instrument, or class of instruments, for which they are designed. The organization of the spectrum analyzer-specific messages is explained in the next section of this manual. *Functional Groups* provides a summary of the messages. *Command and Query Definitions* describes the individual messages in detail, and *Programming* provides some programming examples.

Programmed commands and data are transmitted over the interface to the instrument as soon as they are delivered to the driver. If the command is a query (FREQ? for example), the spectrum analyzer formats a response immediately and sends it back to the computer. The control program is responsible for handling incoming data in a timely fashion.

Preparing the Software

After completing the setup procedures your equipment is ready for RS-232 operation, but you must still provide the software needed to control the spectrum analyzer. When creating new software this is usually a two step process. The first step is to establish the programming environment. Next you can create and run the control program. If you are using ready-made control software, simply follow the supplier's instructions.

The programming requirements for RS-232 control are more complex than those for GPIB operation. *Programming* contains a complete example of an interactive RS-232 control program. This program is functionally similar to the GPIB program example located later in this section (refer to *A GPIB Instrument Control Program* on page 1–23).

GPIB Operation (Option 03)

Option 03 adds a General Purpose Interface Bus (GPIB) port. This GPIB port conforms to the IEEE 488.1 Standard and to the Tektronix Interface Standard for GPIB Codes, Formats, Conventions, and Features. This standard promotes ease of operation and, so far as possible, makes this spectrum analyzer compatible with other Tektronix instruments and with GPIB instruments from other manufacturers.

The IEEE 488.1 Standard establishes electrical levels, connector configuration, and signal protocols for communication between two or more electronic instruments using a common multi-line bus structure. The bus structure, known as the GPIB, consists of eight data lines, eight dedicated control signal lines, a shield, and various grounds.

Data is transferred via eight data lines in a bit parallel, byte serial fashion. That is, the eight bits of a data byte are placed on the eight data lines simultaneously. As soon as they are transferred, the next 8-bit data byte is placed on the lines and is transferred. Data consists of instrument commands and queries, control settings, parameter values, or display information.

The eight control lines are divided into three transfer control (handshake) lines and five interface management lines. Handshaking and interface management are necessary because the bus operates asynchronously. This means that signals can be generated by one instrument without regard for what other instruments may be doing, or the rate at which other instruments can carry out an operation. For instance, two instruments may try to send information simultaneously, or a high speed instrument may try to send data to a slower instrument.

Instruments connected to the bus are designated as talker, listener, or both talker and listener. A listener can only receive information over the bus and a talker can only send information. A talker and listener can do both (but not simultaneously).

One instrument is usually designated as the system controller. This is generally a computer that determines, through software, when specific instruments are activated as talkers or listeners. Each instrument is assigned a unique address between 0 and 30, but only 15 instruments can be connected to the bus simultaneously.

The following example illustrates how data transfer typically takes place (except in the case of abnormal events; see *Status Reporting*).

1. The instrument on the bus that is designated as system controller determines (through operator intervention or program control) that it needs to send a message to one of the other instruments.

2. Using the data and interface management lines, the controller first addresses the desired instrument as a listener. This is called LISTENING an instrument.

3. Instruments on the bus are normally idle, and they signal using the handshake lines when they are ready to receive data. The controller then places the first byte of the message on the bus, indicating the type of information it wants.

4. Next the controller signals, using the handshake lines, that the data byte is ready.

5. As the listener accepts the data byte, it signals over the handshake lines that it has done so. The controller then removes the data from the data lines.

6. The process from steps 3, 4, and 5 is repeated until the entire message has been transferred.

7. The controller indicates that the last data byte has been sent. Depending on the option selected, one of two methods is used: 1) Signaling over the end or identify (EOI) interface management line simultaneously with the last data byte, or 2) Appending the ASCII codes for carriage return (CR) and line feed (LF) to the end of the message and simultaneously signaling EOI.

8. When the message is complete, the controller normally UNLISTENS the instrument. If a message requires a response, the controller then addresses the instrument as a talker (TALKS the instrument).

9. Now the instrument places the first byte of the response on the data bus and signals that it is ready.

10. After the controller reads the byte, it signals (over the handshake lines) that it has done so and is ready to receive more data. The process repeats until EOI is detected, at which point the controller normally UNTALKS the instrument.

The data transfer process is transparent to you. It is carried out by the spectrum analyzer, the GPIB board in your controller, and the device driver software (generally supplied with the GPIB board). In the following subsections you will learn how to set up your spectrum analyzer for GPIB operation. See *Appendix B: GPIB System Concepts* for additional information concerning IEEE 488.1 and the GPIB.

Operation Over the GPIB

The following equipment is required to operate the 2714 or 2715 Spectrum Analyzer over the General Purpose Interface Bus (GPIB):

- System controller
- Software device driver
- 2714 or 2715 equipped with the GPIB interface (Option 03)
- Interconnecting cable
- Application software
- Printer or plotter (optional)

Figure 1–3 shows an example of a simple GPIB system consisting of a printer and plotter.

System Controller

The system controller can be any general purpose computer equipped with a GPIB board. Specially built controllers can also be used, but are beyond the scope of this manual. The techniques and programs discussed in this manual are appropriate to the IBM family of personal computers (PCs) and their function-alike counterparts, which support the MS-DOS, PC-DOS, or OS/2 environments.

To function as a controller, your computer must be equipped with a GPIB board. Tektronix supplies three National Instruments GPIB boards for your convenience:

- PC-GPIB Package provides a PCII/IIA board; order S3FG210
- AT-GPIB Package provides a 16-bit AT Bus interface board; order S3FG220
- MC-GPIB Package provides a 16-bit Micro Channel interface board; order S3FG230

Software Device Driver

The device driver is a program (usually supplied with the GPIB board) that tells your computer how to access the board. For the National Instruments PCII, PCIIA, or PCII/IIA GPIB boards, the device driver is a file named `GPIB.COM`. An additional program is usually supplied that enables you to correctly configure the driver by providing information such as the instrument address and the type of message terminator. The National Instruments program is named `IBCONF.EXE`.

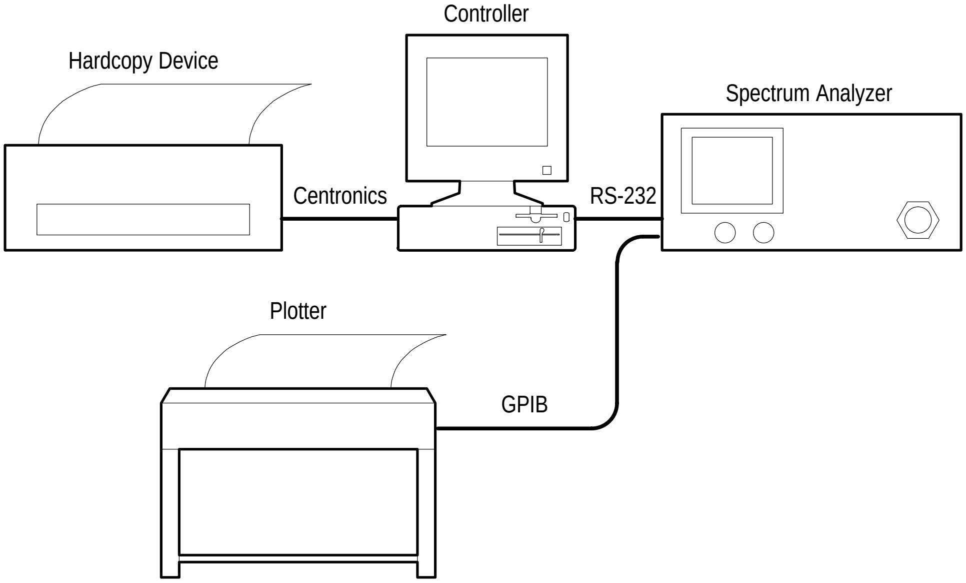


Figure 1–3: Typical Small Instrument System for GPIB

2714 or 2715 Equipped with the GPIB Interface (Option 03)

Your 2714 or 2715 must be equipped with the Option 03 GPIB interface to operate over the General Purpose Interface Bus. Refer to *RS-232 Operation (Option 08)* on page 1–1 for configuration information if your instrument has the RS-232 interface. Press the key sequence **[UTIL] [4] [9]** to see a list of the installed options and capabilities.

Interconnect Cable

An appropriate interconnect cable is required to connect the controller to the spectrum analyzer. Cables may be purchased from Tektronix by ordering one of these part numbers:

- P/N 012-0991-01 (1 meter)
- P/N 012-0630-01 (2 meter)

Application Software

Application software is the program or programs that control and acquire data from the spectrum analyzer. You can write your own programs with the information in this manual. However, you will need the applications interface software supplied by the GPIB board manufacturer. For the PCII/IIA board and the QuickBASIC language, these programs have names such as `QBIB4.OBJ`, `QBIB4728.OBJ`, and `QBDECL4.BAS`. The programs include the BASIC device function calls which enable you to communicate easily over the GPIB. The function calls are an integral part of your application programs.

Printer or Plotter (Optional)

A printer, a plotter, or both can be added to your system to provide hard-copy output. Printers are preferred for character-based data such as parameter values or instrument settings. Plotters provide superior results when displaying graphical data. A convenient approach is to install a printer on a parallel port of the controller and a GPIB-compatible plotter on the bus. With this approach, graphical data can be plotted directly from the spectrum analyzer when the controller is not available.

See *Setting the TALK ONLY Option* on page 1–18.

Setting Up for GPIB Operation

Your equipment must be correctly configured before GPIB operations can be performed. The following tasks must be completed:

- Installation of cables between the system components
- Configuration of the spectrum analyzer and device driver
- Installation of the device driver into controller memory
- Configuration of the (optional) printer and/or plotter

This section describes each task in detail.

Connecting the Equipment

If your system consists of a controller and spectrum analyzer, you can simply connect one end of the interconnecting cable to each instrument. A star configuration, daisy chain configuration, or combination of these (Figure 1–4) should be used when more than two instruments are on the bus. Up to 15 instruments can be connected.

To maintain electrical performance of the bus, use only one 2-meter cable per instrument, and ensure that at least 2/3 of the connected instruments are powered up.

Configuring the Spectrum Analyzer

Turn on the power to the spectrum analyzer. Press the key sequence:

[UTIL] [4] [0] [0]

A GPIB PORT CONFIGURATION Menu appears. It should resemble the one shown in Figure 1–5. You will use this menu to configure the GPIB parameters.

Placing the 2714 or 2715 Online

Item 0 of the GPIB PORT CONFIGURATION Menu, STATUS, controls the GPIB ONLINE/OFFLINE status (see Figure 1–5). After all preparations have been completed and GPIB operations are ready to begin, press **[0]** on the KEYPAD to toggle item 0 until the STATUS indicates ONLINE. The spectrum analyzer is then ready to exchange information over the GPIB.

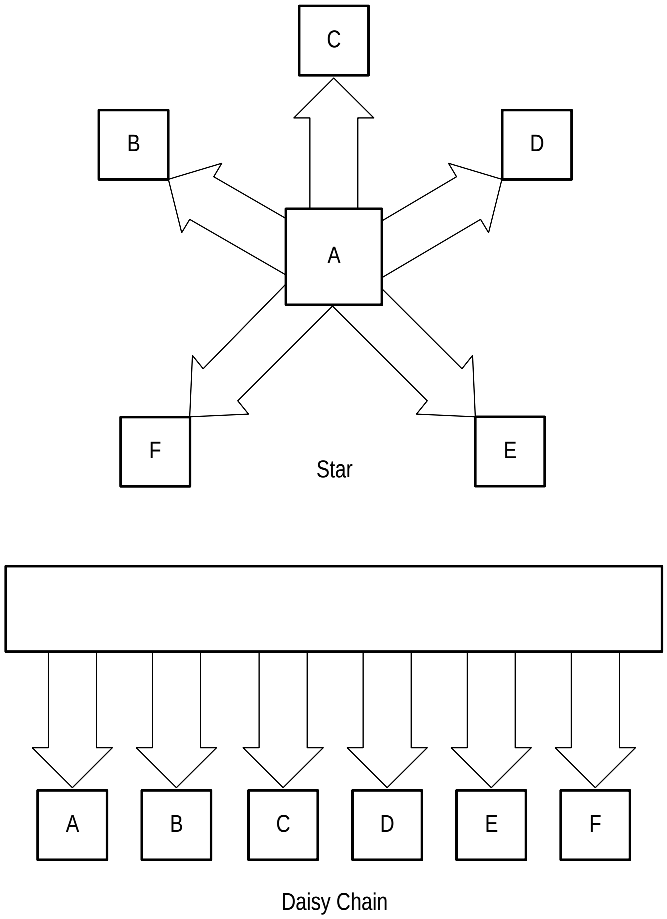
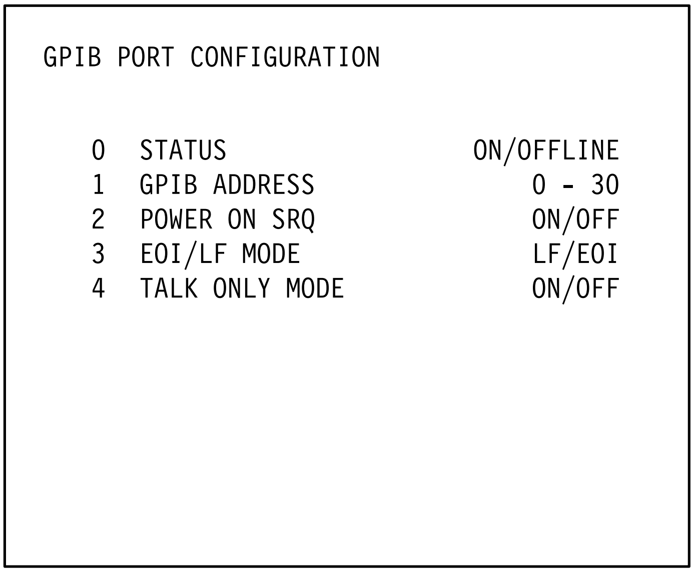


Figure 1–4: Connecting Multiple Instruments on the GPIB

Setting the GPIB Device Address

Item 1 of the GPIB PORT CONFIGURATION Menu, GPIB ADDRESS, sets the spectrum analyzer’s GPIB device address. You must assign a primary address to the spectrum analyzer (the 2714 or 2715 does not support secondary addresses). The address can have a value from 0 through 30. However, addresses 0 and 30 are usually reserved for system controllers.



```
GPIB PORT CONFIGURATION


   0  STATUS                ON/OFFLINE
   1  GPIB ADDRESS              0 - 30
   2  POWER ON SRQ              ON/OFF
   3  EOI/LF MODE               LF/EOI
   4  TALK ONLY MODE            ON/OFF
```

Figure 1–5: The Spectrum Analyzer's GPIB PORT CONFIGURATION Menu

The address you assign is not critical, but it must not be the same address used for any other instrument on the bus.

NOTE. *The GPIB address assigned to the spectrum analyzer must be the same as the one that was used to configure the device driver for the spectrum analyzer.*

To assign the address, select item 1, GPIB ADDRESS, from the GPIB PORT CONFIGURATION Menu. Follow the on-screen prompts to enter the desired address using the KEYPAD for data entry. If the spectrum analyzer is the only instrument on the bus, we suggest using 1 as the address. The address you set is read immediately by the spectrum analyzer and is permanently retained in non-volatile memory.

The Power-On SRQ

Item 2 of the GPIB PORT CONFIGURATION Menu, POWER ON SRQ, causes the spectrum analyzer to produce an SRQ at power up. To generate a POWER ON SRQ, press **[2]** on the KEYPAD until the status changes to ON.

Normally there is no need to have the spectrum analyzer generate an SRQ when it powers up. Therefore, the default setting of item 2, POWER ON SRQ, is OFF. However, some test sequences require that the power to the spectrum analyzer is removed (power down). Under these conditions it may be beneficial for the program to sense the return of power.

Setting the Message Terminator

Item 3 of the GPIB PORT CONFIGURATION Menu, EOI/LF MODE, selects the message terminator. Whenever a message is transmitted over the bus, the instrument sending the message must signify to other instruments on the bus (including the system controller) that the message has been completed. This is done in one of two ways:

- The interface management line named End Or Identify (EOI) is brought to its low state simultaneously with the last data byte that is transmitted.
- The ASCII codes for carriage return (CR) and line feed (LF) are appended to the message string. EOI is still asserted (brought to its low state) simultaneously with the transmission of LF.

All Tektronix instruments and controllers are equipped to use the EOI selection. You should, therefore, toggle item 3 of the GPIB PORT CONFIGURATION Menu until its status changes to EOI. The LF OR EOI setting is included for controllers that do not use the EOI signal line. The selection you choose is permanently retained in non-volatile memory.

Setting the TALK ONLY Option

Item 4 of the GPIB PORT CONFIGURATION Menu, TALK ONLY MODE, selects the spectrum analyzer's TALK ONLY mode.

TALK ONLY mode must be selected to send the spectrum analyzer's output directly to a plotter without the use of a controller. Complete these steps to send the spectrum analyzer's display directly to a plotter:

1. Disconnect all instruments except the spectrum analyzer and the plotter from the bus.
2. Place the plotter in the LISTEN ONLY mode (usually done with controls on the plotter).
3. Press the key sequence **[UTIL] [4] [0]** and then press **[4]** until the TALK ONLY status indicates ON.
4. Press the front-panel key labelled **[PLOT]**.

TALK ONLY mode must be disabled when the spectrum analyzer is used with a controller, because the spectrum analyzer must talk to and listen to the controller. To use the spectrum analyzer with a controller, press **[4]** on the KEYPAD until the status indicates OFF. The system controller will determine when the spectrum analyzer should be addressed as a talker or listener.

Configuring the Device Driver

Instructions for configuring the device driver should be included with your GPIB board. For example, complete the following steps when using a National Instruments PCII/IIA board:

1. Run the `IBCONF.EXE` program to configure the driver.

2. Follow the on-screen prompts and ensure that the BOARD CHARACTERISTICS screen resembles one of those shown in Table 1–1 and 1–2.

3. Create or edit the DEVICE CHARACTERISTICS screen for a device named TEK_SA (see Table 1–3).

NOTE. *You must assign the same GPIB address to the spectrum analyzer that was used when configuring the device driver for the spectrum analyzer. Use the EOI message terminator for all Tektronix controllers.*

Table 1–1: National Instruments PCII Board Characteristics

Primary GPIB address	0
Secondary GPIB address	NONE
Timeout setting	T30s
EOS byte	00H
Terminate Read on EOS	no
Set EOI with EOS on Write	no
Type of compare on EOS	7-bit
Set EOI with last byte of Write	yes
GPIB-PC model	PC2
Board is system controller	yes
Local lockout on all devices	no
Disable auto serial polling	yes
High-speed timing	no
Interrupt jumper setting	7
Base I/O address	2B8H
DMA channel	1
Internal clock freq (in MHz)	8

Table 1–2: National Instruments PCIIA Board Characteristics

Primary GPIB address	0
Secondary GPIB address	NONE
Timeout setting	T30s
EOS byte	00H
Terminate Read on EOS	no

Table 1–2: National Instruments PCIIA Board Characteristics (Cont.)

Set EOI with EOS on Write	no
Type of compare on EOS	7-bit
Set EOI with last byte of Write	yes
GPIB-PC model	PC2A
Board is system controller	yes
Local lockout on all devices	no
Disable auto serial polling	yes
High-speed timing	no
Interrupt jumper setting	7
Base I/O address	02E1H
DMA channel	1
Internal clock freq (in MHz)	6

Table 1–3: TEK_SA Device Characteristics

Primary GPIB address	1
Secondary GPIB address	NONE
Timeout setting	T30s
EOS byte	00H
Terminate Read on EOS	no
Set EOI with EOS on Write	no

Installing the Device Driver

Before your computer can transfer information over the GPIB, it must know how to access the GPIB board and the spectrum analyzer. The device driver tells it how. If you are using a National Instruments PCII/IIA board, the device driver is a program named `GPIB.COM` created and modified by another National Instruments program named `IBCONF.EXE`. If you are using a board from another manufacturer, the appropriate driver should have accompanied your board.

The device driver program must be installed whenever you wish to use the GPIB. Use the following procedure. Refer to your DOS manual if you need help creating or modifying files.

1. Copy `GPIB.COM` to your computer’s root directory.
2. Add the following line to your `CONFIG.SYS` file:

```
device=GPIB.COM
```

3. If `CONFIG.SYS` does not already exist in the controller's root directory, create a new `CONFIG.SYS` file.

4. Reboot your controller.

The GPIB device driver is loaded into memory whenever you boot your computer. It then remains in memory until the computer is turned off or until a reboot is performed.

Configuring the (Optional) Printer or Plotter

A variety of printers and plotters are available that can be used with your system. We recommend a serial or parallel printer connected to the appropriate computer port, and/or an HPGL-compatible plotter connected to the GPIB. This arrangement allows data to be sent directly from the spectrum analyzer to the plotter when the system controller is unavailable. The Tektronix HC100 plotter is recommended. Its four pens provide a useful complement to the four-trace capability of the 2714 or 2715.

Printer Configuration. Configuration of the printer is independent of the GPIB. Consult your printer and computer manuals for information on setting up the printer and corresponding computer communications port.

Plotter Configuration. Plotter configuration procedures vary. Consult your plotter manual for the configuration appropriate to your plotter.

When using a Tektronix HC100 plotter, set its rear-panel DIP switches as follows:

OFF				16	8	4	2	1	1
ON	GPIB	HPGL	STD	Address (Listen Only = 31)					0
	Down	Down	Down	Up	Up	Up	Up	Up	

All bits should be set when the Tektronix HC100 plotter is in LISTEN ONLY mode, and its power must be cycled to load the settings into memory. You must also correctly configure the plotter DEVICE CHARACTERISTICS using the `IBCONF` file.

***NOTE**. Be sure to use the same GPIB address for the HC100 DIP switches and the DEVICE CHARACTERISTICS.*

Communicating with the 2714 or 2715

The GPIB enables remote or automated control of instruments on the bus (in this case, a spectrum analyzer). An application program (often called a test, measurement, or control program) determines spectrum analyzer operations by exchanging messages with the spectrum analyzer. The messages can be of the generic GPIB type, or they can be instrument-specific.

Generic messages are usually carried out by GPIB hardware and the GPIB device driver without intervention by the operator or programmer. They typically implement routine housekeeping chores such as instrument addressing, handshaking, requesting service, or terminating messages.

The instrument-specific messages are also referred to as device-dependent messages. They are generally understood by, and meaningful to, only the instrument or class of instruments for which they are designed. The organization of the instrument-specific messages is explained in the next section of this manual. *Functional Groups* provides a summary of the messages. *Command and Query Definitions* describes the individual messages in detail, and *Programming* provides some programming examples for the National Instruments GPIB/2714 or 2715 combination working in the QuickBASIC environment.

The spectrum analyzer is addressed as a talker or listener to send or receive messages, depending on whether messages are being sent to or received from the system controller. The GPIB system software provided with your GPIB card automatically addresses the spectrum analyzer as a talker or listener depending on the callable subroutine used. The device-dependent messages are then transferred between the controller and the spectrum analyzer over the GPIB as one or more eight-bit bytes of information. Proficiency in controlling the spectrum analyzer is the key to programming these messages efficiently.

Preparing the Software

After completing the setup procedures your equipment is ready for GPIB operation, but you must still provide the software needed to control the spectrum analyzer. When creating new software this is usually a two step process. The first step is to establish the programming environment. Next you can create and run the control program. If you are using ready-made control software, simply follow the supplier's instructions.

When creating your own QuickBASIC software, you must ensure that QuickBASIC has the necessary GPIB information. Use the following procedure. Refer to your DOS manual if you need help creating or modifying files.

- Copy the files `QBIB4.OBJ, GPIB.QLB, GPIB.LIB, BQLB45.LIB, and QBIB4728.OBJ` from the National Instrument's disk to the QuickBASIC directory.

- Create the Quick library by typing this command from the DOS command line:

```
LINK /Q QBIB4.OBJ QBIB4728.OBJ,GPIB.QLB,,BQLB45.LIB
```

- To make your QuickBASIC program a stand-alone *.EXE file, you need an additional library file. Type this command from the DOS command line:

```
LIB GPIB.LIB + QBIB4.OBJ + QBIB4728.OBJ;
```

- Start QuickBASIC using this command:

```
QB /L GPIB.QLB
```

This procedure ensures that the National Instruments GPIB subroutines needed to control devices on the bus are present in the QuickBASIC environment. When using another version of QuickBASIC, use the analogous files and procedures indicated in the READ-QB.DOC document file from National Instruments.

A GPIB Instrument Control Program

You will learn more about controlling the spectrum analyzer in Sections 4 through 6 in this manual. However, we have provided a simple program here so you can check the operation of your system and observe typical interactions between the controller and spectrum analyzer. Be aware that the program is very basic; it contains no error checking and may hang up the controller (requiring you to reboot) if incorrect or unacceptable commands or queries are entered. It will, however, accept uppercase or lowercase entries.

Alternatively, you can use the IBIC program supplied with the National Instruments PCII/IIA GPIB board. It enables you to communicate with the spectrum analyzer, but requires that you learn how to use a few simple subroutines such as IBWRT() and IBRD(). See your National Instruments documentation for details.

Follow these steps to use the Example Program 1-1 located on the following pages (REM $INCLUDE: 'QBDECL4.BAS').

1. Start QuickBASIC according to the instructions in *Installing and Configuring the Device Driver* on page 1–9. Enter the program. Be sure to enter the program exactly as it is written. The spectrum analyzer must be named TEK_SA by the IBCONF program.

2. Place the spectrum analyzer ONLINE by selecting item 0 from the GPIB PORT CONFIGURATION Menu (press the key sequence **[UTIL] [4] [0] [0]**). The instrument is nominally ONLINE, but is not yet handshaking with the controller.

3. Start the program. The computer display shows:

```
2714 or 2715 SHOULD NOW BE HANDSHAKING
NDAC SHOULD BE DISPLAYED
PRESS ANY KEY TO CONTINUE
```

   When the spectrum analyzer is handshaking with the controller, NDAC (Not Data ACcepted) is displayed at the lower right of the spectrum analyzer’s screen. NDAC is asserted most of the time. It is unasserted only briefly following receipt of a message to indicate that the message has been accepted (see *Appendix B: GPIB System Concepts*).

4. Press any key. The word REMOTE should appear at the lower left of the spectrum analyzer's screen and the controller should display these messages:

   ```
   2714 or 2715 SHOULD NOW BE IN REMOTE MODE
   PRESS ANY KEY TO CONTINUE
   ```

   The National Instruments software places the spectrum analyzer in remote mode whenever a message is sent (the message HDR ON was transmitted). Unless the GPIB local lockout command is issued, any spectrum analyzer key which alters its measurement status may be pressed to return the spectrum analyzer to local mode. (For example, the **[MENU]** keys do not change the status, but items selected from a menu may change the status.)

5. Press the **[VID FLTR]** key twice. The REMOTE message should disappear from the spectrum analyzer screen.

6. Press any key. This message should appear:

   ```
   ENTER MESSAGE TO SEND
   ```

   Enter the message you want to send, which can be either a command or query. For example, enter this query requesting the spectrum analyzer to identify itself:

   ```
   ID?
   ```

7. REMOTE should reappear on the spectrum analyzer screen. The word TALKER or LISTENER will also appear momentarily. These words are displayed when the spectrum analyzer enters the indicated mode, but because of timing considerations, they do not always appear for short commands, queries, and responses. You should see a response similar to the following one on the controller's screen:

   ```
   ID TEK/2714 or 2715,V81.1,"VERSION 02.28.92 FIRMWARE",
   "GPIB", "NVM 12.88", "OPT NVM 12.88";
   ```

8. The actual response depends on the options in your instrument.

9. You will then be asked:

   ```
   SEND MORE (Y/N)?
   ```

Enter “Y” to send more; other answers end the program.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example: GPIB Test Program

```
REM $INCLUDE: 'QBDECL4.BAS'
   RD$ = SPACE$(3000)
   CLS
   CALL IBFIND("GPIB0", BD%)
   V% = 0
   CALL IBSRE(BD%, V%)
   CALL IBFIND("TEK_SA", BD%)
   PRINT "2714 or 2715 SHOULD NOW BE HANDSHAKING"
   PRINT "NDAC SHOULD BE DISPLAYED"
   PRINT:PRINT "PRESS ANY KEY TO CONTINUE"
   DO WHILE INKEY$ = ""
      LOOP
   WRT$ = "HDR ON"
      CALL IBWRT(BD%, WRT$)
   PRINT "2714 or 2715 SHOULD NOW BE IN REMOTE MODE"
   PRINT:PRINT "PRESS ANY KEY TO CONTINUE"
   DO WHILE INKEY$ = ""
      LOOP
SEND.RCV:
   CLS
   PRINT:PRINT "ENTER MESSAGE TO SEND"
      PRINT : INPUT WCALL IBWRT(BD%, WRT$)
   QUES = INSTR(1, WRT$, "?")
   HOLD.TIME = TIMER
   DO WHILE TIMER < HOLD.TIME + 1
      LOOP
   IF QUES = 0 THEN GOTO MORE
      CALL IBRD(BD%, RD$)
   PRINT:PRINT "THE REPLY IS:"
   PRINT:PRINT MID$(RD$, 1, IBCNT%)
MORE:
   PRINT:PRINT
   INPUT "SEND MORE (Y/N)? "; Y$
   IF Y$ = "Y" THEN GOTO SEND.RCV
END
```

Message Structure

Instrument-Specific Message Structure

Communications between the system controller and the 2714 or 2715 are accomplished by instrument-specific messages. Instrument-specific messages exchanged over the RS-232 or GPIB interface control the measurement and display functions of the spectrum analyzer. These messages are always transmitted over the data lines (with the exception of the EOI message terminator).

Instrument-specific messages control parameters such as center frequency, span/division, reference level, and resolution bandwidth. It is the system programmer's task to efficiently compile a series of messages in a script designed to implement specific tests and measurements. The script, or control program, is written in a conventional computer language and embodies specific spectrum analyzer commands and queries.

When the GPIB interface is installed, generic GPIB messages are exchanged between the system controller and the 2714 or 2715 in addition to instrument-specific messages. Generic GPIB messages exercise control over the bus and carry out routine system operations such as instrument addressing, handshaking, requesting service, and terminating messages. GPIB messages may be transmitted over the handshake lines or interface management lines (uni-line messages), or they may be transmitted over the data lines (multi-line messages). The GPIB hardware and software usually sends and receives these messages in a way that is transparent to the system operator or programmer. Refer to *Appendix B: GPIB System Concepts* for additional information about uni- and multi-line messages.

What Is A Message?

An instrument-specific message consists of three or more 8-bit bytes of information that are transferred between the spectrum analyzer and the system controller. Each byte represents an ASCII character or binary data. A message may be an input message or an output message. It may contain one or more message units.

For instance, here is an example of a message from Mary to John:

```
John

dinner - put in oven; washing machine - start; bank -
withdraw how much?; cat - let her in

Bye bye
```

John’s response may resemble this one:

```
Mary

$100

Bye bye
```

He could also respond this way:

```
Mary

Withdraw - $100

Bye bye
```

These messages are structured in a similar manner. Each contains a salutation (`John` and `Mary`). Each message consists of one or more message units. For instance, Mary’s message to John has four units. One of these is a query and is identified by a question mark (?). Each message unit begins with a “header” describing what the message is about (such as `dinner` or `cat`). The header is separated from its object or argument by a dash (–), which is the argument delimiter. Message units are separated or delimited by semicolons (;).

Each of John’s messages to Mary consist of a single response indicating how much money she should withdraw. If John thinks Mary will remember her own question (`withdraw how much?`), he may simply reply “`$100`” as in the first example. However, to relate his response to her question he may answer “`Withdraw – $100`” as in the second example. The latter form is equivalent to receiving a response from the spectrum analyzer when `HDR ON` is selected (see *Command and Query Definitions*). Both messages close with a message terminator “`Bye bye`”.

The instrument-specific messages for the 2714 or 2715 are constructed in a similar way. The following definitions clarify the structure.

Message Unit

A message unit is a single command, query, or response.

Input Message

An input message is one or more message units, along with any message unit delimiters (separators) and a message terminator, transmitted from the controller to the 2714 or 2715.

Output Message

An output message is one or more message units, along with any message unit delimiters (separators) and a message terminator, transmitted from the 2714 or 2715 to the controller.

Message Unit Delimiter

A semicolon (;) must be used to delimit or separate message units in a message. Following the last message unit, the use of a delimiter is optional with one exception. The 2714 or 2715 always appends a message unit delimiter as the last data byte when it sends a response.

If desired, you may substitute the line feed character for the semicolon in the case of responses (see the `MSGdlm` command in *Command and Query Definitions*). Do not confuse the line feed character with the optional message terminator discussed next.

Message Terminator (RS-232)

When a controller sends data to the 2714 or 2715, the spectrum analyzer interprets both CR and LF as message terminators. Messages sent by the spectrum analyzer over the RS-232 interface can be terminated in the following ways:

- By CR only (carriage return, ASCII 13)
- By LF only (line feed, ASCII 10)
- By CR-LF (carriage return followed by line feed)

See *Setting the Message Terminator* in the *Setting Up for RS-232 Operation* part of *Introduction to Programming* for instructions about configuring the message terminator for RS-232 instruments.

Message Terminator (GPIB)

Messages exchanged over the GPIB interface (Option 03) can be terminated in one of two ways:

- The EOI interface management line is brought low (asserted) simultaneously with the last byte of the message.
- ASCII codes for carriage return (CR) and line feed (LF) are appended to the end of message, and the EOI interface management line is asserted simultaneously with transmission of the LF character.

All Tektronix instruments assert the EOI line. The CR-LF terminator option is provided for instruments that do not use the EOI line. If you are using such an instrument, select the CR-LF option (see *Setting the Message Terminator* in the *Setting Up for GPIB Operation* part of *Introduction to Programming*). Terminator control is handled automatically by the GPIB hardware and software.

Command

A command generally consists of a command mnemonic or header, header delimiter, argument(s), and argument delimiter. For example, the command

```
SAVE A:ON, B:ON
```

enables the SAVE A and SAVE B functions. In this example, `SAVE` is the command header. The space between `SAVE` and `A:ON` is the header delimiter.

A:ON and B:ON are arguments, and the comma (,) following A:ON is the argument delimiter.

Some commands have no arguments, or only one argument, and may not require header or argument delimiters. For instance, the command

```
GTL
```

returns the 2714 or 2715 to local control without adding arguments or delimiters.

Query

A query consists of a query mnemonic or header, a question mark (?), header delimiter, and an argument. Many queries do not require an argument.

For example, the query

```
CURve?
```

may be used with arguments A, B, C, or D to determine the contents of a display register. The query

```
CFSF?
```

simply returns a response that indicates whether the on-screen frequency is Center Frequency or Start Frequency.

Response

A response is information returned as the result of a query. It consists of an optional response mnemonic or header, header delimiter, argument(s), and argument delimiter. For example,

```
CFSF CENTER

CFSF START
```

are the possible responses to the CFSF? query.

Mnemonic or Header

A header is a short name associated with each command, query, or response (for example, FREq, REF, MAR). Whenever possible, headers are mnemonic in nature so the name indicates the function.

Header Delimiter

A header delimiter is a blank space. Command headers, response headers, and the question mark (?) following a query header are delimited or separated from any following arguments by a space. The space is optional if there are no arguments.

Digit

Digits are any of these numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

Units

Units specify the unit of measure. Except in the case of decibels (dBs), only the first letter of the units is used. Message context determines how the first letter is interpreted. For instance, the command

```
FREq 10 M
```

is the same as

```
FREq 10 MHZ
```

For either command the M is interpreted as MHz because frequency corresponds to the FREq command. In a similar manner, the letter M would indicate milliseconds when used within the TIMe 10 M command.

Engineering units can be appended to certain number arguments (FREq 10E+6 is the same as FREq 10 MHZ).

Commands such as REFlvl require the entire dB unit (dBmV, dBm, etc.) to avoid confusion. You can place as many spaces as desired between the number and its units.

Responses with numerical arguments DO NOT append units to the argument. The programmer must keep track of the units, or use a query that responds with the units currently in use (for instance, the RLUnit? query).

Argument

An argument is the value that a command, query, or response transfers to or from its associated 2714 or 2715 setting(s). For instance, from the command

```
FREq 200 MHZ
```

the value of 200 MHz is transferred to the center frequency setting. Arguments may be numbers (with or without units), characters, strings, or linked with a colon (:). A block of binary data may comprise the argument of some waveform commands and responses.

Number Argument

Number refers to a decimal number consisting of one or more digits. Three number formats are possible:

nr1	Integer (no decimal point)
nr2	Floating point number (decimal point required)
nr3	Integer or floating point number in scientific notation (2.0E+3 or 2.000E+3, for example, instead of 2000)

Character Argument

A character argument consists of one or more letters usually expressing a word or mnemonic. For instance, ON and OFF are arguments for commands that control 2714 or 2715 functions such as AREs (auto resolution bandwidth).

String Argument

String arguments consist of one or more characters, including spaces, enclosed in quotation marks ("). They are used with commands such as TITLe to convey messages meant to be displayed or plotted. Double quotation marks ("") must be used when the quotation marks are to be part of the message. In this example

```
PRINT ""PRESS ANY KEY TO CONTINUE""
```

the message "PRESS ANY KEY TO CONTINUE" is displayed on the screen.

Link Argument

Link arguments provide a method of passing related argument parameters. A colon character (:) separates linked arguments. For example, the VRTdsp command can set both the vertical display mode and its related scale factor by linking arguments as follows:

```
VRT LOG:5
```

The argument LOG selects logarithmic display mode, and the linked argument 5 sets the vertical scale factor to 5 dB/Division.

Binary Block Argument

A binary block argument is a sequence of binary numbers. The sequence is preceded by the ASCII code for percent (%), a two-byte binary integer representing the number of binary numbers, and a checksum byte. Following the sequence is an additional checksum byte that provides an error check of the binary block transfer.

Refer to the CURve? command in *Command and Query Definitions* for examples.

Argument Delimiter

A comma (,) must be used to delimit or separate multiple arguments in a message unit. It should not be used as the last character in a message.

Message Buffering (RS-232)

The 2714 or 2715 Spectrum Analyzer buffers each input message that it receives. Message processing begins as soon as messages are received by the 2714 or 2715. It does not wait for the message terminator. Once processing begins, the 2714 or 2715 remains busy until it is done executing the commands in its input buffer unless the process is stopped by a BREAK. BREAK is sensed by the interface as a null character together with a framing error. If an error is detected while transferring a command or query, the rest of the message (up to the message terminator) is discarded.

Buffering is handled by specifying a flow control method for the RS-232 (Option 08) operation. Refer to *Setting Up for RS-232 Operation* on page 1–5 for detailed instructions that describe flow control configuration and use.

HARD flow control, the default method, uses the RTS (Request To Send) and CTS (Clear To Send) handshake lines. In this mode the remaining output line, DTR (Data Terminal Ready), is always asserted TRUE and input lines DCD (Data Carrier Detect) and DSR (Data Set Ready) are ignored.

SOFT flow control uses the CTRL-Q/CTRL-S method. Output lines RTS and DTR are always asserted TRUE and input lines DCD, CTS, and DSR are ignored. This method should not be specified for binary transfers such as waveforms or files.

NONE (no flow control) may also be specified. In this case the user is responsible to ensure that I/O buffers do not overflow. Note that both SOFT flow control or NONE (no flow control) allow the use of a 3-wire interface (GND, TXD, and RXD). In each case RTS and DTR are asserted TRUE and the CTS, DCD, and DSR inputs are ignored. Refer to *Appendix A* for more details on RS-232 connector wiring and the implementation of the interface standard by the 2714 or 2715.

Message Buffering (GPIB)

The 2714 or 2715 Spectrum Analyzer with Option 03 (GPIB) buffers each input message that it receives. Message processing begins as soon as messages are received by the 2714 or 2715; it does not wait for the message terminator. The 2714 or 2715 remains busy until it is done executing the commands in its input buffer, unless the process is stopped by the DCL (Device Clear) or SDC (Selected Device Clear) GPIB messages.

If an error is detected while transferring a command or query, the remainder of the message (up to the message terminator) is discarded.

Output data is ready following execution of each query and is passed to the 2714 or 2715’s output buffer prior to transmission over the bus. The spectrum analyzer begins to transmit an output message after it is addressed as a talker and the data is available. However, the response terminator is not sent until the command terminator is detected in case there are more queries in the input message.

Output continues from the 2714 or 2715 until the end of the information in the output buffer is reached, or until it is interrupted by a DCL (Device Clear), UNT (Untalk), or IFC (Interface Clear) GPIB message. If the 2714 or 2715 is interrupted before the buffer is cleared, output will resume if it is readdressed as a talker.

The buffer is cleared by the DCL or SDC messages. The 2714 or 2715 terminates the output according to the selected message terminator (EOI or CR/LF/EOI) unless it is interrupted.

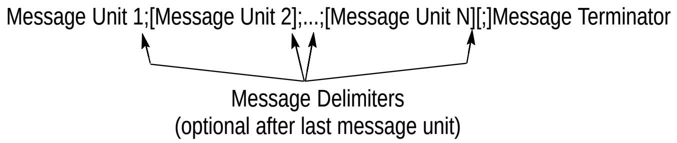
Message Format

Messages are formatted along structural lines. Each message consists of one or more message units separated or delimited by semicolons (;).

***NOTE**. Line feeds can be selected as message unit delimiters in GPIB responses — see the* `MSGblm` *command.*

Message

Each input or output message can be represented graphically as shown in this example:



Message Unit 1;[Message Unit 2];...;[Message Unit N][;]Message Terminator

Message Delimiters
(optional after last message unit)

Items enclosed in square brackets **[ ]** denote optional terms such as command headers and arguments. Each message unit consists of an individual command, query, or response with all necessary arguments and delimiters. Simple messages may consist of individual commands or queries as in these examples:

```
FREq 200MHZ
```

```
SAVe A:ON
```

```
CURve?
```

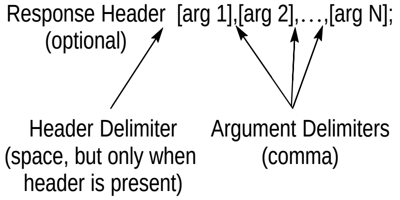
However, message units may be combined using message delimiters (;) to form more complex messages like this one:

```
FREq 200 MHZ;SAVe A:ON;CURve?
```

Notice that commands and queries can be mixed in a single message.

Commands

A command can be represented graphically as shown in this example:



Although multiple arguments are shown, most commands have only one argument. Following are several examples of specific commands:

```
FREq 5.5E+6

REFlvl -12 DBMV

VRTdsp LIN:100MV

RESbw 1.0M

SAVe A:ON,B:ON,C:OFF

TIMe 5 US
```

These examples illustrate several important characteristics of command formatting:

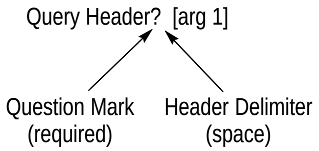
- A header delimiter (space) following the header is required.
- Three number formatting options are available. Arguments may be expressed as integers (-12 and 100), floating point numbers (1.0 and .035), or in scientific notation (5.5E+6).
- When no units are indicated, the appropriate units are inferred from the command header. For instance, `TIMe` implies seconds and `FREq` or `RESbw` imply Hz. Thus, 5.5E+6 in the `FREq` command implies 5.5×10^6 Hz or 5.5 MHz.
- A space between an argument and its units is optional. For example, `TIMe 5 US` and `TIMe 5US` are both valid commands.
- Shortened forms of units may be used (M instead of MHz in the `RESbw 1.0M` command). Only the first letter of the unit is read. The value and units represented by the letter depend on the command it is used with. For instance, M is interpreted as 10^6 Hz (MHz) when used within the `FREQ` command, but it represents 10^{-3} seconds (msec) when used within the `TIME` command.

  Commands that use a dB unit require the entire unit to avoid confusion between the various dB units. For instance, the command `REFlvl` must be followed by a complete unit such as dBm or dBmV. If only the letter "d" is entered, the computer is unable to determine the type of dB unit desired by the programmer.

- Linked arguments (VRTdsp) are always delimited by colons (:). Multiple arguments are always delimited by commas (,).

Queries

A query can be represented graphically as shown in this example:



Most queries recognized by the 2714 or 2715 have no arguments, but a few have one argument. There are no queries with multiple arguments. Following are several examples of queries to which the 2714 or 2715 will respond:

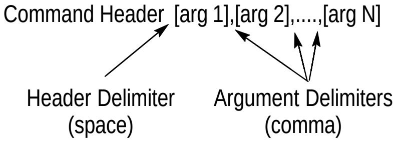
```
FREq?

REFlvl?

VRTdsp?

RESbw?

VIEw? A

TIMe?
```

These examples illustrate several characteristics of query formatting:

- A question mark (?) must follow the query header
- When an argument is used, it must be separated from the question mark by a space (as in VIEw? A)
- A command header can often be (but not always) turned into a query by adding a question mark (?)

Responses

A response can be represented graphically as shown in this example:


Command Header [arg 1],[arg 2],....,[arg N]
Header Delimiter (space)
Argument Delimiters (comma)

Headers are turned on and off with the `HDR` command. With the exceptions of the `SET?`, `PLOT?`, and `WAVfrm?` queries (which never produce response headers), response headers are optional. When `HDR` is `ON`, all responses (except `SET?`, `PLOT?`, and `WAVfrm?`) have headers.

No responses have headers when `HDR` is `OFF`. Further, when `HDR` is `OFF`, the link in the responses to marker queries with linked numeric arguments (such as `MAMpl?` or `MFReq?`) is also turned off. Response headers cannot be selectively suppressed unless `HDR OFF` is set before the response, and then `HDR` is set to `ON` after the response.

Responses always terminate with a semicolon (;) unless `MSGdlm` is set to `LF`, which causes responses to terminate in a line feed.

Following are examples of three queries and their resulting responses. The first line after each query is the response with the headers on (`HDR` is `ON`). The second line is the response to these queries with the headers off (`HDR` is `OFF`).

```
FREq?

    FREq 200.00E+6;

    200.00E+6;

VIEw?

    VIEw WATERFALL:OFF,A:ON,B:OFF,

        C:OFF,D:ON,MINUSA:OFF;

    WATERFALL:OFF,A:ON,B:OFF,

        C:OFF,D:ON,MINUSA:OFF;

MAMpl? DELTA

    MAMpl DELTA:18.5;

    18.5;
```

Most responses consist of an optional header and the response argument. However, responses such as the `WFMpre?` response have many arguments separated by commas (,). Others, including the response to `CURve?`, contain hundreds of data words in a single argument called a binary block. `SET?` is a special query that returns many arguments separated by semicolons (;) so the response can be read back to the spectrum analyzer as a series of commands.

Headers

The 2714 or 2715 understands uppercase letters, lowercase letters, or a mixture of upper and lowercase letters in command and query headers. Our examples use the long form of command and query headers. However, most (but not all) command and query headers can be written using only the first three letters. That is, `FRE` means the same as `FREq` and `FOF` means the same as `FOFfset`.

As you become more familiar with the commands and queries, you will find that the shortened forms are quicker to use. However, the long forms are easier to read and are always used by the instrument in response headers.

Throughout the remainder of this manual the long form for headers will be used. The required letters will be capital letters and the optional characters will be lowercase letters. These examples show how headers will appear in this manual:

```
FREq

VRTdsp

MAMpl
```

You can also use variations of the short and long header forms. `VRT`, `VRTd`, `VRTds`, and `VRTdsp` are all acceptable header forms for the vertical display command or query.

Space ( )

The space character is used to separate a command or response header from its first argument, or to separate the question mark (?) following a query header from its argument. The space is optional when there are no arguments.

Comma (,)

Commas are used to separate or delimit multiple arguments in commands and responses. They should not be used elsewhere.

Semicolon (;)

Semicolons are normally used to separate or delimit multiple message units in a single message. They may also (optionally) follow the last argument of a command or query. They should not be used elsewhere. The line feed character can be substituted for the semicolon.

Colon (:)

Colons are used to connect the two parts of a linked argument. They should not be used elsewhere.

Carriage Return and Line Feed (RS-232)

When a controller sends data to the 2714 or 2715, the spectrum analyzer interprets CR, LF or CR-LF as a message terminator. Messages sent by the spectrum analyzer over the RS-232 interface can be terminated in the following ways:

- By CR only (carriage return, ASCII 13)
- By LF only (line feed, ASCII 10)
- By CR-LF (carriage return followed by line feed)

Line Feed (GPIB)

The line feed character can be used instead of a semicolon (;) to delimit message units in a single message. The 2714 or 2715 will substitute line feeds for semicolons in its responses when `MSGdlm` is set to LF. Line feed can also be used as a message terminator with controllers that do not support the GPIB EOI protocol. These two uses are separate and not exclusive.

Functional Groups

Functional Groups

The instrument-specific commands and queries have been listed several different ways in this manual to make them more convenient to the user. The level of detail increases as you progress through the manual.

Table 3–1 provides a quick reference that contains all instrument-specific commands and queries. Tables 3–2 to 3–16 relate each remote command to a corresponding front panel control or menu. *Command and Query Definitions* provides a detailed description of each command and query. *Command and Query Definitions* includes all commands and queries available for the 2714 or 2715 listed in alphabetical order with a discussion of each, syntax examples, data formats, and other useful information.

The capital letters in a header indicate the minimum number of letters that must be supplied for the 2714 or 2715 to recognize the header. For instance, the query `ACQ?` would produce the same response as the query `ACQmode?`. The lowercase letters are optional, additional letters that may be used to clarify the meaning of the header. The spectrum analyzer accepts either upper or lowercase letters, but it WILL NOT accept headers that contain letters other than those indicated in the following tables. For instance, if the tables show a command in this form,

```
CMEas
```

you can enter the command in any combination of upper and lowercase letters as in these examples:

```
CME

CmEA

cMeas
```

However, attempts to enter the command using a different combination of letters, as in the following examples, will be ignored.

```
CMEASURE

cma
```

An SRQ and an error message will be generated. See *Status Reporting* for additional information.

The Command/Query List

Table 3–1 lists all the commands available for controlling the 2714 or 2715 Spectrum Analyzer. This table provides the correct form of each command and query header. Table 3–1 is a convenient reference for the mnemonics of each command or query for users who are already familiar with instrument functions.

Table 3–1: Commands and Queries

ACQmode
ACQmode?
AREs
AREs?
ARFatt
ARFatt?
ATBI?
ATHrhld
ATHrhld?
AVDest
AVDest?
AVG
AVG?
AVMode
AVMode?
AVNum
AVNum?
BTFreq
BTFreq?
BWMode
BWMode?
BWNum
BWNum?
BWResult?
CALsig
CALsig?
CATv
CATv?
CENsig
CFSF
CFSF?
CLOck
CLOck?
CLRKey
CLRMenu
CMEas
CNBw
CNBw?
CNMode
CNMode?
CNResult?
CNTtrak
CNTtrak?
CONTinue
COUnt?
CREs
CSFreq
CSFreq?
CSDefault
CSInt
CSInt?
CSNorm
CSNorm?
CSRFreq
CSRFreq?
CTFreq
CTFreq?
CTDefault
CRES?
CURve
CURve?
DATe
DATe?
DATime?
DEFMenu
DETector
DETector?
DIR?
DIScor
DIScor?
DLIne
DLIne?
DLLimit
DLLimit?
DLValue
DLValue?
DSRc
DSRc?
EOS
EOS?
ERAse
ERR?
EVEnt?
FCMode
FCMode?
FCOr?
FILE
FILE?
FINe
FINe?
FOFfset
FOFfset?
FOMode
FOMode?
FREq
FREq?
GRAt
GRAt?
GTL
HDR
HDR?
HELp?
HRAmpl
ICDefault
ICFreq
ICFreq?
ID?
INIT
KEY
KEY?
LRAmpl
MAMpl?
MARker
MARker?
MEMory?
MEXchg
MFReq
MFReq?
MHDest
MHDest?

Table 3–1: Commands and Queries (Cont.)

- MKTime
- MKTime?
- MLFtnxt
- MMAx
- MNHld
- MNHld?
- MPOs?
- MRGTnxt
- MSGdlm
- MSGdlm?
- MSTep
- MTUNE
- MVPos?
- MXHld
- MXHld?
- MXRlvl
- MXRlvl?
- MXSpn
- MXSpn?
- NNBw
- NNBw?
- NNMode
- NNMode?
- NNResult?
- NORM
- NORM?
- OBWMode
- OBWMode?
- OBWPcnt
- OBWPcnt?
- OBWResult?
- PKHeight
- PKHeight?
- PLLmode
- PLLmode?
- PLOT?
- POFset
- POFset?
- PRDouts?
- PREamp
- PREamp?
- PROTset
- PROTset?
- PSTep
- PTYpe
- PTYpe?
- RECall
- REDout
- REDout?
- REFlvl
- REFlvl?
- RESbw
- RESbw?
- RFAtt
- RFAtt?
- RLUnit
- RLUnit?
- ROFset
- ROFset?
- ROMode
- ROMode?
- RQS
- RQS?
- RS232
- RS232?
- RST
- RTIme
- RTIme?
- SAVe
- SAVe?
- SET?
- SGErr
- SGErr?
- SGSrch
- SGTrak
- SGTrak?
- SIGswp
- SIGswp?
- SPAn
- SPAn?
- SSBegin
- SSBegin?
- SSEnd
- SSEnd?
- SSResult?
- STByte?
- STEp
- STEp?
- STOre
- STPinc
- STPinc?
- STStop
- SURv
- SURv?
- TAMpl?
- TEXt
- TEXt?
- TFReq?
- THRhld
- THRhld?
- TIMe
- TIMe?
- TIMMode
- TIMMode?
- TITLe
- TITLe?
- TMOde
- TMOde?
- TOPsig
- TRIgger
- TRIgger?
- TTLMode
- TTLMode?
- TUNe
- TVLine
- TVLine?
- TVLMode
- TVLMode?
- TVLStd
- TVLStd?
- UDPDir?
- VDMode
- VDMode?
- VFEnab
- VFEnab?
- VFMode
- VFMode?
- VIDflt
- VIDflt?
- VIEw
- VIEw?
- VMAnttbl
- VMAnttbl?
- VMDEst
- VMDEst?
- VMDIst
- VMDIst?
- VMMkrunit
- VMMkrunit?
- VMOnitor
- VMOnitor?
- VPOlarity
- VPOlarity?
- VRTdsp
- VRTdsp?
- VSYnc
- VSYnc?
- WAIt
- WAVfrm?
- WFMpre
- WFMpre?
- ZERosp
- ZERosp?

Command And Query Functional Groups

Tables 3–2 through 3–16 show how the commands and queries available for programming the 2714 or 2715 correspond to the front panel controls and menu selections. An illustration of each front panel function block or menu is shown. Related commands are placed beside the feature that they control.

The functional groupings and menus appear in the following order:

- FREQ/MKRS function block
- MKR/FREQ Menu
- FREQUENCY, SPAN/DIV, and REF LEVEL function block
- VERT SCALE function block and PLOT and READOUT controls
- INPUT Menu
- SWP/TRIG Menu
- SWEEP and RES BW function blocks
- DISPLAY STORAGE function block
- DISPL Menu
- CATV/APPL Menu
- UTIL Menu
- DEMOD Menu
- Curve and Waveform commands
- System-related commands
- Miscellaneous commands

FREQ/MKRS Functional Block

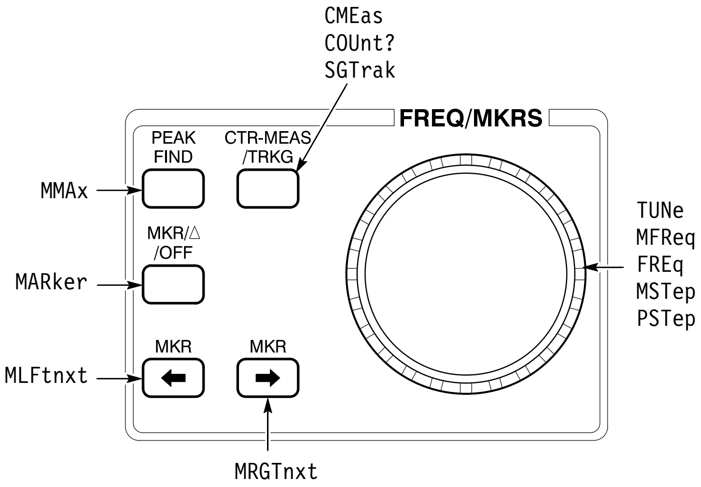
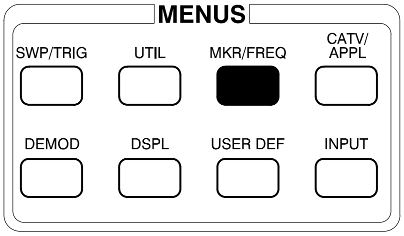
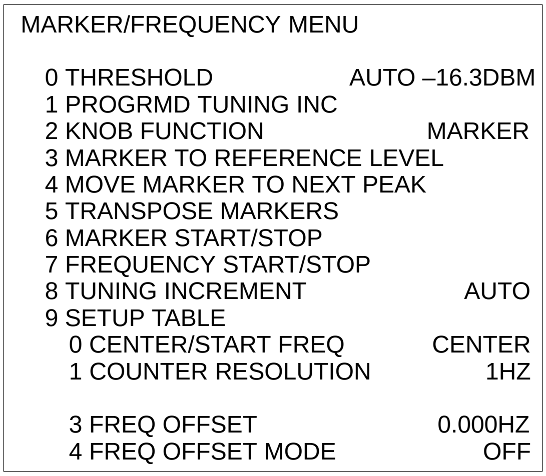


Table 3–2: FREQ/MKRS Front Panel Commands

Header	Function
CMEas	Perform a center measure.
COUnt?	What is the counter reading?
FREq	Set the start or center frequency.
FREq?	What is the start or center frequency?
MARker	Turn one or both markers on and off.
MARker?	What is the current marker status?
MFReq	Set the marker frequency.
MFReq?	What is the frequency of either or both markers?
MLFtnxt	Move the marker to the next signal peak left.
MMax	Move the marker to the highest data point on screen.
MRGTnxt	Move the marker to the next signal peak right.
MSTep	Equivalent to turning the knob one click to the left.
PSTep	Equivalent to turning the knob one click to the right.
SGTrak	Turn signal tracking on and off.
SGTrak?	Is signal tracking on or off?
TUNe	Change frequency.

MKR/FREQ Menu





```
MARKER/FREQUENCY MENU

  0 THRESHOLD              AUTO –16.3DBM   THRhld, ATHrhld
  1 PROGRMD TUNING INC                     STEp
  2 KNOB FUNCTION                 MARKER   TMOde
  3 MARKER TO REFERENCE LEVEL              TOPsig
  4 MOVE MARKER TO NEXT PEAK               LRAmpl, HRAmpl
  5 TRANSPOSE MARKERS                      MEXchg
  6 MARKER START/STOP                      STStop
  7 FREQUENCY START/STOP
  8 TUNING INCREMENT                AUTO   STPinc
  9 SETUP TABLE
    0 CENTER/START FREQ           CENTER   CFSF
    1 COUNTER RESOLUTION             1HZ   CREs, CNTrak

    3 FREQ OFFSET                0.000HZ   FOFfset
    4 FREQ OFFSET MODE               OFF   FOMode
```

MAMpl?, MKTime?, TAMpl?, and TFReq? return on-screen measurement parameters.

MPOs? and MVPos? have no visible effect when the 2714 or 2715 has an analog display (all Display Storage registers disabled).

Table 3–3: MKR/FREQ Menu Commands

Header	Function
ATHrhld	Turn the auto threshold on and off.
ATHrhld?	Is the auto threshold on or off?
CENsig	Set frequency to the marker frequency.
CFSF	Select center or start frequency.
CFSF?	Is center or start frequency being used?
CNTtrak	Turn counter on and off during signal track.
CNTtrak?	Is counter on or off during signal track?
CREs	Set the counter resolution.
CREs?	What is the counter resolution?

Table 3–3: MKR/FREQ Menu Commands (Cont.)

Header	Function
FOFfset	Set the frequency offset.
FOFfset?	What is the frequency offset?
FOMode	Turn frequency offset mode on and off.
FOMode?	Is frequency offset mode on or off?
HRAmpl	Move the marker to next higher amplitude peak.
LRAmpl	Move the marker to the next lower amplitude peak.
MAMpl?	What is the amplitude of either or both markers?
MEXchg	Exchange markers.
MKTime	Set the marker time in zero span mode.
MKTime?	What is the time of either or both markers?
MTUNE	Change marker frequency by a specified amount.
MPOs?	What is the horizontal position of either or both markers?
MVPos?	What is the vertical position of either or both markers?
STEp	Set the frequency increment step size?
STEp?	What is the frequency increment step size?
STPinc	Set the type of frequency increment.
STPinc?	What type of frequency increment is being used?
STStop	Set start/stop frequencies to marker frequencies.
TAMpl?	What is amplitude of tracked signal?
TFReq?	What is frequency of tracked signal?
THRhld	Replace the auto threshold with the specified value.
THRhld?	What is the threshold value?
TMOde	Select the knob function.
TMOde?	What is the knob function?
TOPsig	Change reference level to the marker amplitude.

FREQUENCY, SPAN/DIV, and REF LEVEL Function Block

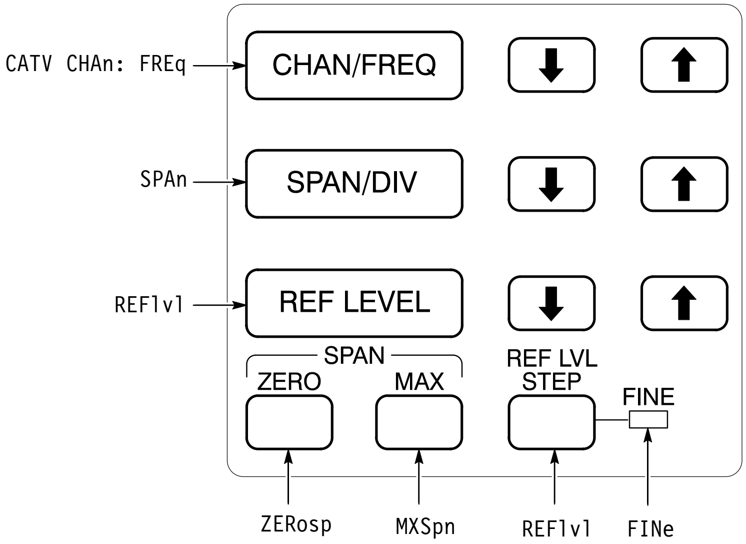


Table 3–4: FREQUENCY, SPAN/DIV and REF LEVEL Front Panel Commands

Header	Function
`CATV CHAn:`	Select specific channel <`num`> or next channel <`NEXt`> in sequence.
`FINe`	Select 1 dB or 10 dB reference level steps.
`FINe?`	Is the reference level step 1 dB or 10 dB?
`FREQ`	Set the center or start frequency.
`FREQ?`	What is the current center or start frequency?
`MXSpn`	Turn MAX SPAN mode on and off.
`MXSpn?`	Is MAX SPAN on or off?
`REFlvl`	Set/increment/decrement reference level.
`REFlvl?`	What is the reference level?
`SPAn`	Select the frequency span per division.
`SPAn?`	What is the frequency span?
`ZERosp`	Turn ZERO SPAN on and off.
`ZERosp?`	Is ZERO SPAN on or off?

VERT SCALE Function Block, PLOT, and READOUT Controls

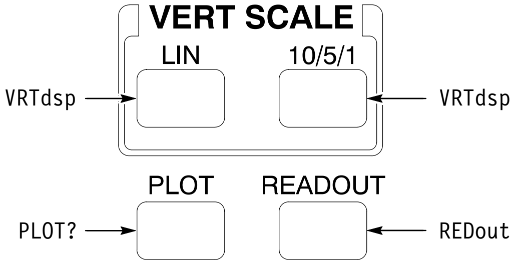
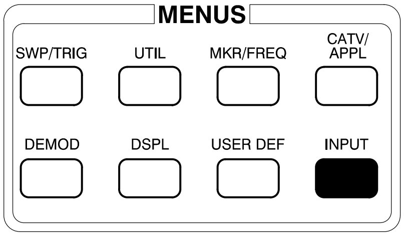
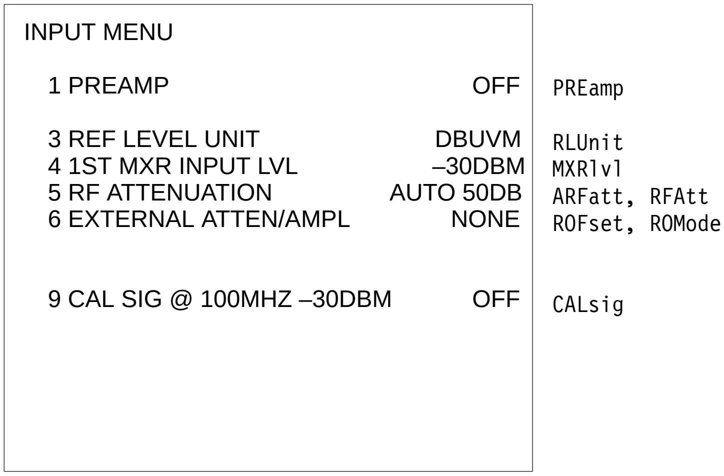


Table 3–5: VERT SCALE, PLOT, and READOUT Front Panel Commands

Header	Function
`PLOT?`	Return screen plot data from the spectrum analyzer.
`REDout`	Turn the on-screen readouts on or off.
`REDout?`	Are on-screen readouts on or off?
`VRTdsp`	Select the vertical scale factor.
`VRTdsp?`	What is the vertical scale factor?

INPUT Menu



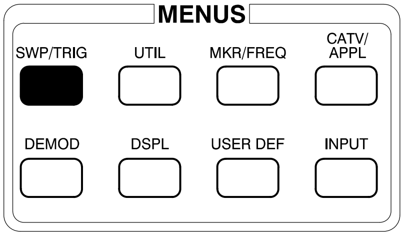


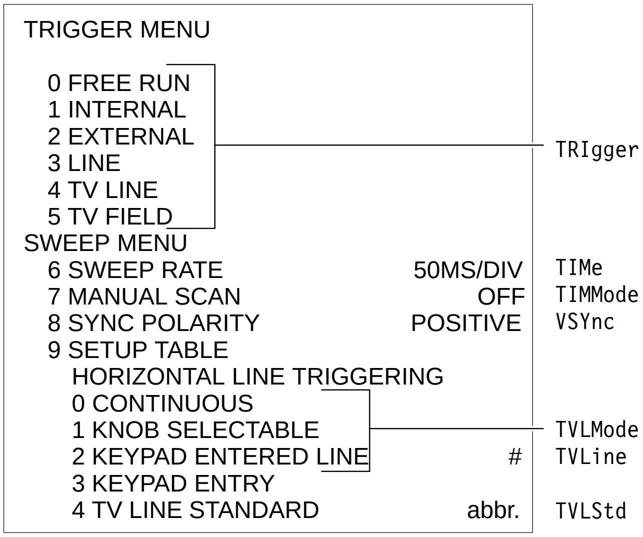
```
INPUT MENU

  1 PREAMP                                OFF     PREamp

  3 REF LEVEL UNIT                      DBUVM     RLUnit
  4 1ST MXR INPUT LVL                  –30DBM     MXRlvl
  5 RF ATTENUATION                 AUTO 50DB      ARFatt, RFAtt
  6 EXTERNAL ATTEN/AMPL                  NONE     ROFset, ROMode


  9 CAL SIG @ 100MHZ –30DBM               OFF     CALsig
```

Table 3–6: INPUT Menu Commands

Header	Function
ARFatt	Turn auto RF attenuation on and off.
ARFatt?	Is auto RF attenuation on or off?
ATBl?	Provide a listing of an antenna correction table.
CALsig	Turn the internal calibration signal on and off.
CALsig?	Is the internal calibration signal on or off?
MXRlvl	Select first mixer level.
MXRlvl?	What is first mixer level?
PREamp	Turn the preamp on and off.
PREamp?	Is the preamp on or off?
RFAtt	Set the RF attenuation to a specific value.
RFAtt?	What is the RF attenuation?
RLUnit	Select reference level unit.
RLUnit?	What is the reference level unit?
ROFset	Set the reference level offset and turn it on and off.
ROFset?	What is the reference level offset?
ROMode	Turn reference level offset mode on and off.
ROMode?	Is reference level offset mode on or off?
VMAnttbl	Select an antenna table.
VMAnttbl?	What antenna table is selected?
VMDIst	Select measurement distance in dBμV/m mode.
VMDIst?	What is the measurement distance?
VMDEst	Select destination register in dBμV/m mode.
VMDEst?	What is the destination register?
VMMkrunit	Select marker units of dBμV/m or Volts/m in dBμV/m mode.
VMMkrunit?	What is the marker unit in dBμV/m mode?

SWP/TRIG MENU





```
TRIGGER MENU

  0 FREE RUN
  1 INTERNAL
  2 EXTERNAL
  3 LINE
  4 TV LINE
  5 TV FIELD
SWEEP MENU
  6 SWEEP RATE                        50MS/DIV
  7 MANUAL SCAN                            OFF
  8 SYNC POLARITY                     POSITIVE
  9 SETUP TABLE
    HORIZONTAL LINE TRIGGERING
    0 CONTINUOUS
    1 KNOB SELECTABLE
    2 KEYPAD ENTERED LINE                    #
    3 KEYPAD ENTRY
    4 TV LINE STANDARD                   abbr.
```

Table 3–7: SWP/TRIG Menu Commands

Header	Function
`TIMe`	Set the sweep rate.
`TIMe?`	What is the sweep rate?
`TIMMode`	Select auto, manual, or programmed sweep.
`TIMMode?`	What is the sweep mode?
`TRIgger`	Select the trigger mode.
`TRIgger?`	What is the trigger mode?
`TVLine`	Select the number of the video raster line to trigger on when TV line triggering is selected.
`TVLine?`	What is the number of the TV line to trigger on?
`TVLMode`	Select continuous or programmed TV line trigger.
`TVLMode?`	Is continuous or programmed TV line trigger used?
`TVLStd`	Select TV standards used in various countries.
`TVLStd?`	What TV standard is being used?

Table 3–7: SWP/TRIG Menu Commands (Cont.)

Header	Function
VSYnc	Select positive or negative video sync polarity.
VSYnc?	Is positive or negative sync polarity being used?

SWEEP and RES BW Function Blocks

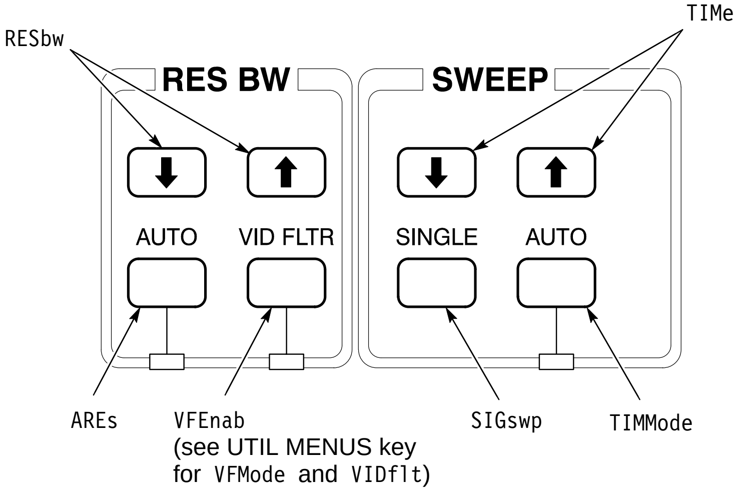


Table 3–8: SWEEP and RES BW Front Panel Commands

Header	Function
AREs	Turn AUTO resolution bandwidth on and off.
AREs?	Is AUTO resolution bandwidth on or off?
RESbw	Select/increment/decrement the resolution bandwidth.
RESbw?	What resolution bandwidth is selected?
SIGswp	Select and arm the single sweep mode.
SIGswp?	What is the status of the single sweep mode?
TIMe	Select/increment/decrement the sweep speed.
TIMe?	What is the sweep speed?
TIMMode	Select auto, manual, or programmed sweep mode.
TIMMode?	What sweep mode is selected?

DISPLAY STORAGE Function Block

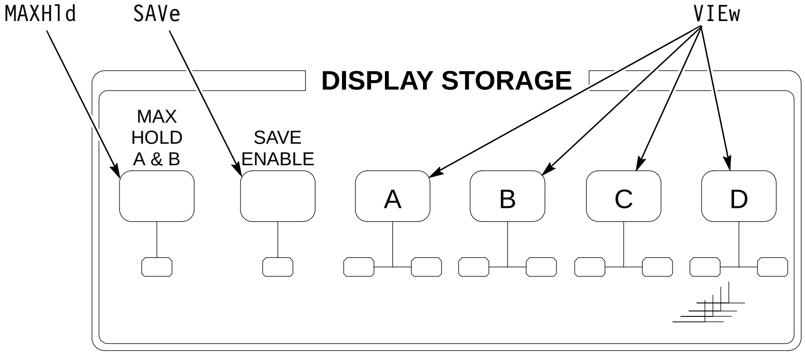
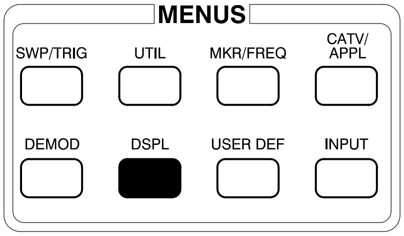
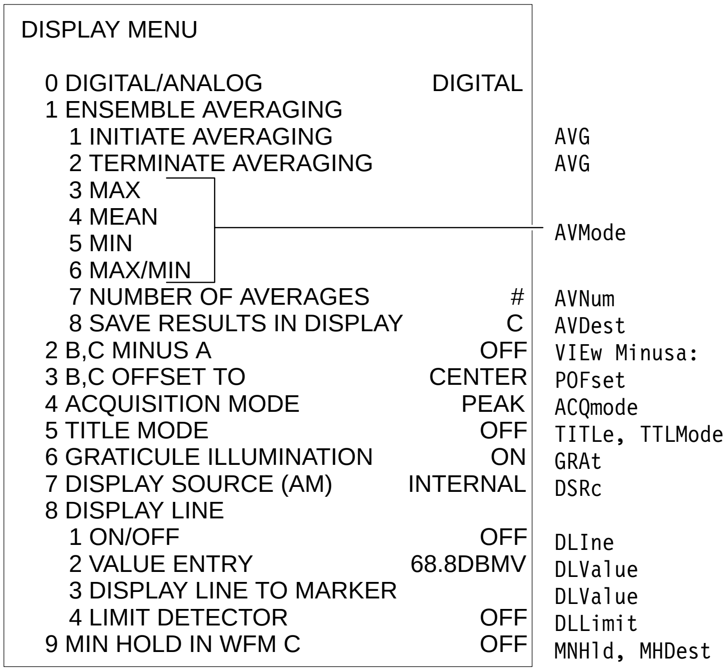


Table 3–9: DISPLAY STORAGE Front Panel Commands

Header	Function
`MXHld`	Turn max hold function on and off.
`MXHld?`	Is the max hold function on or off?
`SAVe`	Turn display storage on or off in any or all registers.
`SAVe?`	Is storage on or off in any or all registers?
`VIEw`	Turn display on and off in any or all registers. Also turns waterfall and B,C minus A modes on and off.
`VIEw?`	What is the display status of any or all registers?

DSPL Menu





```
DISPLAY MENU

  0 DIGITAL/ANALOG                     DIGITAL
  1 ENSEMBLE AVERAGING
    1 INITIATE AVERAGING                          AVG
    2 TERMINATE AVERAGING                         AVG
    3 MAX  ┐
    4 MEAN │
    5 MIN  ├──────────────────────────────────── AVMode
    6 MAX/MIN ┘
    7 NUMBER OF AVERAGES                     #    AVNum
    8 SAVE RESULTS IN DISPLAY                C    AVDest
  2 B,C MINUS A                            OFF    VIEw Minusa:
  3 B,C OFFSET TO                       CENTER    POFset
  4 ACQUISITION MODE                      PEAK    ACQmode
  5 TITLE MODE                             OFF    TITLe, TTLMode
  6 GRATICULE ILLUMINATION                  ON    GRAt
  7 DISPLAY SOURCE (AM)               INTERNAL    DSRc
  8 DISPLAY LINE
    1 ON/OFF                               OFF    DLIne
    2 VALUE ENTRY                     68.8DBMV    DLValue
    3 DISPLAY LINE TO MARKER                      DLValue
    4 LIMIT DETECTOR                       OFF    DLLimit
  9 MIN HOLD IN WFM C                      OFF    MNHld, MHDest
```

NOTE. *The TEXt command and TEXt? query, located in the following list, do not relate directly to a DSPL Menu feature.*

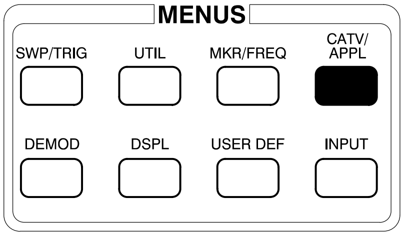
Table 3–10: DSPL Menu Commands

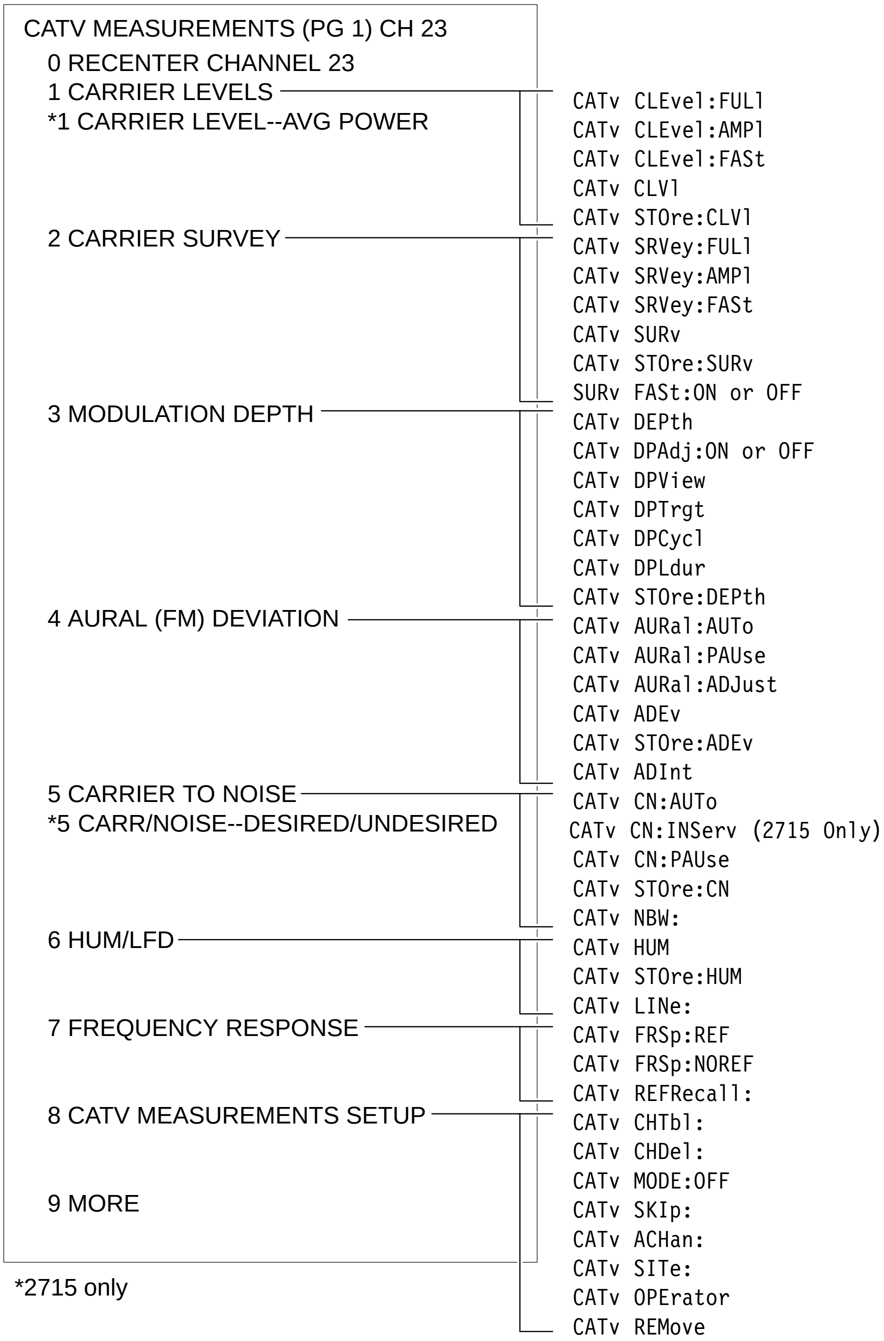
Header	Function
ACQmode	Select peak or max/min acquisition mode.
ACQmode?	What is the acquisition mode?
AVDest	Select destination register for ensemble averaging.
AVDest?	What is the destination register for averaging?
AVG	Turn ensemble averaging on and off.
AVG?	Is ensemble averaging on or off?
AVMode	Select the ensemble averaging mode.
AVMode?	What is the ensemble averaging mode?

Table 3–10: DSPL Menu Commands (Cont.)

Header	Function
AVNum	Select number of sweeps averaged.
AVNum?	What is the number of sweeps averaged?
DLIne	Turn the display line on and off.
DLIne?	Is the display line on or off?
DLLimit	Control the limit detector.
DLLimit?	What is the limit detector status?
DLValue	Set the display line value and turn it on.
DLValue?	What is the display line value?
DSRc	Select the detection mode.
DSRc?	What is the detection mode?
GRAt	Turn the graticule light on and off.
GRAt?	Is the graticule light on or off?
MNHld	Turn min hold function on and off.
MNHld?	Is the min hold function on or off?
MHDest	Select the min hold destination waveform.
MHDest	What is the min hold destination waveform?
POFset	Offset B,C-A mode to center or top of screen.
POFset?	Is B,C-A offset to top or center of screen?
TEXt	Display the indicated text on line 8 of display.
TEXt?	What is the text string being displayed?
TITLe	Display the indicated text as a title in title mode.
TITLe?	What is the title?
TTLMode	Turn title mode on and off.
TTLMode?	Is title mode on or off?
VIEw Minusa:	Turn B,C MINUS A mode on and off.

CATV/APPL Menu (CATV Mode Active)



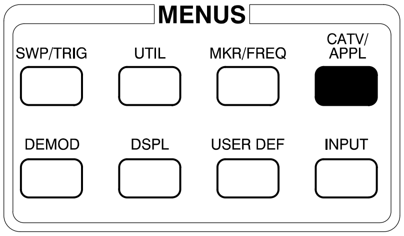


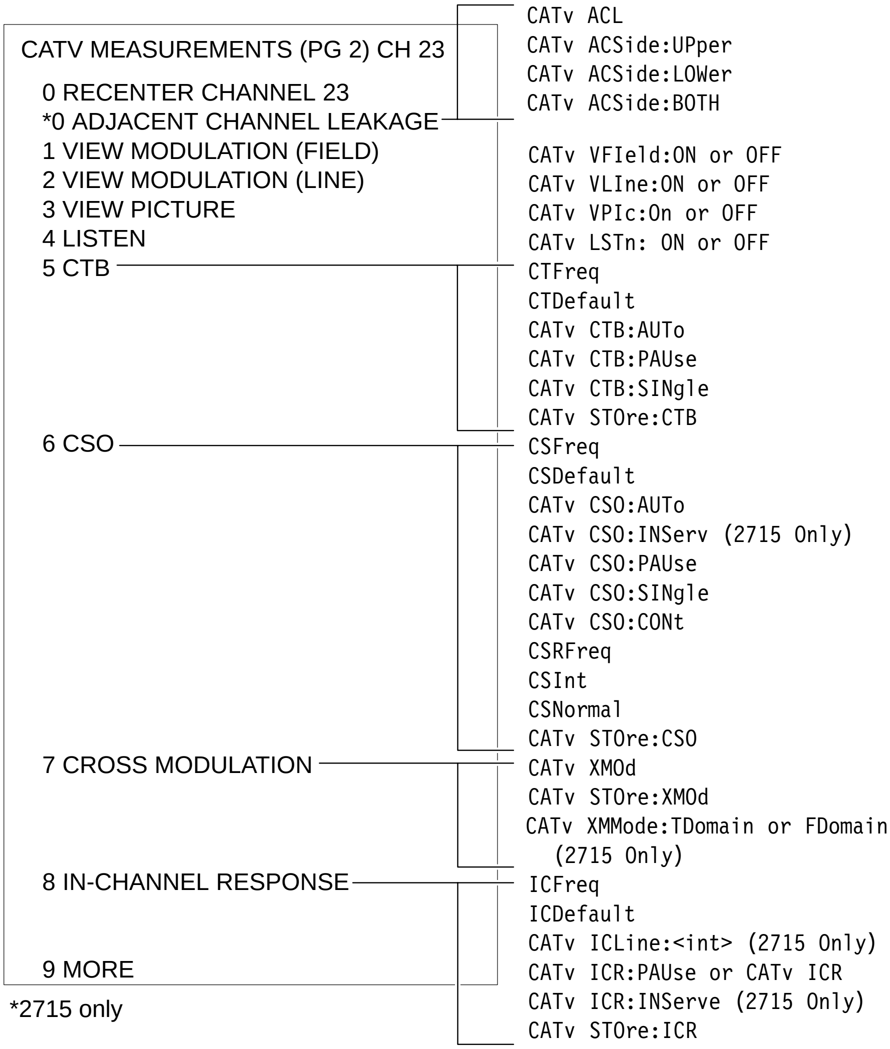
CATV MEASUREMENTS (PG 1) CH 23

0 RECENTER CHANNEL 23
1 CARRIER LEVELS
*1 CARRIER LEVEL--AVG POWER
2 CARRIER SURVEY
3 MODULATION DEPTH
4 AURAL (FM) DEVIATION
5 CARRIER TO NOISE
*5 CARR/NOISE--DESIRED/UNDESIRED
6 HUM/LFD
7 FREQUENCY RESPONSE
8 CATV MEASUREMENTS SETUP
9 MORE

```
CATv CLEvel:FULl
CATv CLEvel:AMPl
CATv CLEvel:FASt
CATv CLVl
CATv STOre:CLVl
CATv SRVey:FULl
CATv SRVey:AMPl
CATv SRVey:FASt
CATv SURv
CATv STOre:SURv
SURv FASt:ON or OFF
CATv DEPth
CATv DPAdj:ON or OFF
CATv DPView
CATv DPTrgt
CATv DPCycl
CATv DPLdur
CATv STOre:DEPth
CATv AURal:AUTo
CATv AURal:PAUse
CATv AURal:ADJust
CATv ADEv
CATv STOre:ADEv
CATv ADInt
CATv CN:AUTo
CATv CN:INServ (2715 Only)
CATv CN:PAUse
CATv STOre:CN
CATv NBW:
CATv HUM
CATv STOre:HUM
CATv LINe:
CATv FRSp:REF
CATv FRSp:NOREF
CATv REFRecall:
CATv CHTbl:
CATv CHDel:
CATv MODE:OFF
CATv SKIp:
CATv ACHan:
CATv SITe:
CATv OPErator
CATv REMove
```

*2715 only





```
CATV MEASUREMENTS (PG 2) CH 23
  0 RECENTER CHANNEL 23
  *0 ADJACENT CHANNEL LEAKAGE
  1 VIEW MODULATION (FIELD)
  2 VIEW MODULATION (LINE)
  3 VIEW PICTURE
  4 LISTEN
  5 CTB
  6 CSO
  7 CROSS MODULATION
  8 IN-CHANNEL RESPONSE
  9 MORE
```

*2715 only

*0 ADJACENT CHANNEL LEAKAGE	CATv ACL CATv ACSide:UPper CATv ACSide:LOWer CATv ACSide:BOTH
1 VIEW MODULATION (FIELD)	CATv VFIeld:ON or OFF
2 VIEW MODULATION (LINE)	CATv VLIne:ON or OFF
3 VIEW PICTURE	CATv VPIc:On or OFF
4 LISTEN	CATv LSTn: ON or OFF
5 CTB	CTFreq CTDefault CATv CTB:AUTo CATv CTB:PAUse CATv CTB:SINgle CATv STOre:CTB
6 CSO	CSFreq CSDefault CATv CSO:AUTo CATv CSO:INServ (2715 Only) CATv CSO:PAUse CATv CSO:SINgle CATv CSO:CONt CSRFreq CSInt CSNormal CATv STOre:CSO
7 CROSS MODULATION	CATv XMOd CATv STOre:XMOd CATv XMMode:TDomain or FDomain (2715 Only)
8 IN-CHANNEL RESPONSE	ICFreq ICDefault CATv ICLine:<int> (2715 Only) CATv ICR:PAUse or CATv ICR CATv ICR:INServe (2715 Only) CATv STOre:ICR

Table 3–11: CATV/APPL Menu Commands (CATV Mode Active)

Header	Function
`CATv ACHan:`	Turn “all channels” mode on or off.
`CATv? ACHan`	What is the on/off status of “all channels” mode?
`CATv ACL`	Performs the adjacent channel leakage test.[1]
`CATv? ACL`	What are the results of the adjacent channel leakage test?[1]
`CATv ACSide`	Specify which side (UPper, LOWer or BOTh) the adjacent channel leakage test will be run.[1]
`CATv? ACSide`	Will return which side of the adjacent channel leakage test will be measured.[1]
`CATv ADEv`	Perform the aural carrier deviation measurement in the current mode.
`CATv? ADEv`	What are the results of the last aural carrier deviation measurement?
`CATv ADInt:<num>`	Specify the test interval in minutes (10 sec to 24 hours) for the aural carrier deviation measurement.
`CATv? ADInt`	Will return the current test interval in minutes.
`CATv AURal:`	Specify the aural carrier deviation measurement mode and run the test.
`CATv? AURal`	What are the results of the last aural carrier deviation measurement?
`CATv CHAn:`	Set the channel number.
`CATv CHAn?`	What is the current channel number?
`CATv CHDel`	Delete the specified channel table.
`CATv CHTbl:`	Set the channel table to the one specified.
`CATv? CHTbl`	What is the current channel table?
`CATv CLEvel:`	Specify the carrier levels measurement mode and run the test.
`CATv CLVl`	Perform the carrier levels measurement in the current mode.
`CATv? CLVl`	What are the results of the last carrier levels measurement?
`CATv CN:`	Perform the specified carrier-to-noise measurement.
`CATv? CN`	What are the results of the last carrier-to-noise measurement?
`CATv CTB:`	Perform the specified CTB measurement.
`CATv? CTB`	What are the results of the last CTB measurement?
`CATv? CTDir`	Return the directory of all loaded channel tables.
`CATv CSO:`	Perform the specified CSO measurement.
`CATv? CSO`	What are the results of the last CSO measurement?
`CATv DEPth:`	Perform the depth of modulation measurement.
`CATv? DEPth`	What are the results of the last depth of modulation measurement?
`CATv DPAdj:`	Turn modulation depth adjustment mode on or off.

[1] **2715 only.**

Table 3–11: CATV/APPL Menu Commands (CATV Mode Active) (Cont.)

Header	Function
`CATv DPCycle:`	Set the cycle delay for the modulation depth adjustment mode.
`CATv? DPCycle`	Return the present cycle delay value for the modulation depth adjustment mode.
`CATv DPLdur:`	Set the target line duration for the modulation depth adjustment mode.
`CATv? DPLdur`	Return the present target line duration value for the modulation depth adjustment mode.
`CATv DPTrgt:`	Set the target line for the modulation depth adjustment mode.
`CATv? DPTrgt`	Return the present target line value for the modulation depth adjustment mode.
`CATv DPView:`	Set the view modulation mode for the modulation depth adjustment mode.
`CATv? DPView`	Return the present status (FIEld or LINe) of the view modulation mode.
`CATv FRSp:`	Perform the frequency response measurement.
`CATv? FRSp`	What are the results of the last frequency response measurement?

`CATv FULLADEv`
`CATv FULLCLVl`
`CATv FULLCN`
`CATv FULLCSO`
`CATV FULLCTB`
`CATv FULLDEPth`
`CATv FULLFRSp`
`CATv FULLHUM`
`CATv FULLICR`
`CATv FULLSURv`
`CATv FULLXMOD`
	What is the complete response, including header information, of the specified measurement?

`CATv HUM`	Perform the hum measurement.
`CATv? HUM`	What are the results of the last hum measurement?
`CATv ICLine:`	Set video line number that test signal resides on for the in-service test.[2]
`CATv? ICLine`	What is the video line number set for the in-service test?[2]
`CATv ICR:`	Perform the in-channel response measurement.
`CATv? ICR`	What are the results of the last in-channel response measurement?
`CATv LINe:`	Select 50 Hz or 60 Hz power line frequency.

[2] **2715 only.**

Table 3–11: CATV/APPL Menu Commands (CATV Mode Active) (Cont.)

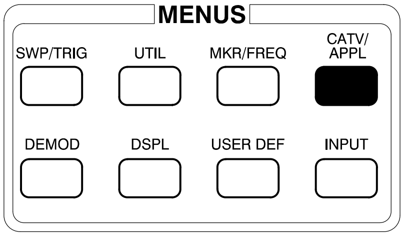
Header	Function
`CATv? LINe`	What is the current power line frequency?
`CATv LSTn:`	Turn the listen mode on or off.
`CATv? LSTn`	What is the on/off status of listen mode?
`CATv MODe:`	Turn CATV mode on or off.
`CATv? MODe`	What is the on/off status of CATV mode?
`CATv NBW`	Set the noise bandwidth used for carrier-to-noise.
`CATv? NBW`	Return the noise bandwidth used for carrier-to-noise.
`CATv OPErator`	Set the current operator as specified.
`CATv? OPErator`	Return the current operator string.
`CATv REFRecall:`	Recall the specified stored reference.
`CATv REMove`	Remove all CATV-related results from NVRAM.
`CATv SITe:`	Set the current site as specified.
`CATv? SITe`	Return the current site string.
`CATv SKIp:`	Turn skip mode on or off.
`CATv? SKIp`	What is the on/off status of “all channels” mode?
`CATv SRVey:`	Specify the carrier survey measurement mode, and run the test.
`CATv STOre:`	Store the results of the last measurement specified.
`CATv SURv`	Perform the carrier survey measurement in the current mode.
`CATv? SURv`	What are the results of the last carrier survey measurement?
`CATv VFIeld:`	Turn view modulation (field) mode on or off.
`CATv? VFIeld`	What is the on/off status of view modulation (field) mode?
`CATv VLIne:`	Turn view modulation (line) mode on or off.
`CATv? VLIne`	What is the on/off status of view modulation (line) mode?
`CATv VPIc:`	Turn view picture mode on or off.
`CATv? VPIc`	What is the on/off status of view picture mode?
`CATv XMMode:`	Select the measurement mode for the cross modulaltion measurement (either Time Domain or Frequency Domain).[3]
`CATv? XMMode`	What is the current cross modulation measurement mode?[3]
`CATv XMOd`	Perform the cross modulation measurement.
`CATv? XMOd`	What are the results of the last cross modulation measurement?
`CONT RST`	Provides continuous measurement for a period of 24 hours.

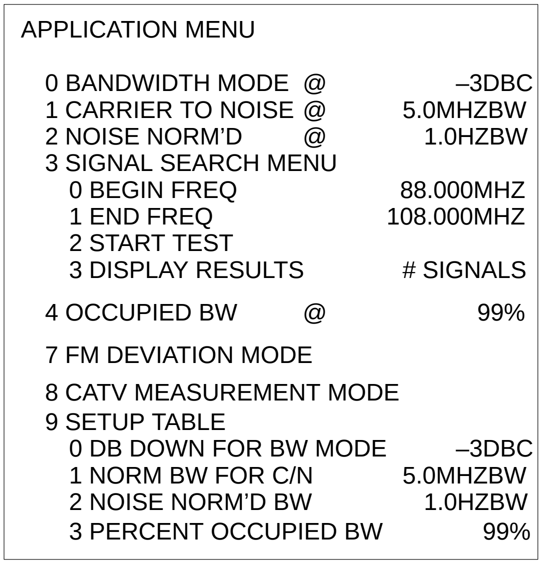
[3] **2715 only.**

Table 3–11: CATV/APPL Menu Commands (CATV Mode Active) (Cont.)

Header	Function
`CSFreq`	Define entries in the AUTO TEST LOCATIONS table used by the CSO test.
`CSFreq?`	Return the auto test table.
`CSDefault`	Default value.
`CTFreq`	Define entries in the AUTO TEST LOCATIONS table used by the CTB test.
`CTFreq?`	Return the auto test table.
`CTDefault`	Default value.
`FCOr?`	What are the frequency correction values?
`FCMode`	ON returns flatness correction values. OFF returns 0.0.
`FCMode?`	What is the status of FCMode (on or off)?
`ICFreq`	Define entries in the AUTO TEST LOCATIONS table used by the In-Channel Response test.
`ICFreq?`	Return the auto test table.
`ICDefault`	Recall the default values for the AUTO TEST LOCATIONS table used by the In-Channel Response test.
`SURv FASt:`	Turn the fast carrier survey mode on or off.
`SURv? FASt`	What is the on/off status of the fast carrier survey mode?

CATV/APPL Menu (CATV Mode Inactive)





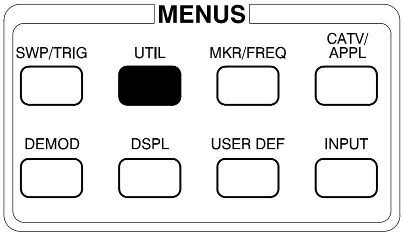
```
APPLICATION MENU

  0 BANDWIDTH MODE  @            –3DBC   BWMode
  1 CARRIER TO NOISE @          5.0MHZBW  CNMode
  2 NOISE NORM'D     @          1.0HZBW   NNMode
  3 SIGNAL SEARCH MENU
    0 BEGIN FREQ                88.000MHZ SSBegin
    1 END FREQ                 108.000MHZ SSEnd
    2 START TEST                          SGSrch
    3 DISPLAY RESULTS          # SIGNALS  SSResult?
  4 OCCUPIED BW      @             99%    OBWpcnt
  7 FM DEVIATION MODE
  8 CATV MEASUREMENT MODE                 CATv:ON
  9 SETUP TABLE
    0 DB DOWN FOR BW MODE        –3DBC    BWNum
    1 NORM BW FOR C/N          5.0MHZBW   CNBw
    2 NOISE NORM'D BW           1.0HZBW   NNBw
    3 PERCENT OCCUPIED BW          99%
```

***NOTE**. BWResult?, CNResult?, and NNResult? return results normally displayed on the screen.*

Table 3–12: CATV/APPL Menu Commands (CATV Mode Inactive)

Header	Function
BWMode	Turn bandwidth mode (BW)) on and off.
BWMode?	Is bandwidth (BW) mode on or off?
BWNum	Set the number of dB down for BW mode.
BWNum?	What is the dB down setting in BW mode?
BWResult?	What is the bandwidth at the specified dB down?
CATv	Set CATV control parameters.
CATv?	Return the CATV control parameter status.
CNBw	Set noise bandwidth for carrier-to-noise (C/N) mode.
CNBw?	What is the noise bandwidth in C/N mode?
CNMode	Turn carrier-to-noise mode on and off.
CNMode?	Is carrier-to-noise mode on or off?
CNResult?	What is the C/N ratio?
DSRc	Set the detection mode.
DSRc?	What is the detection mode?
NNBw	Set the noise bandwidth for normalized noise mode.
NNBw?	What is the noise bandwidth in normalized noise mode?
NNMode	Turn normalized noise mode on and off.
NNMode?	Is normalized noise mode on or off?
NNResult?	What is the normalized noise in the specified bandwidth?
OBWMode	Set occupied bandwidth mode to on, off or idle.
OBWMode?	Is the occupied bandwidth mode on, off or idle?
OBWpcnt	Set percent (1 to 99%) occupied bandwidth.
OBWpcnt?	Return the current occupied bandwidth percent.
OBWResult?	Return the results (Hz) of the most recent occupied bandwidth measurement.
SSBegin	Set the beginning signal search frequency.
SSBegin?	What is the beginning signal search frequency?
SSEnd	Set the ending signal search frequency.
SSEnd?	What is the ending signal search frequency?
SGSrch	Search for signals greater than threshold (THRhld) between beginning and ending search frequencies.
SSResult?	What is the result of the signal search?

UTIL Menu



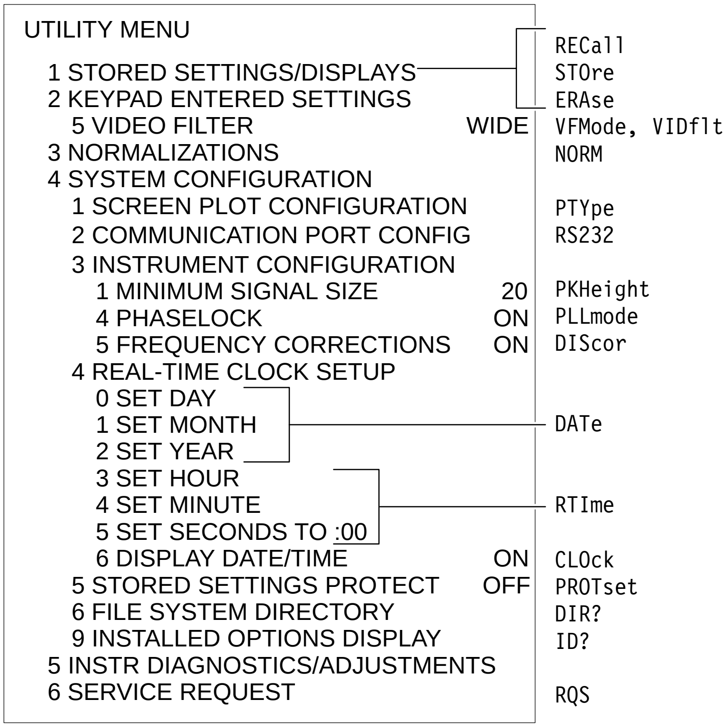


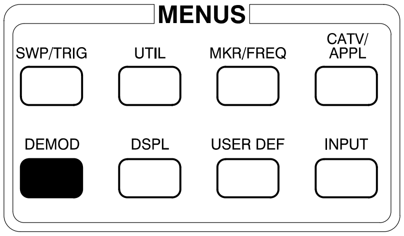
Table 3–13: UTIL Menu Commands

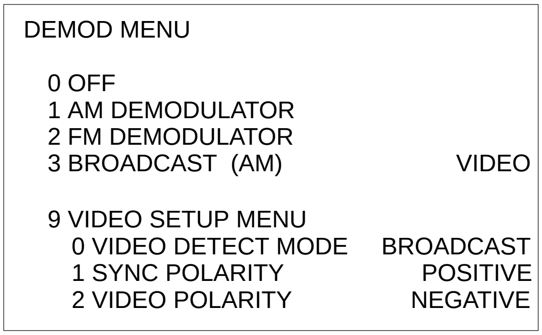
Header	Function
`DIR?`	Return a spectrum analyzer file system directory.
`DIScor`	Turn the frequency corrections on and off.
`DIScor?`	Are the frequency corrections on or off?
`DATe`	Set the real-time clock date.
`DATe?`	What is the real-time clock date?
`DATIme?`	What is the time of day?
`CLOck`	Turn the date and time display on or off.
`CLOck?`	Is the date and time display on or off?
`ERAse`	Erase the stored settings in a particular register.

Table 3–13: UTIL Menu Commands (Cont.)

Header	Function
ID?	List the 2714 or 2715's firmware version and installed options.
INIT	Reset to user-defined or factory power-up settings.
NORM	Carry out the indicated normalizations.
NORM?	Return a list of current normalization parameters.
PKHeight	Set the minimum signal height for marker functions.
PKHeight?	What is minimum signal height for marker functions?
PLLmode	Turn phase lock on and off.
PLLmode?	Is phase lock on or off?
PROTset	Turn stored settings files protection on and off.
PROTset?	Is stored settings files protection on or off?
PTYpe	Specify the plotter type for screen plots.
PTYpe?	What is the specified plotter type?
RECall	Recall a stored settings file.
RTIme	Set the real-time clock time.
RTIme?	What is the real-time clock time?
RS232	Set RS-232 communications parameters.
RS232?	Return all RS-232 communications parameters.
STOre	Store the current settings in a stored settings file.
VFMode	Selects auto or fixed video filter mode.
VFMode?	Is auto or fixed video filter mode selected?
VIDflt	Sets and turns the video filter on and off.
VIDflt?	What video filter is selected?

DEMOD Menu





```
DEMOD MENU

   0 OFF
   1 AM DEMODULATOR                              DETector
   2 FM DEMODULATOR
   3 BROADCAST  (AM)                   VIDEO     VMOnitor

   9 VIDEO SETUP MENU
      0 VIDEO DETECT MODE         BROADCAST     VDMode
      1 SYNC POLARITY              POSITIVE     VSYnc
      2 VIDEO POLARITY             NEGATIVE     VPOlarity
```

Table 3–14: DEMOD Menu Commands

Header	Function
`DETector`	Turn on/select which audio detector is used.
`DETector?`	Which audio detector is being used?
`VDMode`	Select broadcast or satellite video demodulation.
`VDMode?`	Is broadcast or satellite video demodulation used?
`VMOnitor`	Turn the video monitor on or off.
`VMOnitor`	Is the the video monitor on or off?
`VPOlarity`	Select positive or negative video polarity.
`VPOlarity?`	Is positive or negative video polarity selected?
`VSYnc`	Select positive or negative video sync polarity.
`VSYnc?`	Is positive or negative sync polarity being used?

Curve and Waveform Commands

Waveform transfers are not reflected on any menu or function block. They transfer data representing on-screen spectra and their formatting between the 2714 or 2715 and the controlling computer.

Table 3–15: Curve and Waveform Commands

Header	Function
`CURve`	Transfer waveform data to the register specified by `WFMpre`, using the encoding set by `WFMpre`.
`CURve?`	Transfer waveform data from the specified register or from the register set with `WFMpre`, using the encoding specified by the `WFMpre` command.
`WAVfrm?`	Same as `WFMPRE?` followed by `CURVE?`.
`WFMpre`	Specify source or destination register used for transferring waveform data with the `CURve` command or query. Also, specify the encoding to be used on the waveform data.
`WFMpre?`	Request the complete waveform preamble or ask which register and encoding are to be used for waveform transfers.

System-Related Commands

These commands and queries are independent of any spectrum analyzer menu or function block. They represent functions that affect the interaction of the spectrum analyzer and the GPIB controller or the RS-232 interface.

Table 3–16: System-Related Commands

Header	Function
`EOS`	Enable and disable SRQ on end-of-sweep.
`EOS?`	Is SRQ enabled or disabled at end-of-sweep?
`ERR?`	What is the error code?
`EVEnt?`	What is the event code?
`GTL`	Go to local operation.
`HDR`	Turn the response header on and off.
`HDR?`	Is the response header on or off?
`HELp?`	2714 or 2715 sends a list of valid GPIB command headers.
`MSGdlm`	Select semicolon or line feed as response delimiter.
`MSGdlm?`	What is the response delimiter?
`RQS`	Enable or disable SRQs (except power-on SRQ).
`RQS?`	Are SRQs enabled or disabled?
`RST`	Perform a Device Clear operation.

Table 3–16: System-Related Commands (Cont.)

Header	Function
SET?	What are the current spectrum analyzer settings?
SGErr	Enable/disable SRQ when marker cannot find a signal.
SGErr?	Is marker-cannot-find-signal SRQ enabled or disabled?
STByte?	Return GPIB serial poll status byte.
WAIt	Command the spectrum analyzer to wait for the end of sweep.

Miscellaneous Commands

The remaining commands include a set of miscellaneous commands and those which support on-screen menu definition and item selection. Refer to *Programming* for an example of how these commands are used to create an interactive menu on the 2714 or 2715's display screen.

Table 3–17: Miscellaneous Commands

Header	Function
CLRMenu	Clear the menu on the 2714 or 2715's screen.
CLRKey	Clear the last key pressed.
DEFMenu	Write a menu on the 2714 or 2715's screen.
FILE	Store a binary block under a given file name.
FILE?	Return the named file as a binary block.
KEY	Simulate pressing a key.
KEY?	Return the identity of the last key pressed.
MEMory?	How much NVRAM is free?
PRDouts?	Return the 2714 or 2715's on-screen readouts.
UDPDir?	Return list of currently loaded User Defined Programs.

Command/Query

Command and Query Definitions

This section contains an alphabetical listing of all instrument-specific commands and queries. The list defines each command or query. In addition, it contains all the information needed to send messages to the 2714 or 2715, or to interpret the responses from the 2714 or 2715.

Typographical Conventions

Each 2714 or 2715 command is discussed in the following format:

COMmand <arg> (if no argument is needed, <arg> is omitted)

Arguments: `Argument 1`, `argument 2`, ...

(If no argument is required, “None” is listed.)

Uppercase letters are required when entering the first three (sometimes four) letters of a command or query. Lowercase letters may be used for the remaining data if desired. Letters other than those shown in this manual will not be accepted by the 2714 or 2715.

Following each command is a general discussion of its arguments, specific precautions, and other important information.

Actual messages are shown in their correct syntax. When the number of possible messages is limited (such as commands that turn features on and off), all messages are shown as in the following example:

```
COMmand ON
```

```
COMmand OFF
```

Where there is a large range of arguments (such as numeric values), typical examples are shown as in this example:

`COMmand 10.5 kHz` *(for example)*

Typical examples are always followed by this phrase:

(for example)

Each query is discussed in the following format:

QUEry? <arg> (In most cases no argument is needed, and <arg> is omitted.)

Arguments: `Argument 1`, `argument 2`, ...

(If no argument is required, “None” is listed.)

Following each query is a general discussion of its arguments, specific precautions, and other important information. A detailed description of the response to the query is also provided.

The query is shown along with its arguments. The spectrum analyzer response is shown indented on the following line. The response is always shown assuming that `HDR ON` is selected as in the following example:

```
QUEry?

    QUERY ON

    QUERY OFF
```

All responses are shown when the number of possible responses is limited (such as queries that report the on/off status of features). When a large range of responses is possible (such as numeric values), typical examples are shown as in this example:

```
QUERY?

    QUERY 10.500E+3 (for example)
```

Typical examples are always followed by this phrase:

(for example)

List of Commands and Queries

The following list of commands and queries provides detailed information about the 2714 or 2715 instruction set. This section does not attempt to explain the operation of the spectrum analyzer. Refer to the user manual for the 2714 or 2715 for descriptions of the 2714 or 2715 or its features and functions.

ACQmode <arg>

Arguments: `MAXMin`, `PEAK`

This single-argument command designates the display storage acquisition mode.

```
ACQmode PEAK

ACQmode MAXMin
```

ACQmode?

Arguments: None

This simple query returns the currently selected acquisition mode.

```
ACQmode?

    ACQMODE PEAK

    ACQMODE MAXMIN
```

AREs <arg>

Arguments: `ON`, `OFF`

This single-argument command turns automatic selection of resolution bandwidth on and off.

```
AREs ON

AREs OFF
```

AREs?

Arguments: None

This simple query returns the status of automatic resolution bandwidth selection mode.

```
AREs?

    ARES ON

    ARES OFF
```

ARFatt <arg>

Arguments: ON, OFF

This single-argument command turns automatic selection of RF attenuation on and off. The attenuation, linear scale factor, and reference level may change when auto selection is turned on, but not when auto selection is turned off.

```
ARFatt ON

ARFatt OFF
```

ARFatt?

Arguments: None

This simple query returns the status of automatic RF attenuation selection.

```
ARFatt?

    ARFATT ON

    ARFATT OFF
```

ATBl? <arg>

Arguments: None, integer in range of 1 to 5

This is a query with one or no argument that returns a listing of the specified antenna table. The argument is the number of the antenna table to be listed. If a number outside the range is indicated, the last table in the range is returned (for instance, an argument of 6 returns table number 5, an argument of 0 returns table number 1). If no argument is specified, the currently selected table is returned.

```
ATBl? 3

    ATBL "ANTENNA 3

    Cal Distance = 3.0 Meters

    Frequency    Factor(dB)

    -------------------------

    100.0MHz         1.0

    200.0MHz         2.0

    300.0MHz         3.0

        :               :

      1.8GHz      18.0

    -------------------------

    ";
```

(for example)

ATHrhld <arg>

Arguments: ON, OFF

This single-argument command turns automatic selection of signal threshold on and off. The threshold may change when auto selection is turned on, but not when it is turned off.

```
ATHRHLD ON

ATHRHLD OFF
```

ATHrhld?

Arguments: None

This simple query returns the status of automatic signal threshold selection.

```
ATHrhld?

    ATHrhld ON

    ATHrhld OFF
```

AVDest <arg>

Arguments: A, B**,** C

This single-argument command designates the spectrum analyzer display register used as the destination for ensemble average and minimum hold operations. The destination register cannot be changed while a MIN Hold or ensemble average operation is in progress.

```
AVDest A

AVDest B

AVDest C
```

AVDest?

Arguments: None

This simple query returns the spectrum analyzer display register currently selected as the destination for MIN Hold and ensemble average functions.

```
AVDest?

    AVDEST A

    AVDEST B

    AVDEST C
```

AVG <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the currently selected ensemble averaging mode on and off. Ensemble averages will terminate after the requested number of sweeps are averaged, but `AVG OFF` is used to terminate a continuous average. Ensemble averaging cannot be turned on if the analog display mode is active, or if there is a destination register conflict.

```
AVG ON

AVG OFF
```

AVG?

Arguments: None

This simple query returns the on/off status of the currently selected ensemble averaging mode.

```
AVG?

    AVG ON

    AVG OFF
```

AVMode <arg>

Arguments: `MAX`, `MAXMin`, `MEAN`, `MIN`

This single-argument command designates the ensemble average mode.

```
AVMode MAX

AVMode MAXMin

AVMode MEAN

AVMode MIN
```

AVMode?

Arguments: None

This simple query returns the currently selected ensemble averaging mode.

```
AVMode?

    AVMODE MAX

    AVMODE MAXMIN

    AVMODE MEAN

    AVMODE MIN
```

AVNum <arg>

Arguments: Integer number in the range of 0 to 1024

This single-argument command designates the number of sweeps to be averaged by the currently selected ensemble averaging mode. If zero is specified, a continuous average is performed. The default is 16.

`AVNum 128` *(for example)*

AVNum?

Arguments: None

This simple query returns the integer number of sweeps the current ensemble averaging mode will average.

```
AVNum?
```

```
    AVNUM 128
```
(for example)

BWMode <arg>

Arguments: `ON`, `OFF`, `IDLE`

This single-argument command turns the spectrum analyzer bandwidth measurement mode on and off. When bandwidth mode is turned on, marker modes are turned off if they were previously enabled. Bandwidth mode is not allowed if the spectrum analyzer is in analog display mode or Video Monitor mode. Bandwidth mode cannot be enabled for waveforms in the D-register if waterfall mode is enabled.

```
BWMode ON

BWMode OFF

BWMode IDLE
```

`BWMode IDLE` has the same effect as `BWMode ON`.

BWMode?

Arguments: None

This simple query returns the status of the bandwidth measurement mode.

```
BWMode?

    BWMODE ON

    BWMODE OFF

    BWMODE IDLE
```

BWNum <arg>

Arguments: Integer in the range –1 to –70

This single-argument command specifies the integer number of decibels (dB) below the signal peak at which the bandwidth measurement mode measures bandwidth. Units are not allowed and dBc units are assumed. Non-integer values are truncated.

`BWNum -20` *(for example)*

BWNum?

Arguments: None

This simple query returns the integer value of decibels (dB) below the peak at which a signal's bandwidth will be measured by the spectrum analyzer's bandwidth measurement mode.

```
BWNum?

    BWNUM -20
```

(for example)

BWResult?

Arguments: None

This simple query returns the result of the most recent bandwidth measurement in hertz (using the spectrum analyzer's bandwidth measurement feature). The result is updated at the end of the current sweep if bandwidth mode is not idle.

```
BWResult?

    BWRESULT 5.238E+3
```

(for example)

CALSig <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the calibration signal on and off. The RF input signal is disconnected when the calibration signal is turned on.

```
CALSig ON

CALSig OFF
```

CALSig?

Arguments: None

This simple query returns the on/off status of the calibration signal.

```
CALSig?

    CALSIG OFF

    CALSIG ON
```

CATv <arg>

Arguments: See Table 4–1

This multi-argument command configures CATV measurements. Commands are implemented by entering the `CATv` command followed by the desired CATV function. For example, the command

```
CATv MODe:ON
```

enables the CATV mode.

When running a CATV measurement routine (such as `CTB:AUTo`), the 2714 or 2715 pauses when prompts appear on its screen. The pause generates event code 403, which must be cleared before the measurement will continue. A pause may be cleared by any of these methods:

- `CONTinue` resumes the test
- `RST` stops the test
- For RS-232, assert a break condition
- For GPIB, send a DCL command

NOTE. *The 2714 or 2715 is paused during measurement routines for various reasons, such as changing signal connections. To ensure accurate test results, carry out the operation indicated by the screen prompt before continuing program execution.*

Descriptions of the following CATV-related commands can be found later in this section (listed in alphabetical order):

```
CTFreq       ICFreq
CTFreq?      ICFreq?
CTDefault    ICDefault
CSFreq       SURv (FAST:)
CSFreq?      SURv?
CSDefault
```

Table 4–1: CATv Command Arguments

Argument	Function
`ACHan:ON`	Turn all channels mode on. Skip status will be ignored.
`ACHan:OFF`	Turn all channels mode off. Skip status will be honored.
`ACL`[1]	Performs the adjacent channel leakage test.
`ACSide:UPper`[1]	Runs the adjacent channel leakage test on the higher frequency channel.
`ACSide:LOWer`[1]	Runs the adjacent channel leakage test on the lower frequency channel.
`ACSide:BOTh`[1]	Runs the adjacent channel leakage test on both the higher and lower frequency channels.
`ADEv`	Perform the aural carrier deviation measurement in the current mode. This only works on CATV or FM channel types; otherwise event code 900 (FUNC NOT AVAILABLE ON THIS CHAN TYPE) is generated.
`ADInt:<num>`	Where <num> is a number from 10/60 to 1440 specifying the duration of the aural deviation test interval in minutes (from 10 sec. to 24 hours).
`AURal:ADJust`	Will enter the aural deviation “real time” ADJUSTMENT mode. Use RST, RS-232 BREAK or GPIB DCL to leave this mode. Measurement results will be discarded.
`AURal:AUTo`	Will run the aural deviation test in AUTO mode. While the test is running, CONtinue will cause the test to terminate and compute and save the measurement to that point. RST, RS-232 BREAK or GPIB DCL commands will cause the measurement to abort with the results discarded.
`AURal:PAUse`	Will run the aural deviation test in INTERACTIVE mode. While the test is running, CONtinue will cause the test to terminate and compute and save the measurement to that point. RST, RS-232 BREAK or GPIB DCL commands will cause the measurement to abort with the results discarded. Event code 403 (USER REQUEST OR CATV PROMPT) is generated after the measurement is complete. Use `CONTinue` to resume execution, or terminate the measurement with the `RST` command, an RS-232 BREAK, or a GPIB `DCL` command.
`CHAn:NEXt`	Go to the next entry in the channel table. Skip status is honored unless all channels mode is active. Channel number wraps to the beginning if at the end of the table.
`CHAn:<num>`	Sets the current channel number to <num> and installs the base settings for this channel. Channel numbers from 0 to 999 are accepted. Skip status is always ignored. Event code 801 and the message RANGE ERROR is returned if the channel is not defined in the current channel table. The channel number remains unchanged under these circumstances.

[1] **2715 only.**

Table 4–1: CATv Command Arguments (Cont.)

Argument	Function
`CHAn:PREv`	Go to the previous entry in the channel table. Skip status is honored unless all channels mode is active. Channel number wraps to the end if at the beginning of the table.
`CHDel:`	The channel table <`name`> is deleted from the instrument. <`name`> must be delimited with quotes (”). Event code 738 (STORAGE REGISTER EMPTY) is returned if the named channel is not found. The built-in tables STD, HRC, and IRC are always available. When deleted, these tables are removed from nonvolatile memory. However, they always appear within the `CTDir?` query response because they are constructed from permanent ROM-based data.
`CHTbl:<name>`	The current channel table is set to the specified `name`. The `name` must be delimited with quotes (”). Event code 777 (DEFAULT DATA LOADED) is returned if there is no channel with the specified `name`. Under these conditions the channel table is changed to the default (STD) table. The channel number is changed to the number that was active the last time this table was selected, or to the first channel for new tables.

`CLEvel:FULl`
`CLEvel:AMPl`
`CLEvel:FASt`
	Perform the carrier level measurement in the following manner:
	CLEvel:FULl – set mode to ACCURATE FREQUENCY AND AMPL and run test.
	CLEvel:AMPl – set mode to ACCURATE AMPLITUDE ONLY and run test.
	CLEVel:FAST – set mode to FAST AMPLITUDE ONLY and run test.
	If the current channel is identified in the channel table as DIGITAL, then the accuracy modes are irrelevant. The instrument will run the average power measurement.

`CLVl`	Perform the carrier levels measurement in the current mode. If the current channel is identified in the channel table as DIGITAL, then the accuracy modes are irrelevant. The instrument will run the average power measurement.
`CN:AUTo`	Perform the fully automatic carrier-to-noise measurement.
`CN:INServ`	Sets the carrier-to-noise mode to Auto and runs an in-service test.[1]
`CN:PAUse`	Perform the semiautomatic carrier-to-noise measurement. This command adds a pause to the fully automatic carrier-to-noise measurement just before the noise is measured. You may change equipment setup, but you cannot adjust the 2714 or 2715 controls. The measurement proceeds after you respond to the prompt. If the current channel is identified in the channel table as DIGITAL, then CN:PAUse will run the desired-to-undesired measurement.
`CSO:AUTo`	Perform the automatic CSO measurement.

[1] **2715 only.**

Table 4–1: CATv Command Arguments (Cont.)

Argument	Function
`CSO:CONt`	Perform the continuous CSO measurement. Event code 909 (NO BEAT FREQUENCIES ARE DEFINED) is returned if all beat frequencies have been deleted from the setup table.
`CSO:INServ`	Sets the composite second order (CSO) mode to Auto and runs an in-service test.[1]
`CSO:PAUse`	Perform the semi-automatic CSO measurement. In this mode the 2714 or 2715 pauses after measuring the carrier level. It then issues event 401 (prompts user to turn off carrier or modulation). A `CONT` command is then required to continue the measurement sequence. RST will abort the measurement.
`CSO:SINgle`	Perform the single-sweep version of the CSO measurement, which minimizes carrier-off time. In this mode the 2714 or 2715 pauses and generates event code 403 (USER REQUEST OR CATV PROMPT) after measuring the carrier. The user is then prompted to turn the carrier off. A single sweep is then performed between the lowest and highest frequencies in the auto test table (absolute values are ignored). Another event code 403 is generated to prompt the user to turn the carrier on. The measured results are the highest amplitudes detected within a ±50 kHz window around each frequency specified in the auto test table.
`CTB:AUTo`	Perform the automatic CTB measurement.
`CTB:PAUse`	Perform the semi-automatic CTB measurement. In this mode the 2714 or 2715 pauses after measuring the carrier level. It then issues event 401 (prompts user to turn off carrier or modulation). A `CONT` command is then required to continue the measurement sequence.
`CTB:SINgle`	Perform the single-sweep version of the CTB measurement, which minimizes carrier-off time. In this mode the 2714 or 2715 pauses and generates event code 403 (USER REQUEST OR CATV PROMPT) after measuring the carrier. The user is then prompted to turn the carrier off. A single sweep is then performed between the lowest and highest frequencies in the auto test table (absolute values are ignored). Another event code 403 is generated to prompt the user to turn the carrier on. The measured results are the highest amplitudes detected within a ±50 kHz window around each frequency specified in the auto test table. RST will abort the measurement.
`DEPth`	Perform the depth of modulation measurement.
`DPAdj`	Invoke or cancel the modulation depth adjustment mode. The selections are ON or OFF. KEY TERMW and KEY TERMZ can also be used to cancel the mode.
`DPCycl`	Sets up the cycle delay for the modulation depth adjustment mode. The range of values is 0 to 30,000.
`DPLdur:<num>`	Sets up the target line duration for the modulation depth adjustment mode. The range of values is 0.0 to 100.0%.

Table 4–1: CATv Command Arguments (Cont.)

Argument	Function
DPTrgt	Sets up the target line for the modulation depth adjustment mode. The range of values is 0.0 to 100.0%
DPView	Sets up the view modulation mode for the modulation depth adjustment mode. The selections are either FIEld or LINe.
FRSp:REF (or NORef)	Perform the frequency response measurement.
HUM	Perform the hum measurement.
ICLine:<int>	Sets the video line number on which the test signal for the in-service, in-channel response test resides. The argument can be an integer from 1 to 1023 or a –1. A value of –1 designates that no test signal exists so the in-service test will abort.[1]
ICR	Perform the in-channel response measurement.
ICR:INServ	Sets ICR mode to auto and runs an in-sevice test.[1]
ICR:PAUse	Equivalent to the ICR listed previously; perform the in-channel response measurement.
LINe:50	Select 50 Hz power line frequency.
LINe:60	Select 60 Hz power line frequency.
LSTn:OFF	Turn listen mode off.
LSTn:ON	Set listen mode to on so you can listen to demodulated audio.
MODe:OFF	Turn CATV mode off.
MODe:ON	Turn CATV mode on. The 2714 or 2715 will tune to the last active channel and install the base settings.
NBW:<num>	Set the noise bandwidth used for carrier-to-noise measurements to <num>. The range is 1 Hz to 1800 MHz. 4 MHz is the default value.
OPEra- tor:<string>	The current operator is set to <string> delimited by quotes (”). Only the first 28 characters are saved.
REFRecall:<num>	Recall the stored reference <num>.
REMove	Remove all CATV-related results from NVRAM.
SITe:<string>	The current site is set to <string> delimited by quotes (""). Only the first 28 characters are saved.
SKIp:OFF	Turn skip status OFF for the current channel.
SKIp:ON	Set skip status ON for the current channel.

[1] **2715 only.**

Table 4–1: CATv Command Arguments (Cont.)

Argument	Function
STOre:ADEv STOre:CLVl STOre:CN STOre:CSO STOre:CTB STOre:DEPth STOre:FRSp STOre:HUM STOre:ICR STOre:SURv STOre:XMOd	Store the results of the last measurement in NVRAM. For example, the command STOre:CN stores the results of the last carrier-to-noise measurement.
SRVey:FULl SRVey:AMPl SRVey:FASt	Perform the carrier survey measurement in the following manner: SRVey:FULl – set the mode to ACCURATE FREQUENCY AND AMPL and run the test SRVey:AMPl – set the mode to ACCURATE AMPLITUDE ONLY and runt the test SRVey:FASt – set the mode to FAST AMPLITUDE ONLY and run the test
SURv	Perform the carrier survey measurement in the current mode.
VFIeld:OFF	Set view modulation (field) mode to OFF.
VFIeld:ON	Set view modulation (field) mode to ON. This mode allows you to view base-band video in the time domain (as an oscilloscope display). Triggering and sweep speed are automatically set to display one complete video field.
VLIne:OFF	Set view modulation (line) mode to OFF.
VLIne:ON	Set view modulation (line) mode to ON. This mode allows you to view base-band video in the time domain (as an oscilloscope display). The knob is assigned to select which video line (nearest left edge of screen) triggers the display. The default line is the VITS line, or line 17 if no VITS line is identified by the channel table.
VPIc:OFF	Set view picture mode to OFF.
VPIc:ON	Set view picture mode to ON. This mode allows you to view a demodulated television picture on the 2714 or 2715's screen.
XMMode:FDomain	Sets the measurement mode for the cross modulation measurement to the frequency domain method.[1]
XMMode:TDomain	Sets the measurement mode for the cross modulation measurment to the time domain (NCTA) method.[1]
XMOd	Perform the cross-modulation measurement.

[1] **2715 only.**

CATv? <arg>

Arguments: See Table 4–2

This multi-argument query returns the status of various CATV measurement parameters. Queries are implemented by entering the `CATv?` query followed by the desired CATV function. For example, the query

```
CATv? CHAn
```

returns the current channel number.

Table 4–2: CATv? Query Arguments

Argument	Function
ACHan	Returns `ACHAN:ON` or `ACHAN:OFF`.
`ACL`[1]	Returns the results of the adjacent channel leakage test.
`ACSide`[1]	Returns where the adjacent channel leakage test will be performed in relation to the adjacent channel (UPper, LOWer, or BOTh).
ADEv	Returns the results of the last aural carrier deviation measurement as `CATV ADEV:<nr3>`, in Hz. See `FULLADEv`.
ADInt	Returns an NR2 representing the current aural deviation test interval in minutes.
AURal	Returns the results of the last aural carrier deviation measurement as `CATV AURal:<nr3>`, in Hz. See `FULLADEv`.
CHAn	Returns the current channel number as `CATV CHAN:<nr1>`.
CHTbl	Returns the current channel table as `CATV CHTBL:<string>`, where `<string>` is the name of the current table.
CLEvel	Returns the results of the last carrier levels measurement as `CATV CLEvel:<nr2>,<nr3>,<nr2>,<nr3>,<nr2>,<nr3>`. The six results, listed in order called, are visual carrier level in dBmV, visual carrier frequency in Hz, 1st aural carrier level relative to visual carrier in dBc, 1st aural carrier frequency relative to visual carrier in Hz, 2nd aural carrier level relative to visual carrier in dBc, and 2nd aural carrier frequency relative to visual carrier in Hz. If a result is not actually measured, 1.0 E 38 is returned in its place. See `FULLCLVl`.
CLVl	Returns the results of the last carrier levels measurement as `CATV CLVl:<nr2>,<nr3>,<nr2>,<nr3>,<nr2>,<nr3>`. The six results, listed in order called, are visual carrier level in dBmV, visual carrier frequency in Hz, 1st aural carrier level relative to visual carrier in dBc, 1st aural carrier frequency relative to visual carrier in Hz, 2nd aural carrier level relative to visual carrier in dBc, and 2nd aural carrier frequency relative to visual carrier in Hz. If a result is not actually measured, 1.0 E 38 is returned in its place. See `FULLCLVl`.

[1] **2715 only.**

Table 4–2: CATv? Query Arguments (Cont.)

Argument	Function
CN	Returns the results of the last carrier-to-noise measurement as CATV CN: <nr2>, <string>, <nr3>. The first result is the carrier-to-noise ratio in dB. The second result is the string "REL", which means the measurement was made at some frequency relative to the visual carrier. The third result is the frequency offset from the visual carrier at which the measurement was made. Digital channels return 1.0E38 as the frequency. See FULLCN.
CSO or CTB	Returns the results of the last CSO (or CTB) measurement as `CATV CSO` (or `CTB`)`:<count>,<ampl1>,<freq1>... etc.` `<count>` is the number of measurements performed, from 1 to 5. Each measurement has a corresponding amplitude/frequency pair. The format is NR2 for amplitude (in dBc units) and NR3 for frequency (in Hz). See `FULLCSO` (or `FULLCTB`).
CTDir	Returns the directory of all loaded channel tables in the form `CATV CTDIR:"<index1>", "<table name 1>","<index2>", "<table name 2>",...,.` `<index>` is the number used in the file name for GPIB/RS-232 access, and the location within the table channel menu. `<table name>` is the name displayed in the channel table menu.
DEPth	Returns the results of the last depth of modulation measurement as `CATV DEPTH:<nr2>`, expressed as a percentage. See `FULLDEPth`.
DPCycl	Return the present cycle delay value for the modulation depth adjustment mode.
DPLdur	Return the present target line duration value for the modulation depth adjustment mode.
DPTrgt	Return the present target line value for the modulation depth adjustment mode.
DPView	Return the present status (FIEld or LINe) of the view modulation mode.
FULLADEv FULLCLVl FULLCN FULLCSO FULLCTB FULLDEPth FULLFRSp FULLHUM FULLICR FULLSURv FULLXMOD	These queries return the complete response of the related parameter. This includes header information in addition to data returned by the short form of the query. For example, the query `CATv? FULLDEPth` returns `CATV FULLDEPTH:<header>,<nr2>.` The `<header>` information is divided into fields separated by commas (,). The fields appear in the order shown below: **Field** — **Description** 1. `<table>` — Quoted string naming the channel table 2. `<chan>` — Channel number as nr1 3. `<datime>` — Date and time as a quoted string 4. `<site>` — Site as a quoted string 5. `<operator>` — Operator as a quoted string

Table 4–2: CATv? Query Arguments (Cont.)

Argument	Function
HUM	Returns the results of the last hum measurement as CATV HUM:<nr2>,<nr2>,<nr2>. The three results, listed in order, are total hum, fundamental, and 2nd harmonic. All are stated as percentages of the carrier. The fundamental is either 50 Hz or 60 Hz, determined by the CATV LINe command.
ICLine	Returns the video line number on which the test signal for the in-service, in-channel response test resides as CATV ICLine:<nr1>.[1]
ICR	Returns the results of the last in-channel response measurement as CATV ICR:<nr2>, in dB.
LINe	Returns the selected power line frequency as CATV LINE:50 or CATV LINE:60.
LSTn	Returns the status of listen mode as CATV LSTN:ON or CATV LSTN:OFF.
MODe	Returns the status of CATV mode as CATV MODE:ON or CATV MODE:OFF.
NBW	Returns the noise bandwidth currently used for carrier-to-noise measurements as CATV NBW:<nr3>. The noise bandwidth is returned in Hz.
OPErator	Returns the current operator string as CATV OPERATOR:<string>.
SITe	Returns the current site string as CATV SITE:<string>.
SKIp	Returns the current skip status as CATV SKIP:ON or CATV SKIP:OFF.
SRVey	Returns the results of the last carrier survey measurement as CATV SRVey:<chan1>, <ampl1>, <freq1>, <ampl1>, <freq1>, <ampl1>, <freq1>. Each channel surveyed has three corresponding amplitude/frequency pairs. The format is NR1 for the channel, NR2 for amplitude (in the current units, dBmV) and NR3 for frequency (in Hz). See CLEvel
SURv	Returns the results of the last carrier survey measurement as CATV SURV:<chan1>, <ampl1>, <freq1>, <ampl1>, <freq1>, <ampl1>, <freq1>. Each channel surveyed has three corresponding amplitude/frequency pairs. The format is NR1 for the channel, NR2 for amplitude (in the current units, dBmV) and NR3 for frequency (in Hz). See FULLSURv.
VFIeld	Returns the status of the view modulation (field) mode as CATV VFIELD:ON or CATV VFIELD:OFF.
VLIne	Returns the status of the view modulation (line) mode as CATV VLINE:ON or CATV VLINE:OFF.
VPIc	Returns the status of the view picture mode as CATV VPIC:ON or CATV VPIC:OFF.

[1] **2715 only.**

Table 4–2: CATv? Query Arguments (Cont.)

Argument	Function
XMMode	Returns the selected cross modulation measurement mode as CATV XMMode:FDomain or CATV XMMode:TDomain (frequency or time domain).[1]
XMOd	Returns the results of the last cross-modulation measurement as `CATV XMOD:<nr2>`. Units are dBc. See `FULLXMOD`.

[1] **2715 only.**

CENsig

Arguments: None

This is a command with no argument that sets the center frequency to the frequency of the primary marker.

```
CENsig
```

CFSF <arg>

Arguments: `CENter`, `STArt`

This single-argument command designates the displayed frequency as the center or start frequency. The indicated frequency may be adjusted, depending on the current center frequency and frequency span, to ensure the resulting condition is permissible.

```
CFSF CENter

CFSF STArt
```

CFSF?

Arguments: None

This is a simple query whose response indicates whether the spectrum analyzer’s on-screen frequency is the center or start frequency.

```
CFSF?

    CFSF CENTER

    CFSF START
```

CLOck <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the date and time display on and off.

```
CLOck ON

CLOck OFF
```

CLOck? Arguments: None

This simple query returns the status of the date and time display.

```
CLOck?
    CLOCK ON
    CLOCK OFF
```

CLRKey Arguments: None

This is a command that clears the last key pressed so that only new key presses are reported by the `KEY?` query. After using the `CLRKey` command, `KEY?` returns `NULL` until a new key is pressed.

```
CLRKey
```

CLRMenu Arguments: None

This is a command that clears the menu defined with a `DEFMenu` command. This command clears the RAM space used by the User-Defined Menu, takes the spectrum analyzer out of menu mode, and clears the last key pressed as reported by the `KEY?` query. With this command, the spectrum analyzer always returns to the spectral display.

```
CLRMenu
```

CMEas Arguments: None

This is a command that causes the spectrum analyzer to perform a center measure. Use the `COUnt?` query to return the resulting counted value.

```
CMEas
```

CNBw <arg> Arguments: frequency in the range 1 Hz to 1.8 GHz

This single-argument command specifies the bandwidth used by the carrier-to-noise (C/N) feature to perform a C/N measurement. Hertz are used if no units are appended.

`CNBw 4.0 MHz` *(for example)*

CNBw?

Arguments: None

This simple query returns the noise bandwidth in hertz used by the carrier-to-noise (C/N) mode to perform a C/N measurement.

```
CNBw?
    CNBW 4.0E+6 (for example)
```

CNMode <arg>

Arguments: ON, OFF, IDLE

This single-argument command sets the carrier-to-noise (C/N) mode's on/off status. All marker modes are turned off when C/N mode is enabled.

```
CNMode ON
CNMode OFF
CNMode IDLE
```

The command `CNMode IDLE` has the same effect as `CNMode ON`.

CNMode?

Arguments: None

This simple query returns the status of the carrier-to-noise mode.

```
CNMode?
    CNMODE OFF
    CNMODE ON
    CNMODE IDLE
```

The response `CNMode IDLE` indicates that there is no signal, the AM detector is not selected, the noise is too close to the spectrum analyzer noise floor, or analog mode is selected.

CNResult?

Arguments: None

This simple query returns the result in decibels (dB) of the most recent carrier-to-noise (C/N) measurement performed by the spectrum analyzer's C/N feature. The measurement is updated at the end of the current sweep if C/N mode is enabled. `CNMode` must be set to `ON` to obtain valid results.

```
CNResult?
    CNRESULT –4.65E+1 (for example)
```

CNTtrak <arg>

Arguments: ON, OFF

This single-argument command enables and disables the frequency counter in signal track mode. The command sets counter resolution to 1 Hz when the counter is enabled.

```
CNTtrak ON

CNTtrak OFF
```

CNTtrak?

Arguments: None

This is a simple query whose response indicates whether the frequency counter is on or off in signal track mode.

```
CNTtrak?

    CNTTRAK OFF

    CNTTRAK ON
```

CONTinue

Arguments: None

This is a command that resumes an operation after the 2714 or 2715 pauses to display a prompt. Event code 403 (CATV PROMPT) is generated when the 2714 or 2715 is paused. This command clears the prompt in the same way that pressing the **[W]** key does.

```
CONTinue
```

See the CATv command.

COUnt?

Arguments: None

This simple query returns the result (in hertz) of the last frequency count performed as a result of the CMEas command.

```
COUnt?

    COUNT 55.250E+6
```

(for example)

CREs <arg>

Arguments: 1 Hz, 1 kHz

This single-argument command designates the frequency counter resolution.

```
CREs 1 Hz

CREs 1 kHz
```

CREs?

Arguments: None

This simple query returns the currently selected counter resolution. Values of 1 Hz and 1 kHz are possible.

```
CREs?

    CRES 1.E+0

    CRES 1.E+3
```

CSDefault

Arguments: None

This command sets the CATV CSO test frequencies to the default values.

CSFreq/CTFreq
<table index>, <arg>,
<frequency in Hz>

Arguments: REL, ABS, DEL

This command defines entries in the AUTO TEST FREQUENCIES table used by the automatic modes of the CTB (or CSO) test. `<table index>` is an integer value from 1 to 5. The `<arg>` parameter defines whether the frequency is absolute (ABS) or relative (REL) to the carrier, or whether the value should be deleted (DEL). The `<frequency in Hz>` parameter may define an offset value from the channel carrier, or a specific frequency. Offset values must be within the ±1800 MHz range. Frequencies must be within the 0 to 1800 MHz range.

The REL argument defines a test location relative to the carrier frequency of the current channel. For example, the following command defines (for table index 2) a test location that is 1 MHz above the channel carrier frequency:

`BTFreq 2, REL, +1 MHz` *(for example)*

The ABS argument defines an absolute test frequency. The following command defines a test frequency of 200 MHz for table index 3:

`BTFreq 3, ABS, 200 MHz` *(for example)*

The DEL argument deletes the table element defined by the `<table index>` argument. A frequency parameter must be present, but its value is ignored. This example shows how the value in table index 4 is deleted:

`BTFreq 4, DEL, 50 MHz` *(for example)*

CSFreq?/CTFreq?

Arguments: None

This simple query returns the complete AUTO TEST FREQUENCIES table in this format:

```
BTFreq 5,1 <type>,<freq>,2,<type>,<freq>,... 5,<type>,<freq>
```

The `<type>` field contains one of the three parameters (REL, ABS, or DEL).

The <freq> field is the frequency in the corresponding table location. If the table type is REL, <freq> is a value within the ±1800 MHz range. If the table type is ABS, <freq> is a value within the 0 to 1800 MHz range. If the table type is DEL, <freq> is 0.

CSInt

Arguments: Time in seconds from 1 to 21,600

This command sets the CATV continuous CSO test interval, for example:

```
CSInt 10;
```

CSInt?

Arguments: None

Returns the test interval for the CATV continuous CSO test, in the format:

```
CSINT 10;
```

CSNorm

Arguments: Time in hours from 1 to 24

This command sets the CATV continuous CSO normalization interval, for example:

```
CSNorm 1;
```

CSNorm?

Arguments: None

Returns the normalization interval for the CATV continuous CSO test, in the format:

```
CSNORM 1;
```

CSRFreq

Arguments: Carrier frequency for the CATV continuous CSO test. Format is:

```
<arg>,<freq in Hz>
```

where <arg> is REL, ABS, or DEL. See discussion CSFreq/CTFreq for further discussion of <arg>. DEL sets the carrier frequency to the default.

This command sets the carrier frequency for the CATV continuous CSO test. For example, the following command sets the carrier frequency to 200 MHz:

```
CSRFreq ABS, 200MHz;
```

CSRFreq?

Arguments: None

Returns the carrier frequency for the CATV continuous CSO test, in the format:

```
CSRFREQ ABS, 200E+6;
```

CTDefault

Arguments: None

This command sets the CATV CTB Auto test frequencies to default.

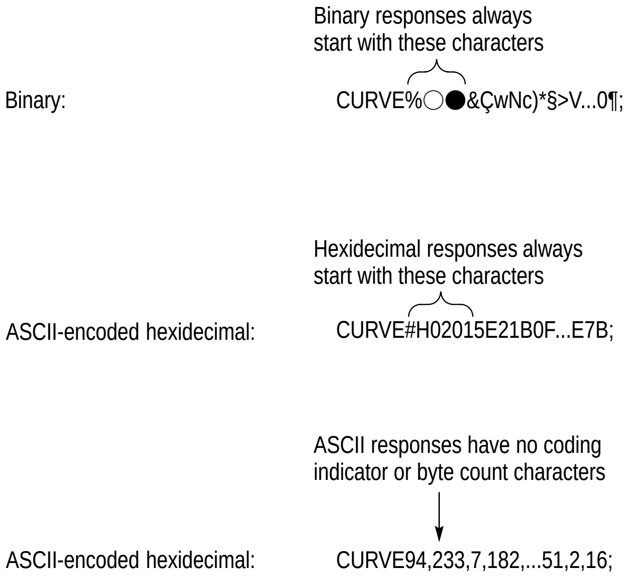
CURve <arg>

Arguments: Complex block of data described below.

This single-argument command enables a block of curve data to be sent to one of the spectrum analyzer's display registers. The data block represents the 512 horizontal points in a waveform. The encoding of the data points (ASCII-encoded decimal, ASCII-encoded hexadecimal, or binary) is determined by the current waveform preamble (see the `WFMpre` command). The register (A, B, C, or D) to which the data is sent is also determined by the preamble.

Data transferred to the spectrum analyzer can be previously returned waveforms or artificially generated curves. To ensure that the transferred data is not immediately overwritten, it should be transferred to a saved register. (Other methods are also possible, such as transferring to an active register in single sweep mode.)

If you were to display the message on your controller screen (for instance, by printing the response to the `CURve?` query with `HDR ON`), the response would resemble the following example:



Binary responses always start with these characters

Binary: CURVE%○●&ÇwNc)*§>V...0¶;

Hexidecimal responses always start with these characters

ASCII-encoded hexidecimal: CURVE#H02015E21B0F...E7B;

ASCII responses have no coding indicator or byte count characters

ASCII-encoded hexidecimal: CURVE94,233,7,182,...51,2,16;

CURve<space><ind><bc$_{hi}$><bc$_{lo}$><d$_{1}$><d$_{2}$>...<d$_{512}$><checksum>; where:

CURve	Command header
space	Header delimiter
ind	Absent when ASCII-encoded decimal is used; equals #H when ASCII-encoded hexadecimal is used; equals the % sign when binary is used
bc$_{hi}$	High order byte of the number of data points (always 512) plus one in the data block: absent in decimal, 00000010 in binary, 02 in hexadecimal
bc$_{lo}$	Low order byte of the number of data points (always 512) plus one in the data block: absent in decimal, 00000001 in binary, 01 in hexadecimal
d$_{1}$d$_{2}$...d$_{512}$	1st data point, 2nd data point,...512th data point of the curve; each data point may be represented by one to four bytes depending on encoding: ■ Binary: one byte ■ Hexadecimal: two bytes representing the hexadecimal numerals 0–F ■ Decimal: two to four bytes representing a comma delimiter plus one to three decimal numerals 0–9
checksum	Absent for ASCII-encoded decimal; otherwise: ■ 2’s complement of {[Sdi + bchi + bclo] MOD 256}

Figure 4–1: Format of Curve Data

Figure 4–1 shows the format of curve data. The checksum definition ensures that the sum (modulo 256) of the data points plus point count plus checksum equals zero.

Here are several more points worth noting about the various encodings:

- Binary encoding uses one byte per data point making it more compact and faster than the other techniques.
- Hexadecimal always uses two bytes per data point. Thus it requires no delimiter to separate the points.
- Decimal uses a variable number of bytes (1 to 3) to encode each data point, and therefore requires a data point delimiter (the comma).
- The use of delimiters in decimal encoding makes this form compatible with many spread sheets and word processors. This enables you to create custom waveforms for later transmission to the spectrum analyzer, or to edit waveforms previously returned from the spectrum analyzer.

The spectrum analyzer's graticule is represented by 500 intervals horizontally and 240 intervals vertically. The graticule corner coordinates are represented as shown in Figure 4–2.

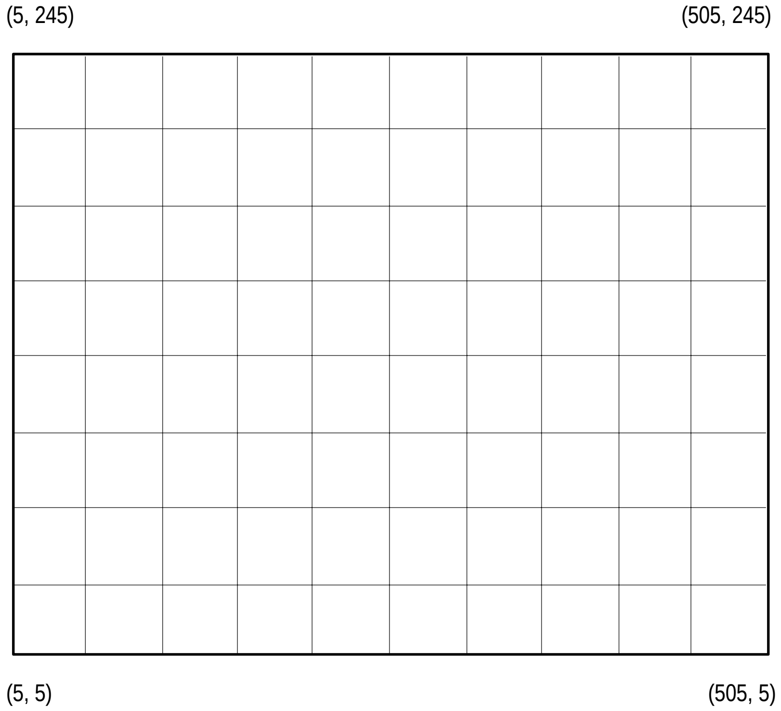


Figure 4–2: Spectrum Analyzer Graticule Coordinates

Points along the horizontal axis are numbered from 0 to 511. The resulting graticule is represented by points 5 to 505. Values outside this range extend beyond the graticule area. Therefore, the sixth data point (5) along the horizontal axis crosses the left graticule line; the 505th point crosses the right graticule line.

Vertically, the data are digitized into 256 values from 0 to 255. The resulting graticule is represented by points 5 to 245 (Figure 4–2). Values outside this range extend beyond the graticule area. Therefore, the sixth data point (5) along the vertical axis crosses the bottom graticule line. The 245th point crosses the top graticule line. See *Programming* for programming examples using `CURve` and `CURve?`.

CURve? or **CURve?** <arg>

Arguments: None, A, B, C, D

This is a query with either one or no argument that returns a complex response representing the contents of a spectrum analyzer waveform display register.

```
CURve?

CURve? A

CURve? B

CURve? C

CURve? D

    CURVE 94,233,7,...151,2,16; (for example)
```

The format of the response is determined by the previous `WFMpre` command. If no argument is specified, the source register of the curve data is also determined by the previous `WFMpre` command. See the `CURve` command discussion for data formats and other details.

The `CURve?` response with `HDR ON` will resemble one of the forms shown in the `CURve` command discussion.

Data may be returned from an active or inactive register whether or not it is saved (the register always contains data even if it is not displayed). Contrast this function with the `FILE` command which returns a stored curve file whether or not it is currently in a register (in binary only, and not in curve data format).

See *Programming* for GPIB programming examples using `CURve?`.

DATe <arg>

Arguments: Date in the form "DD–MON–YY"

This is a command with a string argument in the above format. DD is a two-digit day of the month, YY is the last two digits of the year, and MON represents the first three letters of the month. This command sets the real-time clock date. The DD and YY fields may be a single digit. If so, they are treated as if they had a leading digit of 0. The MON field may be any mixture of uppercase and lowercase letters. Note that these elements are separated by dashes (–), and the quotation marks (") must be present.

```
DATe "10–JAN–90"
```

DATe?

Arguments: None

This is a query that returns the current date in the format `DD-MON-YY`. `DD` is a two-digit day of the month, `YY` is the last two digits of the year, and `MON` represents the first three letters of the month.

```
DATe?
```

```
DATE "10-JAN-90"
```

DATIme?

Arguments: None

This is a query that returns the current date and current time in two comma-separated strings. The date is in the format `DD-MON-YY`. `DD` is a two-digit day of the month, `YY` is the last two digits of the year, and `MON` represents the first three letters of the month. The current time is in the format `HH:MM:SS` where `HH` is the hour, `MM` the minute, and `SS` the seconds.

```
DATIme?
```

```
DATIME "10-JAN-90","13:30:27"
```

DEFMenu <arg>

Arguments: `Ln: "string"`

This is a command with arguments separated by commas in the form above. The `Ln`: is called the link header, where `n` is a number between 1 and 16 that defines a display line number. The link argument is a user-defined string that appears on the specified display line. Only 32 characters of the string are used. Excess characters are discarded. If fewer than 32 characters are contained in the link argument, spaces are added to increase the string length to 32 characters.

This command replaces the spectrum analyzer display (spectral or menu) with a User-Defined Menu on the screen. If the instrument is already in a User-Defined Menu, only the lines referenced in this command are replaced. Use `CLRMenu` to clear a User-Defined Menu. The `DEFMenu` command also clears the last key press, but only if the User-Defined Menu space is clear.

See *Programming* for an example.

```
DEFMenu L0:"TEST MENU"
```

```
DEFMenu L0:"TEST MENU",L2:"TEST 1",L3:"TEST 2"
```

DETector <arg>

Arguments: AM, AMFm, FM, OFF

This single-argument command determines the type of signal detector used for the audio output. Depending on the argument, the AM, FM or both detector outputs are presented at the spectrum analyzer audio output. The audio output cannot be used in Video Monitor mode.

```
DETector AM

DETector AMFm

DETector FM

DETector OFF
```

DETector?

Arguments: None

This simple query returns the current status of the audio source. This may be the output of the AM detector, FM detector, neither, or both.

```
DETector?

    DETECTOR AM

    DETECTOR AMFM

    DETECTOR FM

    DETECTOR OFF
```

DIR? or **DIR?** <arg>

Arguments: None or <"string">

This query may be used with or without an argument.

When used without an argument, DIR? returns a formatted system file directory listing that is similar to the one displayed by the key sequence **[UTIL] [4] [6]**.

Each line in the listing (except the first and last) is formatted as in this example:

filename, read/write enabled (R or W), size in bytes

```
DIR?

    DIR " 12.88,    ,      0

    TMPDBG, RW, 16380

        12.88, ,      0

          :

    DSET00, RW, 386
```

```
      SETOBU, RW, 386
           :
      DSET38, RW , 386
         12.88, ,     0
   ";  (for example)
```

Each line is separated by a line feed.

An argument specifying a file `<string>` may follow the `DIR?` query to return the directory entry for that file. The referenced file becomes the default file name for the `FILE?` query.

```
DIR? "DSET00";
   DIR "DSET00, RW, 386";
```

Event 844 (FILE NOT FOUND) is returned if the file cannot be found. Event 103 (COMMAND ARGUMENT ERROR) is returned if the filename is not formatted properly. Both conditions return the default response `" ,, 0"`.

DIScor <arg>

Arguments: `ON`, `OFF`

This single-argument command enables and disables the spectrum analyzer’s frequency corrections. When `DIScor` is `ON` (frequency corrections are disabled), the message “FREQ COR OFF” appears on the spectrum analyzer screen.

```
DIScor ON
DIScor OFF
```

DIScor?

Arguments: None

This simple query returns the current on/off status of the spectrum analyzer’s frequency corrections. Note that `DISCOR ON` means the frequency corrections are off.

```
DIScor?
   DISCOR ON
   DISCOR OFF
```

DLIne <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the display line feature on and off. `DLINE` cannot be turned on if the A-register is being used (waterfall mode, min hold, ensemble averaging, etc.). Attempting to do so generates an event 787 (DESTINATION WAVEFORM CONFLICT).

```
DLIne ON
```

```
DLIne OFF
```

DLIne?

Arguments: None

This simple query returns the current on/off status of the display line feature.

```
DLIne?
```

```
DLINE ON
```

```
DLINE OFF
```

DLLimit <arg>

Arguments: `OFF`, `OVEr`, `OVUNder`, `UNDer`

This single-argument command controls the status of the display line limit detector. When the limit detector is on, an SRQ and an event 895 are generated when the limit condition is exceeded. The following table shows the four arguments and their resulting condition.

Argument	Condition
`OVEr`	Alarm when signal > display line
`UNDer`	Alarm when signal < display line
`OVUNder`	Alarm when signal > display line or when signal < threshold
`OFF`	No alarm

```
DLLimit OFF
```

```
DLLimit OVEr
```

```
DLLimit OVUNder
```

```
DLLimit UNDer
```

DLLimit?

Arguments: None

This simple query returns the current status of the display line limit detector.

```
DLLimit?

    DLLIMIT OFF

    DLLIMIT OVER

    DLLIMIT OVUNDER

    DLLIMIT UNDER
```

DLValue <arg>

Arguments: `MARker`, amplitude in the range –150 to +100 dBm

This single-argument command turns on the display line and sets its amplitude. A numeric argument sets the amplitude value. The units are the currently selected reference level units. However, the argument must be within a range of –150 dBm to +100 dBm, or an equivalent in alternate units.

The `MARKer` argument sets the display line to the amplitude of the primary marker.

```
DLValue MARker

DLValue –30
```
(for example)

DLValue?

Arguments: None

This simple query returns the display line value. Units are the currently selected ref level units.

```
DLValue?
```

DSRc <arg>

Arguments: `AM`, `EXTernal`, `FM`

This single-argument command designates the source of the signal displayed by the spectrum analyzer as the internal `AM` detector (normal display), internal `FM` detector (useful for FM deviation checks), or an `EXTernal` input.

If `FM` or `EXTernal` is selected, the spectrum analyzer is placed in zero span mode and max/min signal acquisition is selected. `FM` and `EXTernal` are not allowed in DBUVM mode.

```
DSRc AM

DSRc EXTernal

DSRc FM
```

DSRc? Arguments: None

This simple query returns the currently selected source of the signal displayed by the spectrum analyzer.

```
DSRc?

    DSRC AM

    DSRC FM

    DSRC EXTERNAL
```

EOS <arg> Arguments: ON, OFF

This single-argument command enables and disables the end-of-sweep SRQ. When EOS is ON, an end-of-sweep SRQ is normally generated at the end of each spectrum analyzer sweep. However, intermediate end-of-sweep SRQs are suppressed in the case of normalization, User-Defined Programs, plots, signal searches, and ensemble averages until the process is complete. A single SRQ is then issued indicating end-of-process. For instance, if a 10-sweep ensemble average is compiled, an SRQ will not occur following each sweep. Rather, a single SRQ accompanied by event 882 (ENSEMBLE AVERAGE) COMPLETE, occurs after the averaging process is finished.

```
EOS ON

EOS OFF
```

EOS? Arguments: None

This simple query returns the status of the end-of-sweep generator.

```
EOS?

    EOS ON

    EOS OFF
```

ERAse <arg> Arguments: Numerals from 2 to 39 except 9, 19 and 29

This single-argument command specifies a stored settings register to be erased. Registers 2 to 39 may be specified, except for registers 9, 19, and 29, which are invalid register numbers. If the settings are protected (locally or because PROTSET is ON), only the waveforms associated with the indicated settings are erased and an SRQ and event 839 are generated. If register 9, 19, or 29 is specified, an SRQ and Event 701 are generated.

`ERAse 2`

`ERAse 24` *(for example)*

ERR?

Arguments: None

This simple query returns an integer event code. `ERR?` is equivalent to `EVEnt?` and is preserved in the spectrum analyzer for consistency with the command sets of other instruments.

```
ERR?
```

```
    ERR 878
```
(for example)

EVEnt?

Arguments: None

This simple query returns an integer event code. If `RQS` is `ON`, a serial poll must be performed following an SRQ and prior to sending `EVEnt?` to obtain the correct event code. If `RQS` is `OFF`, `EVEnt?` may be sent without a serial poll.

```
EVEnt?
```

```
    EVENT 878
```
(for example)

FCMode

Arguments: ON or OFF

ON enables FCOR? to return flatness correction values. OFF causes FCOR? to always return 0.0. Defaults to OFF on power up. Since the marker amplitude includes the flatness correction, FCOR? results should not be added to marker readings. This command defaults to OFF so that existing software which does add FCOR? results to marker amplitudes will continue to obtain correct amplitude results. If the Flatness value is needed for some other purpose, FCMode ON will enable the FCOR? query.

FCMode?

This query return the status (ON or OFF) of FCMode.

FILE <arg>

Arguments: `"<filename>"`, `<data block>`

This is a complex single- or multiple-argument command that transfers a previously stored spectrum analyzer file from the controller to the spectrum analyzer. When used with only the `<filename>` argument, the command establishes the name of the file to be transferred to the controller by the next `FILE?` query. The `"<filename>"` must be surrounded by quotation marks (”). Tables 4–3 and 4–4 list the various user-alterable file types and allowable filenames.

The `FILE` command and `FILE?` query are intended to transfer files, through the mechanism of up-loading and subsequent downloading, between different spectrum analyzers. For instance, you might develop a User Definable Program (UDP) in one 2714 or 2715, transfer it to the controller using the `FILE?` query, and subsequently download it to a number of other spectrum analyzers using the

FILE command. Be aware, however, that if files are transferred between spectrum analyzers with different installed options, uncertain results may occur.

The FILE command/query can also be used to back up important settings files, or the reference normalizations, in preparation for changing the NVRAM battery.

When files are originally returned from the spectrum analyzer with the FILE? query, HDR is usually set ON. This causes the ASCII strings FILE and <filename> to precede the actual data. They are stored as the first bytes of the disk file. It is then unnecessary to transmit the FILE header or the <filename> when restoring the file to the spectrum analyzer.

Read the disk file into a string variable called FILEDAT$ as an example. The string variable will be of exactly the form needed to send the file to the spectrum analyzer as in this example:

```
FILE "<filename>",<data block>
```

Simply transmit FILEDAT$ to the spectrum analyzer.

Table 4–3: File Types

File Type	Description
Settings	Each file saves the control settings for a particular register (A, B, C, D) in a numbered location (00–39, excluding 09, 19, & 29). The numbers correspond to the stored settings listed under **[UTIL] [1]**.
Curves	Each file saves the curve data from a particular register (A, B, C) in a numbered location (00–39, excluding 09, 19, & 29). The numbers correspond to the curves saved with the settings listed under **[UTIL] [1]**. D-register curves are never saved.
User-Defined Programs	Each file saves a keystroke command sequence representing a user-defined program in a numbered location (00 to 08); numbers correspond to the programs stored under **[USER DEF]**.
Antenna Tables	Each file saves an antenna table representing antenna data for a particular antenna in a numbered location (01 to 05); numbers correspond to the tables stored under **[INPUT] [3] [9]**.
Normalization	Normalization files save data generated by normalizing the spectrum analyzer, including reference normalizations.
Channel Table	Each file saves data pertaining to a particular channel in numbered locations. Numbers correspond to tables accessed under **[CATV APPL] [8] [1]**.

Table 4–4: Valid File Names

Curves	Settings	User Definable Programs	Antenna Tables	Normalizations
nWFM00	nSET00	UDP00	ACF1	NORM
:	:	:	:	
nWFM09	nSET09	UDP08	ACF5	
n=A,B,C	n=A,B,C,D			

Several other files of little use to the average user may be present, but should not be altered. These are listed in Table 4–5.

Table 4–5: Miscellaneous Files

Name	Description
12.88	Version
REF0N	CATV frequency response reference
SEARCH	Signal search configuration
SETUP	Instrument configuration
S_CENT	Centronics configuration
S_GPIB	GPIB configuration
S_PLOT	Plotter configuration
S_RTC	Real-time clock configuration
S_CATV	CATV mode configuration

Other files of a temporary nature may be created by the spectrum analyzer for internal purposes. These files should not be altered.

If HDR was OFF when the file was returned, "FILE" must precede the disk file. In this case, transmit the following message:

```
FILE FILEDAT$
```

See *Programming* for programming examples.

FILE? or FILE? <arg>

Arguments: None, "<filename>"

This is a simple query that returns a file stored in the spectrum analyzer to the controller. When used without an argument, FILE? returns the file specified by the previous FILE command. When used with a <filename> argument, FILE? returns the named file. The filename, when specified, must be in quotes ("). All filenames must match (including case) one of those listed in Table 4–4.

The filenames in the spectrum analyzer are established by its firmware. A directory of currently created files can be viewed by pressing the key sequence **[UTIL] [4] [6]** or **[UTIL] [5] [4] [1] [0]**. Files are created within the spectrum analyzer's memory only as required. That is, a BSET03 settings file is only present when B-register settings have been stored previously in the third storage location.

DSET00 and SET0BU are two special files created automatically by the spectrum analyzer. They contain the D-register settings used when the spectrum analyzer was last turned off. They are listed as LAST POWER-DOWN, which is accessed by pressing the key sequence **[UTIL] [1] [0]**. SET0BU is a backup in case DSET00 becomes corrupted at the next power-up.

The FILE? query enables you to store a spectrum analyzer file on disk for later restoration to the same or another spectrum analyzer. The file is in binary format, and the first bytes of the response are the ASCII character codes for <filename>. If HDR is ON before issuing the FILE? message, the query and response have this format:

```
FILE? "<filename>"

    FILE "<filename>",<data block>
```

The response is in exactly the format needed to send a file to the spectrum analyzer. The general approach to file transfer is to read the response (including header and filename) into a string variable and write the variable to a disk file. The presence of FILE "<filename>" within the disk file makes it possible to restore the file to the spectrum analyzer without explicitly sending the FILE command or specifying the spectrum analyzer file name.

Follow this sequence to store a spectrum analyzer file:

1. Send HDR ON to the 2714 or 2715.
2. Send FILE? "<filename>" query to 2714 or 2715.
3. Read response into string variable FILEDAT$.
4. Write FILEDAT$ to disk file MYPROG.

Use this sequence to restore the file:

1. Read file MYPROG to string variable FILEDAT$.

2. Send FILEDAT$ to the 2714 or 2715.

Be aware that files can theoretically occupy up to 64 Kbytes. Binary blocks are limited to 64 Kbytes because of the 16-bit byte count.

See *Programming* for programming examples.

FINe <arg>

Arguments: `ON`, `OFF`

This single-argument command selects 1 dB reference level steps when set to `ON` and 10 dB steps when set to `OFF`.

```
FINe ON

FINe OFF
```

FINe?

Arguments: None

This simple query returns the current on/off status of the reference level steps. `ON` equals 1 dB/step and `OFF` equals 10 dB/step.

```
FINe?

    FINE ON
    FINE OFF
```

FOFfset <arg>

Arguments: Frequency in the range –1000 GHz to +1000 GHz

This single-argument command turns on frequency offset mode (see `FOMode`) and sets the spectrum analyzer frequency offset value. A value of 0 turns off frequency offset mode. Frequency units may be appended; otherwise hertz are assumed.

When enabled, the value of the frequency offset affects the display and any subsequent `FREq`, `MFReq`, `STStop`, `SSBegin` and `SSEnd` commands.

```
FOFfset 5.15 GHz (for example)
```

FOFfset?

Arguments: None

This simple query returns the value of the frequency offset in hertz. Zero is returned if the frequency offset is disabled.

```
FOFfset?

    FOFFSET 0
    FOFFSET 5.15E+9  (for example)
```

FOMode <arg>

Arguments: ON, OFF

This single-argument command turns frequency offset on or off. When frequency offset is enabled, the last offset frequency value is used (see FOFfset). When enabled, the value of the frequency offset affects the display and any subsequent FREq, MFReq, STStop, SSBegin, and SSEnd commands.

```
FOMode ON
```

```
FOMode OFF
```

CATV mode is turned off when FOMode is turned on. In addition, CATV mode is not available when FOMode is on.

FOMode?

Arguments: None

This simple query returns the current on/off status of the frequency offset mode.

```
FOMode?

    FOMODE OFF

    FOMODE ON
```

FREq <arg>

Arguments: frequency in the range –10 MHz to 1.8 GHz

This single-argument command sets the center or start frequency to the indicated value. Frequency units may be appended, otherwise hertz are assumed. Interpretation as center or start frequency depends upon the setting of CFSF. If start frequency is selected, the span is checked and the start frequency may be adjusted to ensure the center frequency is never more than 1.8 GHz. The argument is offset by the FOFfset command if FOMode is enabled.

FREq 193.25 MHz *(for example)*

FREq?

Arguments: None

This simple query returns the currently selected center or start frequency.

FREq?

    FREQ 193.25E+6 *(for example)*

GRAt <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the graticule illumination on and off.

```
GRAt OFF

GRAt ON
```

GRAt?

Arguments: None

This simple query returns the current on/off status of the graticule illumination.

```
GRAt?

    GRAT OFF

    GRAT ON
```

GTL

Arguments: None

This is a command that removes the spectrum analyzer from the remote state and returns it to local mode. It is intended as the RS-232 equivalent of the GPIB universal GTL command (see *Appendix B: GPIB System Concepts*).

```
GTL
```

HDR <arg>

Arguments: `ON`, `OFF`

This single-argument command turns the header on and off in a query response. When `HDR` is `on`, a command header describing the nature of the response precedes the response. This also makes the response an executable command.

```
HDR OFF

HDR ON
```

The following table lists several queries and their potential responses with `HDR ON` and `HDR OFF`. In the case of most linked numerical arguments (not those resulting from `VRTdsp?`, `WAVfrm?`, and `WFMpre?` queries), the link is turned off along with the command header. The link remains for linked character arguments (see `MFREq?` and `VIEw?` in the table) when `HDR` is `OFF`.

HDR ON	HDR OFF

`FREq?`
`FREQ 193.25E+6`
	`FREq?`
	`193.25E+6`

`GRAt?`
`GRAT ON`
	`GRAt?`
	`ON`

`MFREq? DELta`
`MFREQ DELTA:4.5E+6`
	`MFREq? DELta`
	`4.5E+6`

`VIEw? A`
`VIEW A:ON`
	`VIEw? A`
	`A:ON`

HDR?

Arguments: None

This simple query returns the current on/off status of the response header.

```
HDR?

    HDR OFF

    HDR ON
```

HELp?

Arguments: None

This simple query returns a list of all instrument-specific commands, including commands for all options whether present or not.

```
HELp?

    HELP ACQMODE,AQP,...WFMPRE,ZEROSP
```

HRAmpl

Arguments: None

This is a command with no argument that moves the primary marker from its current position to the peak of the next higher on-screen signal. If the marker is not enabled, the command enables a marker. If signal track is enabled, `HRAmpl` turns signal track off, enables the marker, and assigns the knob function to marker control. If there is no higher peak and `SGErr` is on, an SRQ and event 896 are generated.

```
HRAmpl
```

ICDefault

Arguments: None

This is a command with no argument that recalls the default values for the AUTO TEST LOCATIONS table used by the In-Channel Response test. The default values, listed in order within the table, are –0.5 MHz, +0.5 MHz, +1.0 MHz, +2.0 MHz, +3.0 MHz, and +3.58 MHz.

```
ICDefault
```

ICFreq <table index>, <arg>, <frequency>

Arguments: `REL`, `DEL`

This command defines entries in the AUTO TEST LOCATIONS table used by the automatic mode of the In-Channel Response test. `<table index>` is an integer value from 1 to 6 that defines the table location. The `<arg>` parameter defines whether the frequency is relative (`REL`) to the carrier, or whether the value defined by the `<table index>` parameter should be deleted (`DEL`). The `<frequency>` parameter is the frequency in Hz. It may be a positive or negative value within the range of 0 to 1800 MHz.

The `REL` argument defines a test location relative to the carrier frequency of the current channel. For example, the following command defines (for table index 2) a test location that is 1 MHz above the channel carrier frequency:

`ICFreq 2, REL, +1 MHz` *(for example)*

The `DEL` argument deletes the table element defined by the <table index> argument. A frequency parameter must be present, but its value is ignored. This example shows how the value in table index 4 is deleted:

`ICFreq 4, DEL, +2.5 MHz` *(for example)*

ICFreq?

Arguments: None

This simple query returns the complete AUTO TEST LOCATIONS table in this format:

```
ICFREQ 6,1 <type>,<freq>,2,<type>,<freq>,... 6,<type>,<freq>
```

The `<type>` field is either `REL` (relative) or `DEL` (delete).

The `<freq>` field is the frequency in the corresponding table location. If the table type is `REL`, `<freq>` is a value within the ±1800 MHz range. If the table type is `DEL`, `<freq>` is 0.

ID? Arguments: None

This query returns the instrument identification, firmware version, and installed options.

```
ID?

    ID TEK/2714 or 2715,V81.1,"VERSION 02.28.92

    FIRMWARE","300HZ,1,10,100KHZ,1MHZ

    RBW FLTR","GPIB","NVM 12.88","OPT NVM 12.88" (for example)
```

The items in quotes (”) indicate the firmware version and options installed in the instrument. These may vary depending on how your instrument is equipped.

INIT Arguments: None

This is a command that places the spectrum analyzer in its power-up configuration. Factory default power-up settings are used unless user-defined power-up settings have been implemented.

```
INIT
```

KEY <arg> Arguments: Various mnemonics as listed in Table 4–6.

This single-argument command simulates pressing a key on the spectrum analyzer front panel. The general form of the command resembles this example:

```
KEY <arg>
```

where <arg> is the mnemonic for the key press to be simulated. All permissible mnemonics are listed in Table 4–6. For instance, you send the following message to simulate pressing the **[INPUT Menu]** key:

```
KEY INPutmenu (for example)
```

To turn on the calibration signal using the KEY command, you can send this message:

```
KEY INPutmenu;KEY M9 (for example)
```

The KEY command is not as efficient as using a dedicated instrument-specific command, such as CALSig ON, to achieve the same result. It requires more time and memory space. Nevertheless, the command does provide an alternative if you have difficulty implementing a dedicated command. In fact, you can perform all remote programming using only the KEY command. However, because of its increased memory requirement and decreased speed, use of the KEY command is discouraged as a general purpose programming technique. Note that there are no KEY commands for the **[PLOT]** and **[POWER]** keys.

Table 4–6: Arguments of the KEY Command

Mnemonic	Key
`APplmenu`	Application Menu
`A`	Display A
`B`	Display B
`BS`	Backspace key
`C`	Display C
`D`	Display D
`CTRMeas`	Center Measure
`DEMMenu`	Demodulator/Generator Menu
`DIspmenu`	Display Menu
`FIne`	Fine Ref Lvl steps
`FREQAsgn`	Assign Keypad to Frequency, or to Channel in CATV Mode
`FREQDown`	Frequency Down
`FREQUp`	Frequency Up
`INPutmenu`	Input menu
`KNOBRight`	Turn knob right 1 step
`KNOBLeft`	Turn knob left 1 step
`M0`	0 key
`M1`	1 key
`M2`	2 key
`M3`	3 key
`M4`	4 key
`M5`	5 key
`M6`	6 key
`M7`	7 key
`M8`	8 key
`M9`	9 key
`MAXHold`	Max Hold
`MAXSpan`	Max Span
`MKREnab`	Marker On/Off/Delta
`MKRLeft`	Marker Left
`MKRMenu`	Marker/Frequency Menu
`MKRPeak`	Marker Peak
`MKRRight`	Marker Right

Table 4–6: Arguments of the KEY Command (Cont.)

Mnemonic	Key
PERIOD	Period
RDOut	Readout
REFAsgn	Assign keypad to Ref lvl
REFDown	Ref Lvl Down
REFUp	Ref Lvl Up
RESAuto	Auto ResBW
RESDown	ResBW Down
RESUp	ResBW Up
SAve	Save
SGLswp	Single Sweep
SPANAsgn	Assign keypad to Span
SPANDown	Span Down
SPANUp	Span Up
SWPAuto	Sweep Auto
SWPDown	Sweep Time Down
SWPMenu	Swp/Trig Menu
SWPUp	Sweep Time Up
TERMW	First Terminator
TERMX	Third Terminator
TERMY	Second Terminator
TERMZ	Fourth Terminator
USErdef	User Def Menu
UTilmenu	Utility Menu
VIDflt	Video Filter
VRTLIn	Lin Mode
VRTLOg	Log Mode
Zerospan	Zero Span

KEY? Arguments: None

This simple query returns the identity of the last key pressed that has not been reported by a previous KEY? query. The name of the key uses the same syntax as the KEY command. If the key name returned is NULL, then no keys have been pressed since the last CLRMenu command or KEY? query has been executed. This query also clears the last key press, ensuring that the same key press is not reported more than one time.

```
KEY?
```

```
    KEY M1
```
(for example)

```
    KEY NULL
```
(for example)

***NOTE**. NULL is returned if the **[PLOT]** key is the last key pressed.*

LRAmpl Arguments: None

This is a command with no argument that moves the primary marker from its current position to the peak of the next lower on-screen signal. If the marker is not enabled, the command turns on a marker. If signal track is enabled, LRAmpl turns it off, enables the marker, and assigns the knob function to marker control. If there is no lower peak and SGErr is on, an SRQ and an event 896 are generated.

```
LRAmpl
```

MAMpl? <arg> Arguments: None, PRImary, SECond, DELta

This is a simple query with one or no arguments. It returns a linked response indicating the amplitude of the primary (<arg> = none or PRImary) or secondary (<arg> = SECond) marker, or their amplitude difference (<arg> = DELta). The applicable units are those currently selected for the reference level unit.

```
MAMpl?
```

```
    MAMPL PRIMARY:6.8
```
(for example)

```
MAMpl? SECond
```

```
    MAMPL SECOND:2.4
```
(for example)

```
MAMpl? DELta
```

```
    MAMPL DELTA:4.4
```
(for example)

MARker <arg>

Arguments: `ON`, `OFF`, `SINgle`, `DELta`

This single-argument command turns markers on and off. Turning on a marker places the knob function in marker control and disables signal track mode, bandwidth measurement mode, noise measurement mode, and C/N measurement mode. The markers cannot be turned on in analog display or Video Monitor modes, or in waterfall mode unless the D register is enabled.

`MARker ON`	*(Turns on primary marker)*
`MARker SINgle`	*(Turns on primary marker)*
`MARker DELta`	*(Turns on both markers)*
`MARker OFF`	*(Turn off all markers)*

MARker?

Arguments: None

This simple query returns the on/off status of the markers.

```
MARker?

    MARKER SINGLE

    MARKER DELTA

    MARKER OFF
```

MEMory?

Arguments: None

This query returns two integer numbers separated by a comma. The first number represents the total amount of free NVRAM. The second number represents the largest contiguous block of free NVRAM. The values depend on the options installed and the number of waveforms, settings, programs, and other data stored in the spectrum analyzer's memory. Values are always multiples of 16.

```
MEMory?

    MEMORY 16464,3296
```
(for example)

MEXchg

Arguments: None

This is a command that requires no argument. It interchanges the primary (stationary) and secondary (moveable) markers. If delta marker mode is not active, the command generates an SRQ and event code 825.

```
MEXchg
```

MFReq <arg>

Arguments: Number in the range –10 MHz to 1.8 GHz

This single argument command sets the frequency of the primary marker. Hertz are assumed unless units are appended. The overall range is –10 MHz to 1.8 GHz, but the value specified must be within the current spectrum analyzer on-screen frequency span or an SRQ and event code are generated. This command is not valid in zero span mode.

`MFReq 193.25 MHz` *(for example)*

MFReq? <arg>

Arguments: None, `PRImary`, `SECond`, `DELta`

This is a simple query with one or no arguments. It returns a linked response indicating the frequency of the primary (`<arg> = none` or `PRImary`) or secondary (`<arg> = SECond`) marker, or their frequency difference (`<arg> = DELta`). The units are hertz.

```
MFReq?

    MFREQ PRIMARY:193.25E+6
```
(for example)

```
MFReq? SECond

    MFREQ SECOND:197.75E+6
```
(for example)

```
MFReq? DELta

    MFREQ DELTA:4.5E+6
```
(for example)

MHDest <arg>

Arguments: `A`, `B`, or `C`

This single-argument command selects the MIN Hold destination waveform.

```
MHDest A

MHDest B

MHDest C
```

MHDest?

Arguments: None

This simple query returns the MIN Hold destination waveform.

```
MHDest?

    MHDEST A

    MHDEST B

    MHDEST C
```

MKTime <arg>

Arguments: Number in the range 0 to 20

This single-argument command sets the time of the primary marker. The command is valid only in zero span mode. Seconds are assumed unless units are appended. The specified value must be within the current spectrum analyzer on-screen time span or an SRQ and event code are generated.

`MKTime 204 Msec` *(for example)*

MKTime? <arg>

Arguments: None, `PRImary`, `SECond`, `DELta`

This is a simple query with one or no arguments. It returns a linked response indicating the time of the primary (`<arg> = none` or `PRImary`) or secondary (`<arg> = SECond`) marker, or their time difference (`<arg> = DELta`). The units are seconds.

`MKTime?`

`MKTIME PRIMARY:4.67E-4` *(for example)*

`MKTime? SECond`

`MKTIME SECOND:8.98E-4` *(for example)*

`MKTime? DELta`

`MKTIME DELTA:4.31E-4` *(for example)*

MLFtnxt

Arguments: None

This is a command that requires no argument. It moves the primary marker from its current position to the next signal peak to the left. If signal track is enabled, `MLFtnxt` turns signal track mode off, enables the primary marker, and assigns the knob function to marker control. If `SGErr` is `ON` and a peak does not exist, an SRQ and event code are generated.

```
MLFtnxt
```

MMAx

Arguments: None

This is a command that requires no argument. It moves the primary marker from its current position to the highest on-screen signal peak. If signal track is enabled, `MMAx` turns signal track mode off, enables the primary marker, and assigns the knob function to marker control. If `SGErr` is `ON` and a higher peak does not exist, an SRQ and event code are generated.

```
MMAx
```

MNHld <arg>

Arguments: OFF, ON

This single-argument command turns the minimum hold feature on and off. This command is not allowed under the following conditions:

- Analog mode is being used
- Waterfall mode is enabled
- The destination register is A and display line is on
- dBμV/m is enabled and destination register is the same as for MIN Hold
- Ensemble averaging is enabled

```
MNHld ON

MNHld OFF
```

MNHld?

Arguments: None

This simple query returns the on/off status of the minimum hold feature.

```
MNHld?

    MNHLD ON

    MNHLD OFF
```

MPOs? <arg>

Arguments: None, PRImary, SECond, DELta

This is a query with one or no argument. It returns a linked integer response indicating the horizontal position of the primary (<arg> = none or PRImary) or secondary (<arg> = SECond) marker, or their horizontal difference (<arg> = DELta). See the CURve command for an explanation of screen coordinates.

```
MPOs?

    MPOS PRIMARY:356  (for example)

MPOs? SECond

    MPOS SECOND:233  (for example)

MPOs? DELta

    MPOS DELTA:123  (for example)
```

MRGTnxt

Arguments: None

This is a command that requires no argument. It moves the primary marker from its current position to the next signal peak to the right. If signal track is enabled, `MRGTnxt` turns signal track mode off, enables the primary marker, and assigns the knob function to marker control. If `SGErr` is on and a peak does not exist, an SRQ and event code are generated.

```
MRGTnxt
```

MSGdlm <arg>

Arguments: `Lf` (line feed), `Semicolon`

This single-argument command selects a line feed character or semicolon as a message delimiter.

```
MSGdlm Lf
```

```
MSGdlm Semicolon
```

MSGdlm?

Arguments: None

This is a simple query whose response indicates the currently selected message delimiter.

```
MSGdlm?
```

```
    MSGDLM LF
```

```
    MSGDLM SEMICOLON
```

MSTep

Arguments: None

This is a command that requires no argument. It is equivalent to turning the frequency/markers knob one click in the counterclockwise direction. The spectrum analyzer’s response depends upon the currently selected knob function.

```
MSTep
```

MTUNE <arg>

Arguments: Value in the range –1.8 GHz to +1.8 GHz

This single argument command changes the frequency of the primary marker by the indicated amount. Negative values indicate a decrease in frequency. Although the range is ±1.8 GHz, the value specified must position the new marker frequency within the spectrum analyzer’s on-screen frequency span or an SRQ and event code are generated.

`MTUNE 546 kHz` *(for example)*

MVPos? <arg>

Arguments: None, `PRImary`, `SECond`, `DELta`

This is a query with one or no argument. It returns a linked integer response indicating the vertical position of the primary (`<arg> = none` or `PRImary`) or secondary (`<arg> = SECond`) marker, or their vertical difference (`<arg> = DELta`). The `0`-point is at the bottom of the screen. See the `CURve` command for a more complete explanation of screen coordinates.

```
MVPos?

    MVPOS PRIMARY:356  (for example)

MVPos? SECond

    MVPOS SECOND:233  (for example)

MVPos? DELta

    MVPOS DELTA:123  (for example)
```

MXHld

Arguments: `OFF`, `ON`

This single-argument command turns the maximum hold feature on or off.

```
MXHld ON

MXHld OFF
```

MXHld?

Arguments: None

This simple query returns the current on/off status of the maximum hold feature.

```
MXHld?

    MXHLD ON

    MXHLD OFF
```

MXRlvl <arg>

Arguments: `NOMinal`, number in range –50 to –20

This single-argument command sets the required level of signal amplitude at the input of the spectrum analyzer’s first mixer to produce full-screen (top graticule line) deflection in 2 dB steps. Odd values are rounded. Units are not allowed; the number is interpreted as dBm. `NOMinal` selects the factory default value of –30 dBm.

```
MXRlvl NOMinal

MXRlvl –24  (for example)
```

MXRlvl? Arguments: None

This simple query returns the signal amplitude required at the input of the spectrum analyzer's first mixer to deflect the display to the top graticule line.

```
MXRlvl?
```

```
    MXRLVL –30
```
(for example)

MXSpn <arg> Arguments: `OFF`, `ON`

This single-argument command turns the maximum span mode on or off. The spectrum analyzer returns to the previously selected span/division when MAX Span is turned off.

```
MXSpn ON

MXSpn OFF
```

MXSpn? Arguments: None

This simple query returns the current on/off status of the maximum span feature.

```
MXSpn?

    MXSPN ON

    MXSPN OFF
```

NNBw <arg> Arguments: Number in the range 1 Hz to 1.8 GHz

This single-argument command sets the noise bandwidth for normalized noise mode measurements. Units may be appended; otherwise hertz are assumed.

```
NNBw 4 MHz
```
(for example)

NNBw? Arguments: None

This simple query returns the noise bandwidth in hertz to be used for normalized noise mode measurements.

```
NNBw?

    NNBW 4.0E+6
```
(for example)

NNMode <arg>

Arguments: `OFF`, `ON`, `IDLE`

This single-argument command turns the normalized noise measurement mode on or off. The command also sets `TMODE` to `MARKER`, `ACQMODE` to `MAXMIN`, and `SGTRAK` to `OFF`. The command cannot be used in analog mode, video monitor mode, waterfall mode when the D-register is off, or in linear display mode. Normalized noise mode measurements cannot be made on a saved waveform. The `NNMode IDLE` command is equivalent to `NNMode ON`. The `IDLE` response indicates that the mode is enabled but no signal is present to measure.

```
NNMode ON

NNMode OFF

NNMode IDLE
```

NNMode?

Arguments: None

This simple query returns the current status of the normalized noise measurement mode. `IDLE` indicates when the noise is too close to the spectrum analyzer noise floor, the AM detector is not enabled, MAX Span is active, the waveform is saved, or the spectrum analyzer is in analog display mode.

```
NNMode?

    NNMODE ON

    NNMODE OFF

    NNMODE IDLE
```

NNResult?

Arguments: None

This simple query returns the result of the most recent normalized noise measurement. The result is updated at the end of each sweep when the normalized noise measurement mode is enabled. The units are those selected as reference level units.

```
NNResult?

    NNRESULT −93.5
```
(for example)

NORM

Arguments: `ALL`, `AMPlitude`, `FREquency`

This single-argument command instructs the spectrum analyzer to carry out the indicated normalizations.

`NORM ALL`	*(all normalization except reference)*
`NORM AMPlitude`	*(amplitude normalizations)*
`NORM FREquency`	*(frequency normalizations)*

NORM? Arguments: None

This simple query returns a listing of the current normalization parameters. The following list shows the format of the response. Actual values for each category will vary.

```
NORM?

   NORM "TEK 2714 or 2715
   CURRENT NORMALIZATION VALUES:
   ===================================
   MISCELLANEOUS - NORM VALUES
                   :
   -----------------------------------
   VCO NORMALIZATIONS
                   :
   -----------------------------------
   CF NORMALIZATIONS
                   :
   -----------------------------------
   REFERENCES
                   :
   -----------------------------------
   VERTICAL SCALE OFFSETS
                   :
   -----------------------------------
   LOG NORMALIZATIONS
                   :
   -----------------------------------
   FILTER SENSITIVITY
                   :
   -----------------------------------
   FILTER AMPLITUDES
                   :
   -----------------------------------
   VR FINE GAIN
                   :
   -----------------------------------
   VR GAIN STEPS
                   :
   -----------------------------------
   RF ATTEN, PREAMP & DET GAIN
                   :
   ===================================
   ";
```

OBWMode <arg>

Arguments: `ON`, `OFF`, `IDLE`

This single-argument command specifies occupied bandwidth measurement mode. The occupied bandwidth measurement mode is enabled if on or idle.

```
OBWMode ON

OBWMode OFF

OBWMode IDLE
```

OBWMode?

Arguments: None

This simple query returns the status of the occupied bandwidth measurement mode.

```
OBWMode?

    OBWMODE ON

    OBWMODE OFF

    OBWMODE IDLE
```

OBWPcnt <arg>

Arguments: Numeral in the range of 1 to 99

This single-argument command specifies the percentage (1% to 99%) of occupied bandwidth for occupied bandwidth measurements.

`OBWPcnt 40` *(for example)*

OBWPcnt?

Arguments: None

This simple query returns the occupied bandwidth percentage.

`OBWPcnt?`

`OBWPCNT 40` *(for example)*

OBWResult?

Arguments: None

This simple query returns the result of the most recent occupied bandwidth measurement (in hertz).

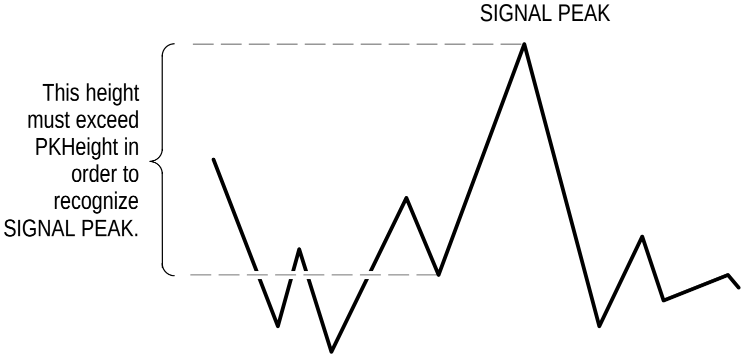
`OBWResult?`

`OBWRESULT 4.0E+6` *(for example)*

PKHeight <arg>

Arguments: Integer in the range 2 to 255

This single-argument command specifies how high a signal peak must be so it is recognized by the NEXT LOWER and NEXT HIGHER marker functions. The signal height is specified in vertical display increments relative to the nearest local minimum in its skirts. 20 is the default value. Units are not allowed.


SIGNAL PEAK
This height must exceed PKHeight in order to recognize SIGNAL PEAK.

```
PKHeight 50
```
(for example)

PKHeight?

Arguments: None

This simple query returns a number that represents the signal height in vertical display increments required for the peak to be recognized by the NEXT LOWER and NEXT HIGHER marker functions.

```
PKHeight?
    PKHEIGHT 20
```
(for example)

PLLmode <arg>

Arguments: `OFF`, `ON`

This single-argument command enables or disables the 1st LO phase lock system. If the `PLLmode` is on, the spectrum analyzer's 1st LO automatically phase locks for spans of 20 kHz/div or less.

```
PLLmode ON

PLLmode OFF
```

PLLmode? Arguments: None

This simple query returns the current on/off status of the 1st LO phase lock system.

```
PLLmode?

    PLLMODE ON

    PLLMODE OFF
```

PLOT? Arguments: None

This simple query returns a complex response from the spectrum analyzer that provides screen plot information for printing or plotting. The result of this command is similar to pressing the **[PLOT]** key on the front panel. The printer or plotter must speak the HPGL language or be compatible with Epson FX codes. The appropriate printer type must be specified locally or with the PTYpe command.

```
PLOT?

    <screen data array up to 61.1kbyte long>
```

The data array can be up to 61.1 Kbytes for Epson printers and 37 Kbytes for HPGL plotters. PLOT? never produces a response header, even if HDR is ON. See *Programming* for programming examples.

POFset <arg> Arguments: CENter, TOP

This single-argument command specifies whether to offset the result of the B,C MINUS A register arithmetic feature to the top or center of the display.

```
POFset CENter

POFset TOP
```

POFset? Arguments: None

This simple query returns information indicating whether the result of the B,C MINUS A function is offset to the center or top of the spectrum analyzer display.

```
POFset?

    POFSET CENTER

    POFSET TOP
```

PRDouts? Arguments: None

This simple query returns a list of the spectrum analyzer's on-screen readouts. There may be up to 14 readouts as listed below, depending on the status and mode of operation:

- Title
- Center or start frequency
- Reference level
- Span/division
- Resolution bandwidth
- Attenuation or marker/delta frequency/normalized noise/carrier-to-noise/occupied bandwidth/frequency count
- Video filter or marker/delta amplitude/noise bandwidth/dB down for bandwidth mode
- Vertical scale
- Video line/TV channel number/average count/D line/TV channel table
- Single sweep mode/arm
- CALIBRATOR
- UNCAL or FREQ COR OFF
- Real time clock

The readouts are returned as arguments enclosed in quotation marks (”) and separated by commas (,). The response ends in a semicolon (;). If an argument is missing, a null string (“”) is returned. In principal, each argument can be up to 32 characters long (the string length is dimensioned for a maximum of 14 × 32 = 448 characters), but this length is never achieved in practice. The query does not return the spectrum analyzer's general purpose message line, GPIB status line, or user-defined DISPLAY MESSAGE line.

The following response example is returned after initializing the spectrum analyzer to the factory defaults. See *Programming* for programming examples.

```
PRDouts?

    PRDOUTS "","900MHZ","20.0DBM",
        "180MHZ/MAX","5MHZ RBW","ATTN
        50DB","VF WIDE","10DB/","","","","","","";
```

PREamp <arg>

Arguments: OFF, ON

This single-argument command turns the built-in preamplifier on or off.

```
PREamp ON

PREamp OFF
```

PREamp?

Arguments: None

This simple query returns the current on/off status of the built-in preamplifier.

```
PREamp?

    PREAMP ON

    PREAMP OFF
```

PROTset <arg>

Arguments: OFF, ON

This single-argument command turns stored settings protection on or off. Stored settings cannot be erased when PROTset is ON.

```
PROTset ON

PROTset OFF
```

PROTset??

Arguments: None

This simple query returns the current on/off status of the stored settings protection. Protected settings cannot be erased.

```
PROTset?

    PROTSET ON

    PROTSET OFF
```

PSTep

Arguments: None

This is a command that requires no argument. It is equivalent to turning the frequency/markers knob one click in the clockwise direction. The spectrum analyzer’s response depends upon the currently selected knob function.

```
PSTep
```

PTYpe <arg>

Arguments: EPSON, HPGL2, HPGL4

This single-argument command specifies the type of printer or plotter encoding to use for screen data transferred from the spectrum analyzer to the controller in response to the PLOT? query.

```
PTYpe EPSON

PTYpe HPGL2

PTYpe HPGL4
```

PTYpe?

Arguments: None

This simple query returns the type of printer or plotter currently selected for use with the PLOT? query.

```
PTYpe?

    PTYPE EPSON

    PTYPE HPGL2

    PTYPE HPGL4
```

RECall <arg>

Arguments: Integers from 0 to 39 except 9, 19, and 29

This single-argument command instructs the spectrum analyzer to recall the stored settings in the location indicated by the argument. Integers between 0 and 39 (inclusive) are valid except for 9, 19, and 29.

RECall 0 *(for example)*

RECall 24 *(for example)*

REDout <arg>

Arguments: OFF, ON

This single-argument command turns the spectrum analyzer’s on-screen readouts on or off.

```
REDout ON

REDout OFF
```

REDout?

Arguments: None

This simple query returns the on/off status of the spectrum analyzer's on-screen readouts.

```
REDout?

    REDOUT ON

    REDOUT OFF
```

REFlvl <arg>

Arguments: `DEC`, `INC`, `ref level` in the range –70 to +20 dBm

This single-argument command increases, decreases, or sets the reference level. If `INC` or `DEC` is the argument, the command changes the reference level by 1 dB or 10 dB depending on the `FINe` command or the local 1 dB/10dB setting.

When a numeric argument is used, it must be within the range of –70 dBm to +20 dBm (or equivalent in alternate units of dBm, dBmV, dBV, dBμV, dBμW, or dBμV/M). If no units are used, the current reference level units are assumed. If units other than the current units are used, the value is converted to current units.

This command may alter the amount of RF attenuation if automatic RF attenuation is enabled. If LIN mode is active, the scale factor will be computed. All values are interpreted according to the current reference level offset and impedance correction.

```
REFlvl INC

REFlvl DEC

REFlvl 10 DBMV
```
(for example)

REFlvl?

Arguments: None

This simple query returns the current reference level in the currently selected reference level units.

```
REFlvl?

    REFLVL –35.0
```
(for example)

RESbw <arg>

Arguments: `INC`, `DEC`, bandwidth of resolution bandwidth filter

This single-argument command increases, decreases, or selects the resolution bandwidth.

If `INC` or `DEC` is the argument, the command changes the resolution bandwidth to the next installed filter value.

When a numeric argument is used, the installed resolution bandwidth filter closest to the value is selected. This command disables automatic RES BW selection. If units are not attached, hertz are assumed.

```
RESbw INC

RESbw DEC

RESbw 30 kHz  (for example)
```

RESbw?

Arguments: None

This simple query returns the currently selected resolution bandwidth in hertz.

```
RESbw?

    RESBW 3.0E+4  (for example)
```

RFAtt <arg>

Arguments: Number in the range 0 to 50

This single-argument command sets the RF attenuation to a fixed value between 0 and 50 dB in 2 dB steps. Values other than even integers are rounded. Units are not allowed.

```
RFAtt 34  (for example)
```

RFAtt?

Arguments: None

This simple query returns the current RF attenuation value in decibels (dB). This may be a fixed value or one that is automatically selected.

```
RFAtt?

    RFATT 34  (for example)
```

RLUnit <arg>

Arguments: `DBM`, `DBMV`, `DBV`, `DBUV`, `DBUW`, `DBUVM`

This single-argument command specifies the units for the reference level.

The `DBUVM` argument is not allowed under these conditions:

- Linear display mode
- DBUVM result is already saved
- Display source is the FM detector or external source
- There is a destination conflict with ensemble average, minimum hold, or display line

```
RLUnit DBM

RLUnit DBMV

RLUnit DBV

RLUnit DBUV

RLUnit DBUW

RLUnit DBUVM
```

In CATV mode, only `DBMV` is allowed.

RLUnit? Arguments: None

This simple query returns the selected reference level units.

```
RLUnit?

    RLUNIT DBM

    RLUNIT DBMV

    RLUNIT DBV

    RLUNIT DBUV

    RLUNIT DBUW

    RLUNIT DBUVM
```

ROFset <arg> Arguments: Value within the range ±100 dB

This single-argument command sets the reference level offset value. The offset value must be within the –100 dB to +100 dB range. Units are not allowed.

`ROFset -7.5` *(for example)*

ROFset? Arguments: None

This simple query returns the current reference level offset. If the offset is disabled, the query returns 0. The units are decibels (dB).

```
ROFset?
```

`ROFSET -7.5` *(for example)*

ROMode <arg>

Arguments: `OFF`, `ON`

This single-argument command turns the reference level offset on or off.

```
ROMode ON
```

```
ROMode OFF
```

ROMode?

Arguments: None

This simple query returns the current reference level offset mode, either on or off.

```
ROMode?

    ROMODE ON

    ROMODE OFF
```

RQS <arg>

Arguments: `OFF`, `ON`

This single-argument command enables and disables the generation of service requests (SRQ) by the spectrum analyzer. The user request is affected but the power-on SRQ is not.

```
RQS ON
```

```
RQS OFF
```

RQS?

Arguments: None

This simple query returns the spectrum analyzer's current on/off status of service request generation.

```
RQS?

    RQS ON

    RQS OFF
```

RS232 <arg>
(Requires RS-232 Interface)

Arguments: See following table

Arguments of RS232 and RS232?	Values
BAUd	110, 150, 300, 600, 1200, 2400, 4800, 9600
BITs	Number of data bits; 7 or 8
ECHo	Echo mode; ON or OFF
EOL	Query termination; CR, LF, or CRLf
FLOw	Flow control; HARd, SOFt, NONe
PARity	Parity; ODD, EVEn, or NONe
VERbose	Verbose mode; ON or OFF

This is a command that configures the RS-232 interface. The baud rate is set to the closest available value when baud rates other than those listed are entered. One stop bit is selected unless the baud rate is 110, in which case two stop bits are selected.

```
RS232 BAUd:9600
```

```
RS232 VERbose:ON
```

NOTE. *Although it is syntactically correct to send more than one argument per command, it is dangerous to do so. Execution of these commands reprograms the interface. Also note that when* PARity: NONe *is selected, the spectrum analyzer does not add a parity bit on the data word, nor does it expect a parity bit on input. Some terminals transmit an 8-bit word with the 8th bit set to 0 when set to “7-bit, no parity.” This can confuse the effort to ensure compatible settings between the spectrum analyzer and an external device. See the RS-232 program example in Programming.*

RS232? <arg>
(Requires RS-232 Interface)

Arguments: See table for RS232 command

This is a query that returns the current RS-232 parameter of the specified argument. If no argument is given, this command returns all RS-232 settings, separated by commas (,).

```
RS232? FLOw
```

```
    RS232 FLOW:HARD
```
(for example)

RST

Arguments: None

This is a device-dependent command that performs the same function as the GPIB DCL interface message (see Appendix B).

```
RST
```

RTIme <arg>

Arguments: Time in the form "HH:MM:SS"

This command sets the time of the real-time clock. The argument is a string in the above format where HH is the hour in 24-hour format, MM is minutes, and SS is seconds. The seconds are set to zero regardless of the argument given.

Note that the elements are separated by colons (:). Quotation marks (”) must be present. All fields must contain a value; a leading 0 is assumed if a single digit is used.

```
RTIme "13:30:00"
```

RTIme?

Arguments: None

This query returns the current time in the format HH:MM:SS where HH is the hour, MM is the minute, and SS is the seconds.

```
RTIme?
```

```
RTIME "13:30:27"
```

SAVe <arg>

Arguments: A:ON, A:OFF, B:ON, B:OFF, C:ON, C:OFF, and combinations of these arguments

This is a command with a single argument or multiple linked arguments. It saves and deletes waveforms located in NVRAM. ON saves the indicated display register to NVRAM. OFF deletes a saved display register from NVRAM. For instance, the command SAVe A:ON saves the current contents of the A-register to NVRAM and halts the A-register from updating. SAVe A:OFF permits the A-register to be updated each sweep.

Single or multiple arguments can be used in a single command:

SAVe A:ON *(for example)*

SAVe C:OFF *(for example)*

SAVe A:ON,B:OFF *(for example)*

SAVe A:ON,B:OFF,C:OFF *(for example)*

Using the SAVe <link>:OFF command with the destination register for an ensemble average or minimum hold operation terminates these operations.

The DBUVM destination register cannot be turned off without first exiting DBUVM mode.

SAVe? <arg>

Arguments: None, A, B, C

This is a query with one or no argument that returns the storage state of the indicated register or all registers. If no argument is used, the states of the A, B, and C registers are returned. If an argument is used, only the state of the indicated register is returned.

```
SAVe?
    SAVE A:ON,B:OFF,C:OFF
```
(for example)

```
SAVe? A
    SAVE A:ON
```
(for example)

```
SAVe? B
    SAVE B:OFF
```
(for example)

```
SAVe? C
    SAVE C:OFF
```
(for example)

SET?

Arguments: None

This simple query returns a group of command headers and arguments representing the current front panel and menu settings of the spectrum analyzer. The string of commands can be retained for transfer to the same or another 2714 or 2715 spectrum analyzer at a later time when it is desirable to reproduce the same setup. The SET? response enables you to replicate equipment setups. You should not modify the SET? response.

HDR status has no effect on the SET? query. Individual command headers are always returned and the group header (SET) is never returned. Each header and argument is separated by a semicolon (;) to ensure the response represents a functional message.

The SET? response is lengthy, but it is easy to interpret. A sample response follows that includes many of the command headers that can be returned. Some of the headers depend on the options installed on your instrument and its configuration at the time of the SET? query.

If the spectrum analyzer is not in single sweep mode or if display line is not ON, the SIGSWP and DLVALUE commands will be missing.

```
SET?

    VIEW WATERFALL:OFF;RECALL 1;DSRC AM;

    VRTDSP LOG: 10;DETECTOR OFF;CNBW 1.0E+0;

    BWNUM -3; OBWPCNT 99; ACQMODE MAXMIN;

    SPAN 180.E+6;MXSPN ON;ZEROSP OFF;

    CFSF CENTER;FREQ 900.E+6; FOMODE 0.E+3;

    PLLMODE ON;DISCOR OFF;VMANTTBL 1;

    VMDIST 3.0E+0;VMDEST C;VMMKRUNIT DBUVM;

    ROFSET -74.0;ROMODE OFF;RLUNIT DBM;

    WAIT;PREAMP OFF;MXRLVL -30;REFLVL 20.0;

    RFATT 50;ARFATT ON;FILE"",%
```

***NOTE**. If the SET? query is used while a CATV test is running, the returned settings string may contain invalid control settings. The CATV test may change a 2714 or 2715 control parameter while the SET? query is processed.*

SGErr <arg>

Arguments: OFF, ON

This single-argument command enables and disables the generation of a service request (SRQ) when a marker function is unable to find a signal. SGErr ON enables SRQ generation for event 896.

```
SGErr ON

SGErr OFF
```

SGErr?

Arguments: None

This simple query returns the on/off status of service request (SRQ) generation when a marker function cannot find the intended signal.

```
SGErr?

    SGERR ON

    SGERR OFF
```

SGSrch

Arguments: None

This is a command that requires no argument. It instructs the spectrum analyzer to perform a signal search between the current BEGIN and END frequencies for all signals greater than the threshold (see `THRhld`). The BEGIN and END frequencies can be set locally or by using the `SSBegin` and `SSEnd` commands. Results of the search are returned by `SSResult?`.

```
SGSrch
```

SGTrak <arg>

Arguments: `OFF`, `ON`

This single-argument command enables and disables the signal track mode. The command does not work in zero span mode.

```
SGTrak ON

SGTrak OFF
```

SGTrak?

Arguments: None

This simple query returns the on/off status of the signal track mode.

```
SGTrak?

    SGTRAK ON

    SGTRAK OFF
```

SIGswp

Arguments: None

This is a command that requires no argument. It selects and arms the single sweep mode. The sweep does not actually occur until the trigger conditions for the currently selected trigger mode are satisfied. Any `TRIGGER` command cancels single sweep mode.

```
SIGswp
```

SIGswp?

Arguments: None

This simple query returns the current status of the single sweep mode.

```
SIGswp?

    SIGSWP ON

    SIGSWP OFF

    SIGSWP ARM
```

SPAn <arg>

Arguments: `0`, `INC`, `DEC`, value in the range 1 kHz to 180 MHz

This single-argument command increases, decreases, or sets the span/division.

When used with the `INC` or `DEC` argument, the command changes the span/division in the indicated direction in the normal 1-2-5 sequence.

The span/division is set to the indicated value for numeric arguments other than zero. If the value is out of range, the end point is substituted and an SRQ and event code are generated. If the 0 (zero) argument is used, zero span mode is activated.

```
SPAn INC

SPAn DEC

SPAn 0

SPAn 25 kHz  (for example)
```

SPAn?

Arguments: None

This simple query returns the current span/div in hertz.

```
SPAn?

    SPAN 2.5E+4  (for example)
```

SSBegin <arg>

Arguments: Value in the range 9 kHz to 1.8 GHz

This single-argument command specifies the BEGIN frequency for signal search mode. The BEGIN frequency must be less than the END frequency. Units may be appended, otherwise hertz are assumed. The value is offset by `FOFfset` if `FOMode` is on.

```
SSBegin 54 MHz  (for example)
```

SSBegin?

Arguments: None

This simple query returns (in hertz) the currently specified BEGIN frequency for the signal search mode.

```
SSBegin?

    SSBEGIN 54.000e+6  (for example)
```

SSEnd <arg>

Arguments: Value in the range 9 kHz to 1.8 GHz

This single-argument command specifies the END frequency for the signal search mode. The END frequency must be greater than the BEGIN frequency. Units may be appended, otherwise hertz are assumed. The value is offset by `FOFfset` if `FOMode` is on.

```
SSEnd 300 MHz  (for example)
```

SSEnd?

Arguments: None

This simple query returns (in hertz) the currently specified END frequency for the signal search mode.

```
SSEnd?

    SSEND 300.00E+6  (for example)
```

SSResult?

Arguments: None

This simple query returns the number of signals detected during a signal search operation, and lists the frequency and amplitude of each signal. Up to 50 frequency/amplitude pairs can be returned. The frequency and amplitude values are separated by commas (,), and the pairs of values are also delimited by commas.

The list begins with the lowest-frequency signal detected and proceeds to the highest. The amplitude units are those currently selected as reference level units; frequency is in hertz.

Following is a typical example of the response when `HDR` is on.

```
SSRESULT <N>,<freq1>,<ampl1>,...,<freqN>,<amplN>;

where:
```

- N= number of signals detected (N ≤50)
- freq1,...,freqN = frequency of 1st,...,Nth detected signal
- ampl1,...,amolN= amplitude of 1st,...,Nth detected signal

If no signal is detected during the search, `SSResult?` returns zero (with `HDR ON`, the response is `SSRESULT 0;`). If the amplitude of a detected signal is off-screen, it is listed as 1.0E+6.

```
SSResult?

    SSRESULT 8,55.250E+6,7.3,...299.75E+6,-4.0;  (for example)
```

See *Programming* for programming examples.

STByte?

Arguments: None

This simple query returns the GPIB serial poll response byte. This command is only useful for instruments equipped with the RS-232 interface. If the query is received by an instrument equipped with the GPIB interface the value 0 is always returned.

```
STByte?
    STBYTE 61
```
(for example)

STEp <arg>

Arguments: `CF`, `MARker`, number in the range 1 Hz to 1.8 GHz

This single-argument command specifies the programmed frequency tuning increment. Specifying the increment also turns on the spectrum analyzer's programmed tuning mode. The `CF` argument selects the current center frequency as the increment, `MARker` selects the current marker frequency, and a numeric argument specifies the increment in hertz. Units may be appended.

```
STEp CF

STEp MARker

STEp 30 kHz
```
(for example)

STEp?

Arguments: None

This simple query returns the currently specified programmed frequency tuning increment in hertz.

```
STEp?
    STEP 3.0E+4
```
(for example)

STOre <arg>

Arguments: Integers 2 to 39 except 9, 19, and 29

This single-argument command stores the spectrum analyzer control settings in the location designated by the argument. Locations 0 and 1 are reserved for the last power-down and factory default power-up settings, respectively. Locations 9, 19, and 29 are invalid.

```
STOre 2

STOre 24
```
(for example)

STPinc <arg>

Arguments: AUTo, TABular, PROg

This single-argument command selects the tuning increment mode as automatic, tabular, or programmed. Refer to the STEp command to set the programmed increment.

```
STPinc AUTo

STPinc TABular

STPinc PROg
```

STPinc?

Arguments: None

This simple query returns the currently selected tuning increment mode.

```
STPinc?

    STPINC AUTO

    STPINC TABULAR

    STPINC PROG
```

STStop <arg>

Arguments: MARker or pair of values in range –10 MHz to 1.8 GHz

This single- or double-argument command sets the start and stop frequencies of the spectrum analyzer display. If MARker is the argument, the start and stop frequencies are set to the current marker frequencies (delta marker mode must be active).

When the argument is a pair of numbers, the first number specifies the start frequency and the second specifies the stop frequency. Units may be appended, otherwise hertz are assumed.

The second number must exceed the first by at least 10 kHz to satisfy the spectrum analyzer’s span requirements. If the second number is greater than the first, but by less than 10 kHz, the stop frequency is set to the start frequency plus 10 kHz.

The stop frequency may be set lower than the start frequency. Under these conditions the spectrum analyzer is tuned to the lower frequency and zero span mode is activated.

```
STStop MARker

STStop 192 MHz,198 MHz
```
(for example)

SURv <arg>

Arguments: `FASt:ON`, `FASt:OFF`

This single-argument command enables or disables the FAST AMPL carrier survey mode. This command is provided for compatibility with previous firmware.

FASt:ON will select the FAST AMPL carrier survey mode. FAST OFF will select the ACCURATE FREQ AND AMPL mode.

```
SURv FASt:ON

SURv FASt:OFF
```

SURv? <arg>

Arguments: `FASt`

This single argument query returns the on/off status of the FAST AMPL carrier survey mode.

```
SURv? FASt
```

`SURV FAST:ON –` if carrier survey mode is FAST AMPL

`SURV FAST:OFF –` if carrier survey mode is not FAST AMPL

TAMpl?

Arguments: None

This simple query returns the amplitude of the signal being tracked. The value is updated at the end of each sweep when signal track mode is enabled. Otherwise the amplitude of the signal last tracked is returned. The units are those currently selected as reference level units.

```
TAMpl?
```

`TAMPL –34.0` *(for example)*

TEXt <arg>

Arguments: <string>

This single-argument command displays a message on line 8 of the spectrum analyzer screen (line 9 if title mode is active). The message is the argument of the command (up to 32 characters). Quotation marks (”) must be used. Only uppercase characters can be displayed, although lowercase characters may be sent. Transmit a null string (`TEXt ""`) to erase the message.

`TEXt "MY MESSAGE"` *(for example)*

TEXt?

Arguments: None

This simple query returns the current contents of the message buffer in the spectrum analyzer. Lowercase letters are returned if lowercase letters were originally sent, even though the on-screen message is displayed in uppercase letters.

```
TEXt?

    TEXT "MY MESSAGE"
```
(for example)

TFReq?

Arguments: None

This simple query returns the frequency (in hertz) of the signal being tracked. The value is updated at the end of each sweep when signal track mode is enabled. Otherwise the amplitude of the signal last tracked is returned.

```
TFReq?

    TFREQ 101.36E+6
```
(for example)

THRhld <arg>

Arguments: Number within the range –174 dBm to 20 dBm

This single-argument command specifies the value of the threshold above which the spectrum analyzer automatically detects signals. Units of dBm, dBmV, dBV, dBμV, dBμW, or dBμV/M can be used, but the equivalent value must be within the range –174 dBm to +20 dBm. If units are not supplied, the current reference level units are assumed. This command also turns off the automatic threshold selection mode.

```
THRhld –10 DBMV
```
(for example)

THRhld?

Arguments: None

This simple query returns the current threshold value (fixed or automatically selected). The units are the currently selected reference level units.

```
THRhld?

    THRHLD –10.0
```
(for example)

TIMe <arg>

Arguments: `INC`, `DEC`, value in the range 1 μs to 2 s

***NOTE**. The* TIMe *argument's range is limited to 100 μs to 2 s when any of the Display Storage registers (A, B, C, D) are active.*

This single-argument command increases, decreases, or sets the sweep speed. This command also turns off the spectrum analyzer's automatic sweep speed selection. Units of ns, μs, ms, or s may be appended; otherwise seconds are assumed.

If the `INC` or `DEC` argument is used, the sweep speed is increased or decreased in the normal 1-2-5 sequence. Sweep times that are not in the sequence are increased to the next valid setting.

Numeric arguments must be within the range of 1 microsecond to 2 seconds. Sweep times less than 100 microseconds are not permitted in display storage mode.

```
TIMe INC

TIMe DEC

TIMe 25 US
```
(for example)

TIMe?

Arguments: None

This simple query returns the currently selected sweep speed. Units are in seconds.

```
TIMe?

    TIME 25.E−6
```
(for example)

TIMMode <arg>

Arguments: `AUTo`, `MANual`, `FIXed`

This single-argument command enables (`AUTo`) and disables (`FIXed`) automatic sweep speed selection. This command also enables manual sweep positioning (`MANual`) using the spectrum analyzer's LEVEL control. When in the `FIXed` mode the sweep speed is controlled locally or with the `TIMe` command.

```
TIMMode AUTo

TIMMode FIXed

TIMMode MANual
```

TIMMode?

Arguments: None

This simple query returns the current time base mode.

```
TIMMode?

    TIMMODE AUTO

    TIMMODE FIXED

    TIMMODE MANUAL
```

TITLe <arg>

Arguments: <string>

This single-argument command displays a title on line 1 of the spectrum analyzer screen. The title is the argument of the command which may be up to 32 characters long. Quotation marks (”) must be used. Only uppercase characters can be displayed. A null string (`TITLe ""`) is transmitted to erase the title. Use the `TTLMode` command to turn the title on or off.

`TITle "SCREEN 1"` *(for example)*

TITLe?

Arguments: None

This simple query returns the spectrum analyzer screen title if one currently exists. If the title was sent in lowercase letters, the returned string will be lowercase even though the title is displayed in uppercase letters on the spectrum analyzer screen.

```
TITLe?
```

`TITLE "SCREEN 1"` *(for example)*

TMOde <arg>

Arguments: `FREquency`, `MARker`, `VIDline`

This single-argument command selects the frequency/marker knob function.

Argument	Function
FREquency	Adjust start or center frequency
MARker	Adjust marker frequency
VIDline	Select video line number if knob selectable, TV line triggering is enabled

```
TMOde FREquency

TMOde MARker

TMOde VIDline
```

TMOde?

Arguments: None

This simple query returns the currently selected function of the frequency/markers knob.

```
TMOde?

    TMODE FREQUENCY
    TMODE MARKER
    TMODE VIDLINE
```

TOPsig

Arguments: None

This command with no argument instructs the spectrum analyzer to change the reference level to the amplitude of the primary marker. A marker must be enabled.

```
TOPsig
```

TRIgger <arg>

Arguments: `EXTernal`, `FRErun`, `INTernal`, `LINe`, `TVField`, `TVLine`

This single-argument command selects the trigger type. `TVLine` also sets the knob function to VIDLINE if knob-selectable TV line mode is enabled.

```
TRIgger EXTernal

TRIgger FRErun

TRIgger INTernal

TRIgger LINe

TRIgger TVField

TRIgger TVLine
```

TRIgger?

Arguments: None

This simple query returns the selected spectrum analyzer trigger type.

```
TRIgger?

    TRIGGER FRERUN

    TRIGGER EXTERNAL

    TRIGGER INTERNAL

    TRIGGER LINE

    TRIGGER TVFIELD

    TRIGGER TVLINE
```

TTLMode <arg>

Arguments: `OFF`, `ON`

This single-argument command turns the spectrum analyzer screen title on and off.

```
TTLMode ON

TTLMode OFF
```

TTLMode?

Arguments: None

This simple query indicates whether the spectrum analyzer's screen title is being displayed (ON) or not displayed (OFF).

```
TTLMode?

    TTLMODE ON

    TTLMODE OFF
```

TUNe <arg>

Arguments: Frequency in the range –1.8 GHz to +1.8 GHz

This single-argument command changes the start or center frequency by the amount of the argument. The resultant frequency must remain within the range of –10 MHz to 1.8 GHz. Units may be appended; otherwise hertz are assumed.

`TUNe 10.8 MHz` *(for example)*

TVLine <arg>

Arguments: Integer in the range 1 to 1024

This single-argument command specifies the number of the TV line to which the spectrum analyzer is to trigger when programmed TV line triggering is enabled. This command also turns off Video Monitor mode if it is enabled. The minimum argument is 1. The maximum value depends on the TV line standard: 525 for NTSC, 625 for PAL and SECAM, and 1024 for OPEN. This command also sets `TRIgger` to `TVLine` and enables the programmed mode.

`TVLine 17` *(for example)*

TVLine?

Arguments: None

This simple query returns the (integer) TV line number to which the spectrum analyzer is to trigger when TV line trigger mode is selected.

```
TVLine?

    TVLINE 17
```
(for example)

TVLMode <arg>

Arguments: `CONT`, `KNOB`, `PROg`

This single-argument command designates the specific TV line trigger mode, and enables TVLine trigger mode. This command turns the Video Monitor mode off if enabled. The arguments and their functions are described in the following table.

Argument	Function
CONT	Trigger on every line
KNOB	Trigger line number selected with the frequency/markers knob
PROg	Trigger line number entered locally or with the `TVLine` command

`TVLMode KNOB` *(for example)*

TVLMode? Arguments: None

This simple query returns the currently selected TV line trigger mode.

```
TVLMode?

    TVLMODE CONT

    TVLMODE KNOB

    TVLMODE PROG
```

TVLStd <arg> Arguments: `NTSC`, `OPEN`, `PAL`, `SECAM`

This single-argument command designates the TV standard when using the TV line trigger mode. This command turns the Video Monitor mode off if enabled, and sets the trigger mode to TV Line. The maximum value of the `TVLIne` command's argument is influenced by the TV standard.

```
TVLStd NTSC

TVLStd OPEN

TVLStd PAL

TVLStd SECAM
```

TVLStd? Arguments: None

This simple query returns the currently selected TV standard.

```
TVLStd?

    TVLSTD NTSC

    TVLSTD OPEN

    TVLSTD PAL

    TVLSTD SECAM
```

UDPDir?

Arguments: None

This simple query returns a list of all currently loaded UDPs in the form `UDPDIR "<index 1>","<title 1>","<index 2>", "<title 2>"`....

```
UDPDir?

    UDPDIR "0","PROG 0","6", "TEST"
```

VDMode <arg>

Arguments: `BROadcast`, `SATellite`

This single-argument command designates the type of detection to be used in the Video Monitor mode. `BROadcast` selects AM detection for use with broadcast television; `SATellite` selects FM detection for use with satellite transponders.

```
VDMode BROadcast

VDMode SATellite
```

VDMode?

Arguments: None

This simple query returns the currently selected detection for Video Monitor mode.

```
VDMode?

    VDMODE BROADCAST

    VDMODE SATELLITE
```

VFEnab <arg>

Arguments: `OFF`, `ON`

This single-argument command turns the video filter on and off.

```
VFEnab ON

VFEnab OFF
```

VFEnab?

Arguments: None

This simple query returns the current on/off status of the video filter.

```
VFEnab?

    VFENAB ON

    VFENAB OFF
```

VFMode <arg>

Arguments: AUTo, FIXed

This single-argument command enables (AUTo) and disables (FIXed) automatic selection of the video filter bandwidth. When fixed mode is first entered, the automatically selected video filter bandwidth is made the current fixed filter bandwidth. A new fixed filter bandwidth can then be selected locally or by using the VIDflt command.

```
VFMode AUTo

VFMode FIXed
```

VFMode?

Arguments: None

This simple query indicates whether the video filter bandwidth is fixed or automatically selected by the spectrum analyzer.

```
VFMode?

    VFMODE AUTO

    VFMODE FIXED
```

VIDflt <arg>

Arguments: OFF, ON, floating point number

This single-argument command turns the video filter on or off, or specifies the filter bandwidth.

The ON and OFF arguments enable and disable the currently selected video filter in the same way as the VFEnab command. A numeric argument is used to specify a particular video filter bandwidth and turn on the video filter. Units may be appended, otherwise hertz are assumed. The video filter bandwidths follow a 1-3 sequence. The video filter bandwidth closest to the specified filter width is selected.

```
VIDflt ON

VIDflt OFF

VIDflt 30 kHz
```
(for example)

VIDflt?

Arguments: None

This simple query returns the currently selected video filter bandwidth in hertz whether or not the filter is enabled.

```
VIDflt?

    VIDFLT 3.0E+4
```
(for example)

VIEw <arg>

Arguments: `A:ON`, `A:OFF`, `B:ON`, `B:OFF`, `C:ON`, `C:OFF`, `D:ON`, `D:OFF`, `Minusa:ON`, `Minusa:OFF`, `Waterfall:ON`, `Waterfall:OFF`, combinations of the above

This is a command with single- or multiple-linked arguments that enables and disables digital display mode in the indicated register. For instance, `VIEw A:ON` turns on the digital display in the A-register; `VIEw A:OFF` turns off the A-register. Single or multiple arguments can be used in a single command. See the following examples:

`VIEw B:ON` *(for example)*

`VIEw A:ON,B:OFF` *(for example)*

`VIEw A:ON,B:OFF,C:OFF` *(for example)*

`VIEw Waterfall:ON,A:OFF` *(for example)*

Multiple arguments must be separated with commas (,).

VIEw?

Arguments: None, `A`, `B`, `C`, `D`, `Minusa`, `Waterfall`

This is a query with one or no argument that returns the on/off status of the indicated register, or all storage registers. If no argument is used, the status of the A, B, C, and D registers, the waterfall mode, and B,C minus A display mode are returned. Only the state of the indicated register is returned when an argument is used.

`VIEw?`

`VIEW Waterfall:OFF,A:ON,B:OFF,C:OFF,`

`D:ON,Minusa:OFF` *(for example)*

`VIEw? B`

`VIEW B:OFF` *(for example)*

VMAnttbl <arg>

Arguments: Integer in the range 1 to 5

This single-argument command designates by number the antenna table to be used for dBμV/M measurements. Units are not allowed.

`VMAnttbl 3` *(for example)*

VMAnttbl?

Arguments: None

This simple query returns the number of the antenna table currently selected for use when making dBµV/M measurements.

```
VMAnttbl?
```

`VMANTTBL 3` *(for example)*

VMDEst <arg>

Arguments: `A`, `B`, `C`

This single-argument command designates the spectrum analyzer display register used as the destination for dBµV/M measurements.

`VMDEst B` *(for example)*

VMDEst?

Arguments: None

This simple query returns the currently selected destination register for dBµV/M measurements.

```
VMDEst?
```

`VMDEST B` *(for example)*

VMDIst <arg>

Arguments: Floating point number

This single-argument command specifies the source-antenna distance at which a dBµV/M measurement is actually performed. Distance may be entered in feet (`FT`), meters (`M`), kilometers (`KM`), or miles (`MI`), but the spectrum analyzer converts all values to meters or kilometers.

`VMDIst 3 M` *(for example)*

VMDIst?

Arguments: None

This simple query returns the currently specified source-antenna distance for dBµV/M measurements. Units are meters.

```
VMDIst?
```

`VMDIST 3.0` *(for example)*

`VMDIST 3.0` *(for example)*

VMMkrunit <arg>

Arguments: `DBUVM`, `VM`

This single-argument command specifies the amplitude units for marker readouts in dBµV/M mode as dBµV/M or volts/m.

```
VMMkrunit DBUVM

VMMkrunit VM

VMMkrunit?

Arguments: None
```

VMMkrunit?

This simple query returns the currently selected units for marker readouts in the dBµV/M mode.

```
VMMkrunit?

    VMMKRUNIT DBUVM

    VMMKRUNIT VM
```

VMOnitor <arg>

Arguments: `OFF`, `ON`

This single-argument command turns the Video Monitor on and off.

```
VMOnitor ON

VMOnitor OFF
```

VMOnitor?

Arguments: None

This simple query returns the current on/off status of the Video Monitor mode.

```
VMOnitor?

    VMONITOR ON

    VMONITOR OFF
```

VPOlarity <arg>

Arguments: `NEGative`, `POSitive`

This single-argument command specifies the polarity of video signals to be received with the Video Monitor mode.

```
VPOlarity NEGative

VPOlarity POSitive
```

VPOlarity?

Arguments: None

This simple query returns the currently specified video signal polarity for Video Monitor mode.

```
VPOlarity?

    VPOLARITY NEGATIVE

    VPOLARITY POSITIVE
```

VRTdsp <arg>

Arguments: `LOG:<num>`, `LIN:<num>`, `FM:<num>`, `EXTernal:<num>`

This command with a single linked argument specifies the vertical scale factor for the indicated display mode. The `DSRC` command must be used to enter FM or EXTernal modes, although the `VRTdsp` command still sets the scale factor.

Link	<num>	Units	Display Type
LOG:	10, 5, 1	dB/div	logarithmic
LIN:	10.8 to 342.33*	µV/div, mV/div	linear
FM:	10, 5, 1	kHz/div	linear
EXTernal:	17.5,87.5,175	mV/div	linear

*** Corresponds to reference level range of –70 dBm to +20 dBm**

If units are not supplied, `LIN` assumes volts (for example, `LIN:.1 = LIN:100 mV`), `FM` assumes hertz, and `EXTernal` assumes volts.

`VRTdsp LOG:5` *(for example)*

`VRTdsp LIN:50 µV` *(for example)*

`VRTdsp FM:5 kHz` *(for example)*

`VRTdsp EXTernal:175E–3` *(for example)*

VRTdsp? <arg>

Arguments: None, `LOG`, `LIN`, `FM`, `EXTernal`

This query has either one or no argument. It returns a linked response. The current scale factor is returned when `VRTdsp?` is used without an argument. When used with an argument, `VRTdsp?` returns the scale factor used when the indicated mode was last entered. Units are decibel (dB) for `LOG`, volts (v) for `LIN` or `EXTernal`, and hertz (Hz) for `FM`.

```
VRTdsp?

    VRTDSP LOG:5  (for example)

VRTdsp? LOG

    VRTdsp? LOG:5  (for example)

VRTdsp? LIN

    VRTDSP LIN:50.0E–3  (for example)

VRTdsp? FM

    VRTDSP FM:5.E+3  (for example)

VRTdsp? EXTernal

    VRTDSP EXTERNAL:175.E–3  (for example)
```

VSYnc <arg>

Arguments: NEGative, POSitive

This single-argument command specifies the polarity of the video sync to be received with the Video Monitor mode.

```
VSYnc NEGative

VSYnc POSitive
```

VSYnc?

Arguments: None

This simple query returns the currently specified video sync polarity.

```
VSYnc?

    VSYNC NEGATIVE

    VSYNC POSITIVE
```

WAIt

Arguments: None

This is a command that requires no argument. It causes the spectrum analyzer to wait for an end-of-sweep to occur before processing any more commands. WAIt can be cancelled by the DCL or SDC GPIB commands, or the RS-232 BREAK command. See also the EOS command.

```
WAIt
```

WAVfrm?

Arguments: None

This simple query is functionally equivalent to the combination of the `WFMpre?` and `CURve?` queries. The `WAVfrm` header is never returned, even if `HDR` is on.

```
WAVfrm?
```

WFMpre <arg>

Arguments: `ENCdg:Asc`, `ENCdg:Bin`, `ENCdg:Hex`, `WFId:A`, `WFId:B`, `WFId:C`, `WFId:D`

This is a command with one or more linked arguments. It is used to designate the source/destination register for the `CURve` command/query, and the data encoding to be used during a `CURve` transfer. Multiple arguments separated by commas (,) may be used. See the `CURve` command for an explanation of data encoding.

In its simplest form an argument(s) always follows the command header. These are examples of the general form to be used:

```
WFMpre WFID:<register>

WFMpre ENCdg:<type>

WFMpre WFID:<register>,ENCdg:<type>
```

where:

```
<register> = A, B, C or D

<type> = Asc, Bin, or Hex
```

For instance, these are all possible `WFMpre` commands:

```
WFMpre WFId:A

WFMpre WFId:B

WFMpre WFId:C

WFMpre WFId:D

WFMpre ENCdg:Asc

WFMpre ENCdg:Bin

WFMpre ENCdg:Hex

WFMpre WFId:D,ENCdg:Asc
```

The last command string is a typical message. It indicates that, in this example, register D is the source/destination for future `CURve` transfers, and ASCII encoding is to be used for the data.

WFMpre? <arg>

Arguments: None, `ENCdg`, `WFId`

This is a query with one or no argument whose response provides information necessary for these `CURve` operations:

- Determine the currently selected source/destination register for `CURve` transfers
- Determine the data encoding for `CURve` transfers
- Interpret the result of a `CURve` query

When `WFMpre?` is used with the `WFId` argument, the query returns the currently selected source/destination register.

```
WFMpre? WFId

    WFMPRE WFID:B  (for example)
```

When `WFMpre?` is used with the `ENCdg` argument, the query returns the currently selected `CURve` data encoding.

```
WFMpre? ENCdg

    WFMPRE ENCDG:ASC  (for example)
```

When `WFMpre?` is issued without an argument, it returns all the information necessary to interpret the response to a `CURve?` query. Following is an example of the response with `HDR ON`:

```
WFMPRE WFID:<register>,ENCDG:<type>,

NR.PT:512,PT.FMT:Y,PT.OFF:<nr1>,XINCR:<nr3>,

XZERO:<nr3>,XUNIT:<xunit>,YOFF:<nr1>,

YMULT:<nr3>,YZERO:<nr3>,YUNIT:<yunit>,

BN.FMT:RP,BYT/NR:1,BIT/NR:8,CRVCHK:

CHKSMO, BYTCHK:None
```

The response identifies the waveform, specifies data encoding, and provides the offsets, scale factors, and units necessary to plot or interpret the curve data. Tables 4–7 and 4–8 define the terms in the response.

```
WFMpre?

    WFMPRE WFID:A,ENCDG:BIN,NR.PT:512,

    PT.FMT:Y,PT.OFF:5,XINCR:3.6e+6,XZERO:0.000,

    XUNIT:HZ,YOFF:245,YMULT:3.333E–1,YZERO:
```

```
20.000E+0,YUNIT:DBM,BN.FMT:RP,BYT/NR:1,
BIT/NR:8,CRVCHK:CHKSM0,BYTCHK:NONE;
```
(for example)

Table 4–7: Arguments of the WFMpre? Query

Argument	Name	Description
WFID:<id>	Waveform ID	ID may be A, B, C or D.
ENCDG:<enc>	Encoding	The possibilities are ASC, BIN or HEX.
NR.PT:512	Number of points	There are 512 data points on every 2714 or 2715 curve.
PT.FMT:Y	Point format	Only Y-data is transmitted. X-data is implicit by the position of the point.
PT.OFF:<nr1>	Point offset	See following formulas.
XINCR:<nr3>	X increment	See following formulas.
XZERO:<nr3>	X zero	See following formulas.
XUNIT:<xunit>	X units	Hz (hertz) or s(seconds).
YOFF:<nr1>	Y offset	See following formulas.
YMULT:<nr3>	Y multiplier	See following formulas.
YZERO:<nr3>	Y zero	See following formulas.
YUNIT:<yunit>	Y units	dBm, dBmV, dBV, dBμV, dBμW, dBμV/M, V or Hz.
BN.FMT:RP	Binary format	A binary positive integer.
BYT/NR:1	Bytes per number	Always 1 byte per number.
BIT/NR:8	Bits per number	Always 8 significant bits.
CRVCHK:CHKSM0	Curve checksum	Last byte of a binary or hex transfer is 2's complement of the modulo-256 sum of bytes following header that starts block; header is % (binary block); #H (ASCII hex block).
BYTCHK:NONE	Byte check	No per-byte error checking.

Table 4–8: Related Formulas

Let `<valN>` = value of the N^{th} data point of the `CURVE` query. Then the X-value of that point is computed as:

XN = XZERO + XINCR X (N – PT.OFF)

The Y-value is computed as:

YN = YZERO + YMULT X (<valN> – YOFF)

The following is an example of the WFMpre? query and the response obtained for the factory default power-up settings.

Using the preceding preamble (`WFMPRE WFID:A,ENCDG......`), we will compute the value of a data point within a `CURve?` response. In this example we have chosen the 255th point and have assumed that the `CURve?` response indicates an integer value of 125 (curve data always have integer values). From the preceding formulas, the X and Y values of the point are given by these expressions:

XN (in xunits) = XZERO + XINCR * (N – PT.OFF)

and

YN (in yunits) = YZERO + YMULT * (VALN –YOFF)

where N = 255 and VALN = 125.

This data is extracted from the preamble:

XZERO = 0	YZERO = 2.0×10^{1}
XINCR = 3.6 X 10^{6}	YMULT = 3.333×10^{-1}
PT.OFF = 5	YOFF = 245
XUNIT = Hz	YUNIT = dBm

Evaluating the expressions, we find these results:

XN = 0 + 3.6 X 10^{6} * (255 – 5) = 900×10^{6} Hz

YN = 20 + .3333 * (125 – 245) = –20 dBm

These results represent screen center for the factory default power-up settings.

ZERosp <arg>

Arguments: `OFF`, `ON`

This single-argument command turns the zero span mode on and off.

```
ZERosp ON

ZERosp OFF
```

ZERosp?

Arguments: None

This simple query returns the current on/off status of the zero span mode.

```
ZERosp?

    ZEROSP ON

    ZEROSP OFF
```

Status Reporting

Status Reporting

Most of this section applies to the GPIB interface (Option 03), because RS-232 protocol does not include device serial polling or an SRQ function. The first subsection, *RS-232 Error Reporting,* provides RS-232 status reporting information. If your instrument has RS-232 (Option 08) installed, review the entire section to understand the function of the status byte and its relationship to SRQ, RQS, and serial polling.

The 2714 or 2715 reports its status to the controller using the status byte and event codes. The status byte is a reporting feature provided in the IEEE-488.1 (GPIB) standard. When the 2714 or 2715 is serially polled by the controller, the 2714 or 2715 places a status byte representing its general condition on the data bus. This status byte is then read by the controller. Event codes are generated by the 2714 or 2715 for transmission to the controller over the data bus in response to an instrument-specific `EVEnt?` or `ERRor?` query.

NOTE*. The 2714 or 2715 supports serial polling; it does not support parallel polling.*

The status byte and event codes can indicate normal or abnormal conditions. Most often they alert you to abnormal conditions. Decoding of the status byte often provides the most efficient approach to detecting general classes of instrument conditions, but event codes provide more detailed information.

Different types of information are produced by decoding status bytes and event codes. For example, suppose the signal on the 2714 or 2715's screen is larger than the current reference level. If the `MMAx` command is used to place the marker on the signal peak, an error condition occurs. The status byte following the `MMAx` command (1110 0000) indicates only that a "device-dependent failure or warning" condition exists, while the event code (810) indicates "signal out of range" (see Table 5–1).

Two techniques are available for monitoring the status of the spectrum analyzer:

- Issue an `EVEnt?` or `ERRor?` query and interpret the numerical response.
- Serially poll the instrument and decode the status byte.

All Tektronix GPIB instruments support the GPIB service request function. When these instruments detect an abnormal condition, they assert (pull to its low state) the dedicated service request (SRQ) line of the GPIB interface management bus to indicate that attention is needed. Therefore, you can construct a subroutine that monitors the SRQ line to classify abnormal conditions. Either of the foregoing techniques can then be used to determine why the SRQ was generated.

RS-232 Error Reporting

Verbose Mode

The RS-232 protocol does not contain a mechanism that duplicates the GPIB SRQ function described later in this section. To fill this need, the RS-232 configuration supplies a VERBOSE mode. This mode is used most often when a “dumb” terminal is connected to the spectrum analyzer. When VERBOSE mode is on, every command issued from the terminal is guaranteed a response from the spectrum analyzer. Three response types are possible:

- The string `"OK"`, returned for a successful command
- A query response, returned for a successful query
- `ERR n`; returned when error number *n* is detected

Errors reported while VERBOSE mode is on have no effect on the status reporting structure described later for GPIB. The RS-232-specific query `STByte?` will return the GPIB serial poll response byte for analysis. Because `STByte?` is a normal query, the Busy bit (bit 5 of the status byte) is always reported to be on.

A query is the only means for returning information to the interface when VERBOSE mode is `OFF`. In this mode, the user must issue an `EVENT?` or `STBYTE?` query to retrieve error information.

Reporting and Clearing Errors

The REQUEST indicator on the 2714 or 2715’s display screen indicates when an error is pending. If `RQS` is on and an error is pending, the REQUEST indicator appears on the screen. The programmer must send an

```
STByte?
```

query (simulating the return of the GPIB serial poll response) followed by either

```
EVEnt? or ERRor
```

to report and clear the error condition.

If `RQS` is off and an error is pending, the REQUEST indicator does not appear on the 2714 or 2715’s screen. Under these conditions an `EVEnt?` or `ERRor?` query is required to report and clear the error condition. Otherwise the error remains pending.

To use the VERBOSE feature, a routine must be set up that reads each possible response and sends each response type for parsing and possible processing. See the RS-232 sample program in *Programming*. This routine uses VERBOSE mode in place of GPIB SRQ.

Parity, Framing, and Overrun Errors

There are three types of errors that are related to RS-232 communications: parity, framing, and overrun. Parity and framing errors may result from mismatches between configuration settings (baud rate, parity, etc.) or from noise on the system. Overrun errors are often caused by a design problem, such as when interrupts cannot be handled at the required rate.

When an error occurs, the appropriate event is declared and all unparsed data are discarded until a terminator is received. Table 5–1 lists the complete set of 2714 or 2715 event codes and status bytes, including three RS-232-specific event codes (numbers 410, 411, and 412).

The status bytes listed in Table 5–1 assume that the Busy bit is off. When the Busy bit is on, add values of 16 (decimal) or 10 (hexadecimal) to the table entry.

Table 5–1: Event Codes

Event Code	Status Byte		Event Description
	Dec	Hex	
0	128	80	No Device-dependent Status To Report
0	0	00	No System Status To Report
101	97	61	Command Header Error
102	97	61	Header Delimiter Error
103	97	61	Command Argument Error
104	97	61	Argument Delimiter Error
105	97	61	Non-numeric Arg. (Numeric Expected)
106	97	61	Missing Argument
107	97	61	Invalid Message Unit Delimiter
108	97	61	Binary Block Checksum Error
109	97	61	Binary Block Byte Count Error
121	97	61	Illegal Hex Character
122	97	61	Unrecognized Argument Type
123	97	61	The Argument Is Too Large
124	97	61	Non-binary Arg. (Binary Or Hex Expected)
151	97	61	Illegal Response Value In Query
201	98	62	Remote Command Received When In Local Mode
202	98	62	Command Aborted – (RTL) Return To Local
203	98	62	I/O Deadlock Detected
205	98	62	Argument Out Of Range
206	98	62	Group Execute Trigger Ignored

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
252	98	62	System Error (Illegal Command)
253	98	62	Integer Overflow (Range 0–65535)
371	99	63	Output Buffer Full (Too Many Queries)
372	99	63	Input Buffer Full (Command Too Long)
401	65	41	Power On
403	67	43	User Request or CATV Prompt
410	99	63	RS-232 Parity Error
411	99	63	RS-232 Framing Error
412	99	63	RS-232 Hardware Overrun
700	224	E0	Error
701	224	E0	Illegal Parameter Passed
704	224	E0	Illegal Command
705	224	E0	Malloc: Ran Out Of Memory
706	224	E0	RunTask: Cannot Start process
707	224	E0	Interrupt Fault At FF
708	224	E0	Interrupt Fault
709	224	E0	Command Not Implemented
710	224	E0	Markers Are Off
711	224	E0	Signal Cannot Be Set Properly
712	224	E0	No Signal At Counter Input
713	224	E0	Counter Frequency Unstable
714	224	E0	Normalization Suggested
715	224	E0	Timer Interrupt Fault
716	224	E0	No Signal (Normalizations)
717	224	E0	Ampl Out Of Range (Normalizations)
718	224	E0	Freq Out Of Range (Normalizations)
719	224	E0	Func Not Avail In Current Mode
720	224	E0	Frequency Normalization Failed
721	224	E0	Amplitude Normalization Failed
722	224	E0	Reference Normalization Failed
723	224	E0	Internal Ref Freq Too Inaccurate

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
724	224	E0	Internal Ref Ampl Too Inaccurate
725	224	E0	Selected Stored Setting Is Empty
726	224	E0	Video Monitor Not Installed
727	224	E0	Satellite Video Monitor Not Installed
728	224	E0	Not Installed
729	224	E0	Counter Not Installed
730	224	E0	Cannot Overwrite Saved Display
731	224	E0	NVM Checksum Error
732	224	E0	Non-Compatible NVM Format
733	224	E0	First Step Must Be Done First
734	224	E0	FREQ Norm Suggested (Inner PLL)
735	224	E0	FREQ Norm Suggested (Set VCO)
736	224	E0	Polynomial Has No Solution
737	224	E0	Last Pwr Down Reg Checksum Err
738	224	E0	Storage Register Empty
739	224	E0	Normalized Result Out of Range
740	224	E0	Function Not Avail. In LIN mode
741	224	E0	Cannot Store – NV Memory Full
742	224	E0	AMPL Norm Suggested (VR Pin DAC)
743	224	E0	Cannot Calc. Vert. Sensitivity
744	224	E0	Cannot Count (VCO,IF)
745	224	E0	Cannot Normalize PLL VCO
746	224	E0	Cannot Count Beat Frequency
747	224	E0	FREQ Norm Suggested (Set Beat)
748	224	E0	FREQ Norm Suggested (1st LO)
749	224	E0	Setting Corrupted
750	224	E0	NVM Fragmentation Error
751	224	E0	NVM Segmentation Error
752	224	E0	Comm Port Not Installed
753	224	E0	Real Time Clock Hardware Failure
754	224	E0	Real Time Clock Not Installed

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
755	224	E0	FREQ Norm Suggested (Find Side)
756	224	E0	FREQ Norm Suggested (Span DAC)
759	224	E0	Insufficient Memory Available
760	224	E0	Not Avail In Short Holdoff Mode
761	224	E0	Short Holdoff Mode Not Installed
762	224	E0	Cannot Overwrite Stored Setting
763	224	E0	Cannot Overwrite Stored Waveform
764	224	E0	Delete Existing Program First
765	224	E0	Editing Buffer Is Empty
766	224	E0	Remove Protection First
767	128	80	Wait Aborted, Sweep Not Armed
768	224	E0	Selected Program Is Empty
769	224	E0	Program Not Executable
770	224	E0	Not Avail In Waterfall Mode
771	224	E0	Amplitude Out Of Calibration
772	224	E0	Illegal Start/Stop/Inc Values
773	224	E0	Delete Existing Table First
774	224	E0	Selected Table Is Empty
775	224	E0	Use ANTENNA SETUP Menu First
776	224	E0	Table Is Too Large To Edit
777	224	E0	Default Data Loaded
778	224	E0	Delete Editing Buffer First
779	224	E0	Warning: Using Empty Antenna Table
780	224	E0	Not Available With DBUV/M Idle
781	224	E0	Mkr Would Overwrite Noise Value
782	224	E0	Function Not Avail In DBUV/M Mode
783	224	E0	No Listener
784	224	E0	Select TALK ONLY Mode First
786	224	E0	QP Filters Not Installed
787	224	E0	Destination Waveform Conflict
788	224	E0	TG Normalization Suggested

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
790	229	E5	Input Buffer Empty (Firmware Error)
791	229	E5	Illegal Event Code (Firmware Error)
792	229	E5	Illegal Command Received From CP
793	229	E5	Illegal Byte Count In Command
801	224	E0	Out Of Range
802	224	E0	None Of The Traces Are Active
803	224	E0	Uncal Off
804	224	E0	Uncal On
805	128	80	Single Sweep Mode
806	128	80	Single Sweep Armed
807	128	80	Single Sweep Trigger
808	224	E0	No Signal Found Above Threshold
809	224	E0	Inactive Marker Off Screen
810	224	E0	Signal Over Range
811	224	E0	Function Not Avail In Max Span
812	224	E0	Ref Level At New Range Limit
813	224	E0	Normalization Complete
814	224	E0	No Signal At Center Of Display
815	224	E0	Not Avail W/ Display Storage On
816	224	E0	500 kHz RBW Used For Counting
817	224	E0	Noise Level Less Than 2 dB
818	224	E0	Start Frequency Changed
819	224	E0	Stop Frequency Changed
820	224	E0	Signal Out Of IF Passband
821	224	E0	No Modulation On Signal
822	224	E0	1st Measurement Complete
823	224	E0	Disconnect Input Signal
824	224	E0	ZERO SPAN Entered
825	224	E0	Must Be In Delta Marker Mode
826	224	E0	Stand By
827	224	E0	Printer Error

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
828	224	E0	Printer Out Of Paper
829	224	E0	Printer Is Not Connected
830	224	E0	Port Off Line
831	128	80	Formatting Plot
832	224	E0	Plot Aborted
833	224	E0	Can’t Count With Corrections Off
834	224	E0	Counter Signal Out Of IF Passband
835	224	E0	Vert Mode/Scale Mismatch On Diff
836	224	E0	Query Not Available
837	224	E0	Average Noise Too Low
838	224	E0	Only Waveforms Saved
839	224	E0	Only Waveforms Deleted
840	224	E0	File System Full
841	224	E0	File System Directory Full
842	224	E0	File Size Error
843	224	E0	Too Many Files Open
844	224	E0	File Not Found
845	224	E0	Protected File
846	224	E0	Cannot Delete File While in Use
847	224	E0	Additional NVRAM Not Installed
848	224	E0	Invalid File Number
849	224	E0	Invalid Device Number
850	224	E0	End of File
851	224	E0	NVM Version Mis-Match
852	224	E0	Fatal Error In File
853	224	E0	Directory Error In File
854	224	E0	Data Error In File
855	224	E0	Table In Use
856	128	80	Clear Event
857	224	E0	Calibrator Doesn’t Match Readout
858	128	80	Return to Local Request

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
859	224	E0	Display Line Off Screen
860	224	E0	DBUV/M Measurement Mode Idle
861	224	E0	Search Terminated, Max Signals
862	128	80	Lock Event
863	128	80	Unlock Event
864	128	80	DCL End
865	128	80	User Defined Program in Process
866	128	80	Plot In Process
867	128	80	Average In Process
868	128	80	Signal Search In Process
869	128	80	Normalizing
870	128	80	CATV Function In Process
871	224	E0	Entering CATV Measurement Mode
872	224	E0	Results Storage Complete
873	224	E0	Short YIG Delay. Go to UTIL/5/4
874	224	E0	Quiet Line Verification Failed
875	224	E0	1st Mix LVL Raised for Vid Demod
880	194	C2	User Defined Program Complete
881	194	C2	Plot Complete
882	194	C2	Ensemble Average Complete
883	194	C2	Signal Search Complete
884	194	C2	Normalization Process Finished
885	194	C2	End of Sweep Detected
886	194	C2	CATV Function Complete
895	228	E4	Display Line Limit Exceeded
896	227	E3	Signal Find Error
900	224	E0	Func Not Available On This Chan Type
901	224	E0	CATV Measurement Failure
902	224	E0	No Reference Defined
903	224	E0	Cannot Overwrite Stored Reference
904	224	E0	Leave Interactive Mode First

Table 5–1: Event Codes (Cont.)

Event Code	Status Byte		Event Description
	Dec	Hex	
905	224	E0	Flatness Corrections Not Avail
906	224	E0	1st Mxr Must Be ≤ –30dB For EMC
907	224	E0	Dspl Line and A, C-A Incompatible
908	224	E0	Not Avail With Current Appl Mode
909	224	E0	No beat frequencies are defined
913	224	E0	No Quiet Video Lines Found
914	224	E0	No Test Line Number Specified

The Service Request

The 2714 or 2715 has complete support for the IEEE-488.1 (GPIB) service request function (see *Appendix B: GPIB System Concepts*). SRQs may be created by an instrument on the GPIB in response to certain equipment states including all abnormal conditions. However, SRQ generation may also be disabled with the RQS command. In addition to this general capability for inhibiting SRQs, certain SRQs may be independently masked.

The following events can generate SRQs:

- Pressing the front panel key sequence **[UTIL] [6].** This is the user request event. The resulting SRQ cannot be independently masked.
- Completion of certain operations. End-Of-Sweep, normalization, User-Defined routine, signal search, plot, or ensemble average. Intermediate operation complete events are blocked. The plot complete event is generated after a plot is formatted and sent to the plotter, not necessarily when the plotter is finished. These SRQs are masked using the EOS command.
- Device-dependent failures or warnings. These SRQs include all 2714 or 2715 error messages that appear on-screen. They cannot be independently masked.
- Power up SRQ mode enabled. The 2714 or 2715 generates an SRQ at power up when the power-up SRQ is enabled (requires GPIB interface). Press the key sequence **[UTIL] [4] [0] [0] [2]** to implement this mode.
- Illegal commands. Command, execution, or internal errors produce SRQs that cannot be independently masked.
- Failure of find-signal commands. An SRQ is generated when the MMAx, MRGTnx, MLFtnxt, HRAmpl, and LRAmpl commands fail to locate a signal. This SRQ can be masked with the SGErr command.

- Crossing the display line limit. An SRQ is enabled when the display line limit detector is on (DLLimit) and a signal crosses the threshold. This SRQ cannot be independently masked.
- Internal hardware/firmware errors. These errors generate an SRQ that cannot be independently masked.

Status Byte

What is a Status Byte?

The status byte usually provides information about instrument conditions according to certain categories. These include normal, abnormal, busy, command error, execution error, etc. This information is different from data returned by the ERRor? and EVEnt? queries, which provide detailed information about the cause of the current status. For instance, the response to an EVEnt? query might describe what kind of error or warning prompted the 2714 or 2715 to assert SRQ and report abnormal status.

The status byte is stored in the status byte register, which is updated as conditions in the 2714 or 2715 change. The spectrum analyzer responds to a serial poll by placing the status byte on the data bus (see *Appendix B: GPIB System Concepts*) to be read by the controller. Only the most recent status byte is placed on the bus. The status byte is cleared by a serial poll of the spectrum analyzer, by the DCL GPIB command, or (if the instrument is first addressed) the SDC GPIB command.

Types of Status Bytes

Status bytes can be divided into two categories: device-dependent and system. Device-dependent status bytes represent conditions unique to the 2714 or 2715. System status bytes have a fixed definition for all Tektronix devices and identify most events common to an IEEE-488.1 system. Bit 8 is always set (1) for device-dependent status bytes and is always reset (0) for system status bytes. Tables 5–2 through 5–5 show general and specific encodings of device-dependent and system status bytes.

Because of the status coding scheme, only one condition can be reported in a single status byte. More than one condition may exist in the 2714 or 2715 at one time. Therefore, the following rules are used to determine which condition is reported by a status byte:

- Each condition is assigned a priority according to Table 5–6. Only one occurrence of each condition is retained in memory.
- When RQS is on, the status byte following an SRQ represents the condition that caused the SRQ (not necessarily the highest priority).
- When RQS is off, the normal device-dependent status shown in Table 5–5 is reported.

- After a status byte is read by the controller, the status byte is updated with the highest priority condition which has not yet been reported.
- All status conditions that have occurred but have not been reported (except power on) can be cleared by a device clear (DCL) command.
- Bit 5 is the current status of the message processor.

Table 5–2: General System Status Bytes

Status Byte Bits	8	7	6	5	4	3	2	1
Bit Value	0	R	E	B	S	S	S	S

Where:

R = SRQ pending (1 = pending, 0 = not pending)
B = Instrument busy (1 = busy, 0 = not busy)
S = System status
E = Normal (0)/Abnormal (1)

Table 5–3: General Device-Dependent Status Bytes

Status Byte Bits	8	7	6	5	4	3	2	1
Bit Value	1	R	D	D	D	D	D	D

Where:

R = SRQ pending (1 = pending, 0 = not pending)
D = Instrument-specific

Table 5–4: Specific System Status Bytes

Status Byte Bits								Byte* Value	Event Description
8	7	6	5	4	3	2	1		
0	0	0	B	0	0	0	0	00,10	No status to report
0	1	0	B	0	0	0	1	41,51	Power on
0	1	0	B	0	0	1	1	43,53	User Request
0	1	1	B	0	0	0	1	61,71	Command error
0	1	1	B	0	0	1	0	62,72	Execution error
0	1	1	B	0	0	1	1	63,73	Internal error

*** Byte value depends on B = 1 or B = 0**

Table 5–5: Specific Device-Dependent Status Bytes

Status Byte Bits								**Byte* Value**	**Event Description**
8	**7**	**6**	**5**	**4**	**3**	**2**	**1**		
1	0	U	B	N	S	A	P		Normal device-dependent status
1	1	0	B	0	0	1	0	C2,D2	Device-dependent operation complete (EOS, Norm, etc.)
1	1	1	B	0	0	0	0	E0,F0	Device-dependent failure/warning
1	1	1	B	0	0	1	1	E3,F3	Signal find error (MFBIG, MRGTNX, etc.)
1	1	1	B	0	1	0	0	E4,F4	Display line limit exceeded
1	1	1	B	0	1	0	1	E5,F5	Firmware error

*** Byte value depends on B = 1 or B = 0**

Where:

U = UDP or CATV measurement executing (1)
N = Normalization(s) executing (1)
S = Signal search executing (1)
A = Ensemble average executing (1)
P = Plot executing (1)
B = Busy (1)

Table 5–6: Event Priorities

Event	**Priority***
Power on	1
Command error	2
Execution error	3
Internal error	4
User request	5
Signal find error (`MRGtnx`, etc.)	6
Display line limit exceeded	6
Device-dependent failure or warning	7
Device-dependent operation complete (EOS, Norm, etc.)	8
Firmware error	9
Normal device dependent status	10
No status to report	10

*** Highest priority = 1**

Example 5-1 shows a QuickBASIC subroutine that can be used with the National Instruments GPIB board and software to report the current status byte. The first five lines must be part of the parent program. IBFIND and IBRSP are callable subroutines supplied by National Instruments. See *Programming* for explanations and additional programming instructions.

Example 5-1 Subroutine to Read the Status Byte

```
REM $INCLUDE 'IBDCL4.BAS'

COMMON SHARED BDNAME$,BD%,SPR$

BDNAME$ = "TEK_SA"

CALL IBFIND (BDNAME$, BD%)

GOSUB SERIAL.POLL

             :

             :

SERIAL.POLL:

   CALL IBRSP (BD%,SPR%)

   PRINT SPR%

RETURN
```

Event Codes

Returning Error and Status Information

Not all GPIB applications need the SRQ function and serial poll sequence capabilities. In fact, the SRQ service routine is often more complex than the application demands. For this reason the Request for Service (RQS) command is implemented in the 2714 or 2715. This command allows the controller to prevent the 2714 or 2715 from asserting SRQ. In this mode of operation, the EVEnt? and ERRor? queries request error and status information.

The EVEnt? and ERRor? queries return the same codes. ERRor? is included for compatibility with Tektronix 490-Series spectrum analyzers. These queries provide more information about the cause of an event than the status byte does. For this reason the event query is useful in the RQS ON and RQS OFF modes.

Event codes are grouped into categories as shown in Table 5–7. Individual event codes are listed near the front of this section in Table 5–1.

Table 5–7: Event Code Categories

Numeric Range	Event
0 – 99	Local events (not used)
100 – 199	Command errors
200 – 299	Execution errors
300 – 399	Internal errors
400 – 499	System events
500 – 599	Execution warnings (not used)
600 – 699	Internal warnings (not used)
700 – 999	Spectrum analyzer dependent events

Example 5-2 shows a QuickBASIC subroutine that can be used with the National Instruments GPIB board and software to report all pending event codes. However, when RQS is on, SERIAL.POLL must be called first to ensure that a valid event code is returned. Further, the response header must be turned off so EVENT.CODE$ is returned exclusively as a number string. IBFIND, IBWRT, and IBRD are callable subroutines supplied by National Instruments. See *Programming* for explanations and additional programming instructions.

Example 5-2 Subroutine for Reading Event Codes

```
REM $INLUDE 'IBDCL4.BAS'

COMMON SHARED BDNAME$, BD%,EVENT.CODE$

BDNAME$ = "TEK_SA"

CALL IBFIND (BDNAME$, BD%)

GOSUB EVENT.FIND

             :

EVENT.FIND:

   WRT$ = "HDR OFF;EVE?"

   DO

      CALL IBWRT (BD%,WRT$)

      CALL IBRD (BD%,EVENT.CODE$)

      PRINT EVENT.CODE$

   WHILE VAL (EVENT.CODE$) <> 0

RETURN
```

Event Code Priorities

Event codes are assigned priorities according to Table 5–6, but only the first event code of a priority is accumulated in the pending event table. However, the `EVEnt?` and `ERRor?` queries return a single code. Therefore, to ensure that all pending event codes are reported, the `EVent?` or `ERRor?` queries must be issued continuously until event code zero (0) is returned.

`EVEnt?` or `ERRor?` returns the highest priority event in the table when `RQS` is off. After RQS has been turned on, either query returns the code corresponding to the event reported in the status byte (not necessarily the highest priority).

When an event code is read, the code is cleared from the pending event table. This action does not clear the status byte. In a similar manner, reading a status byte clears it from the table, but the event code is not cleared. In either `RQS` mode, the DCL or (if the instrument is first addressed) `SDC` GPIB commands may be used to clear all event codes except POWER ON.

Programming

Programming

This section discusses general approaches to RS-232 and GPIB applications on IBM-compatible MS-DOS/PC-DOS computers. The intent is not to teach you programming, but to provide a few useful subroutines and demonstration programs. Experienced programmers may not need to read this section.

There are no uniform standards for all GPIB boards and programming languages. The use of RS-232 is more standardized across the IBM/PC family. However, the examples in this section will be both device- and language-specific.

The programming language used for either interface is Microsoft's QuickBASIC, Version 4.5.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Introduction to RS-232 Programming

The routines in the *RS-232 Program Examples* section can also work with earlier versions of QuickBASIC. However, you must make the changes to function calls as outlined in the `READ-QB.DOC` document file from National Instruments.

No special libraries, device drivers, or third-party software is required for compiling and running the RS-232 programming examples that follow. All functionality of the program is provided by Microsoft QuickBASIC. For instructions on how to compile the following programs, refer to the documentation for QuickBASIC.

After a program example has been compiled, an executable file (with a .EXE extension) will be created. To run the program, simply type the name of the executable program and press the <ENTER> key on your computer keyboard.

Before attempting to run the RS-232 programming examples, connect a null modem cable between the COM1 serial port on your computer and the RS-232 port on the 2714 or 2715. The 2714 or 2715 RS-232 port must also be enabled and configured properly. This information is provided with each programming example in this section. Refer to *RS-232 Operation (Option 08)* on page 1–1 for detailed information on selecting RS-232 communications parameters.

RS-232 Program Examples

This section contains two RS-232 program examples:

- Talk/Listen Demo Program (Example 6–1)
- CATV XMOD Command Demo Program (Example 6–2)

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when these programs are executed. If the 2714 or 2715 is displaying a menu when either program is executed, the program will not run properly.*

The Talk/Listen Demo Program accepts single commands or queries, and performs the command function or returns the query response.

The CATV XMOD Command Demo Program executes the `CATV XMOD` command, and displays the test data in a single-column format.

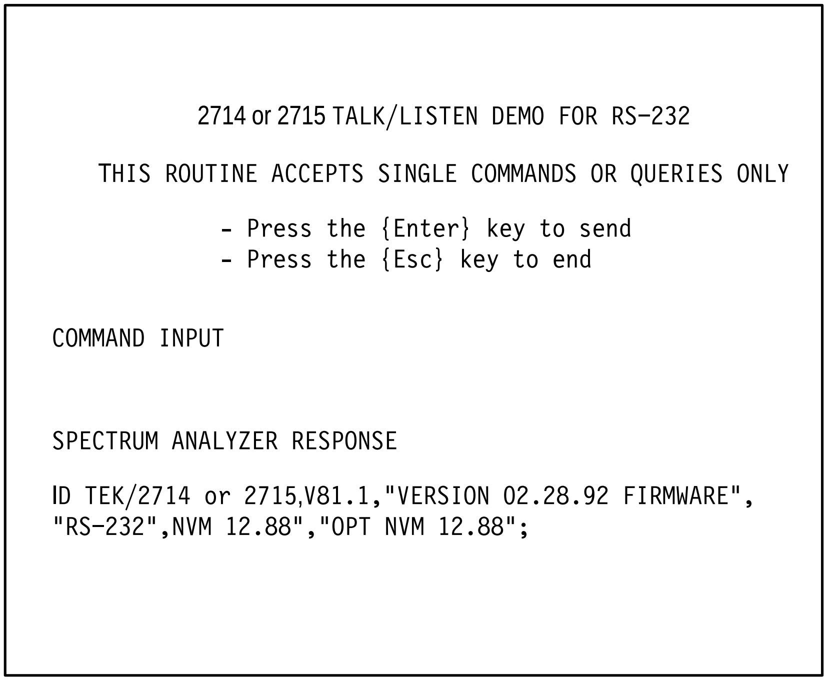
These interactive programs include extensive error checking and a friendly human interface. The code clearly defines program variables and their use, and subroutine linkage. The code also contains useful comments that clarify program requirements.

Each example program configures the 2714 or 2715's RS-232 interface to use the VERBOSE mode (see *Verbose Mode and Error Handling* on page 1–9). As a result, the 2714 or 2715 sends a response back to the program for each command it receives. The program terminates normally if the operator enters **[ESC]** from the computer keyboard. Otherwise, the program analyzes and displays each possible VERBOSE mode response and takes one of the following actions:

- If a command results in the response `OK`, the program displays the response and waits for the next operator entry (Example 6–1), or the program executes the `CATV XMOD` command (Example 6–2).
- If a command or query results in an error, the program retrieves and displays the 2714 or 2715's response, `ERR`, followed by the associated event code. It then awaits the next operator entry.
- If a query does not result in the response `ERR`, the program displays the spectrum analyzer's query response and awaits the next operator entry.

Talk/Listen Demo Program

Program Example 6–1 illustrates interactive control of the 2714 or 2715 with the RS-232 interface. It accepts single commands or queries, and performs the command function or returns the query response. As an example, Figure 6–1 shows how the terminal display appears after sending the `ID?` query when this program is running.



```
          2714 or 2715 TALK/LISTEN DEMO FOR RS-232

   THIS ROUTINE ACCEPTS SINGLE COMMANDS OR QUERIES ONLY

            - Press the {Enter} key to send
            - Press the {Esc} key to end


COMMAND INPUT



SPECTRUM ANALYZER RESPONSE

ID TEK/2714 or 2715,V81.1,"VERSION 02.28.92 FIRMWARE",
"RS-232",NVM 12.88","OPT NVM 12.88";
```

Figure 6–1: Terminal Display of ID? Query Response

The program declares the following procedures:

ENTERCOMMAND

This procedure formats the computer’s display screen to accept operator input and calls SENDCOMMAND. The call to this routine can be replaced by a call to any user-written module, for example, to perform a harmonic distortion analysis. All RS-232 communications are already in place.

SENDCOMMAND

This procedure sends commands to the spectrum analyzer and receives responses from the spectrum analyzer after establishing communications over the RS-232 interface. It also calls RS232.CALLS to perform actual RS-232 communications.

RS232.CALLS

This procedure performs all input/output for RS-232 communications between the controller and the 2714 or 2715. It also checks for communication errors and displays error messages.

Program Example 6–1 performs the following functions:

1. It opens `COM1` and establishes RS-232 communications with the spectrum analyzer using this configuration:

Baud rate:	9600
Data bits:	8
Stop bits:	1
Parity:	None
Verbose:	ON
Echo mode:	OFF
Terminator:	CR LF
Flow control:	NONE

   If the 2714 or 2715 is not configured properly, the program displays an error message. Refer to the description of the `RS232` command in *Command and Query Definitions* and the RS-232 installation procedure in *Introduction To Programming* for more information on RS-232 configuration.

2. The program accepts a spectrum analyzer command that the operator enters at the computer keyboard, and sends the command to the spectrum analyzer.

3. The program terminates when the operator presses the **[ESC]** key on the computer keyboard. This returns control to D0S.

4. The program always displays the 2714 or 2715's VERBOSE response to the command or query, and awaits further operator input.

***NOTE**. The* `SUB RS232.CALLS` *subroutine in this program is identical to the* `SUB RS232.CALLS` *subroutine used in Example 6–2.*

The 2714 or 2715 must have a spectral display on its screen when this program is executed. If the 2714 or 2715 is displaying a menu when the program is executed, it will not run properly.

Example 6–1 Talk/Listen Demo Program

```
'**************************************************************
' This is an example of an interactive program allowing an      *
' operator to communicate with the 2714 or 2715 spectrum analyz-
er.      *
' Commands can be sent to the 2714 or 2715 and responses are read
     *
' back. All communication is via RS-232 and is contained in     *
' the procedure RS232.CALLS and in the main module sub-         *
' routine READ.BUFFER. Communications are assumed to be         *
' passed through COM PORT 1.                                    *
' ***************************************************************
' Communications are established with a baud rate of 9600,      *
' data bits set to 8, parity of NONE, EOL=CRLF, FLOW CONTROL =  *
' NONE, ECHO = OFF, and VERBOSE=ON.                             *
' ***************************************************************
'  **** PROGRAM EXPECTS THE 2714 or 2715 TO BE SET WITH THESE
      *
'       PARAMETERS ****                                         *
' If the 2714 or 2715 is not configured in this way an error
message    *
' is displayed directing the operator to configure the          *
' 2714 or 2715 properly.
      *
'****************************************************************
DECLARE SUB SENDCOMMAND ()
DECLARE SUB ENTERCOMMAND ()
DECLARE SUB RS232.CALLS ()
'
'=================================================
' global variables used for communications linkage
'=================================================
'
COMMON SHARED rd$, wrt$, func%, errflg$,
   end.of.read$, buffer$, wfm%()
'
'===========================================
' ARRAY to hold waveform acquired from 2714 or 2715
'===========================================
'
DIM SHARED wfm%(511)
'
'------------------------------
' begin program execution here
'------------------------------
'
BEGIN.PROGRAM:
'
```

```
'======================
' set up the ESCape key
'======================
'
KEY(15) OFF
KEY 15, CHR$(&H0) + CHR$(&H1)
ON KEY(15) GOSUB END.PROGRAM
KEY(15) ON
'
'==============================
' Set up the communications key
'==============================
'
ON COM(1) GOSUB READ.BUFFER
'
'=========================================================
' Now execute routine which will accept all user input and
' display all responses. Procedure continues until the user
' presses the ESCape key.
'=========================================================
'
CALL ENTERCOMMAND
'
'========================================================
'end program and return to DOS when ESCape key is pressed
'========================================================
'
END.PROGRAM:
'
COLOR 7, 0
CLS
END
'
'**************************************************
'*                                                *
'* read routine for RS232 branched from RS232.    *
'* CALLS subprogram                               *
'*                                                *
'**************************************************
'
'===========
READ.BUFFER:
'===========
'
COM(1) OFF                         'turn automatic branch off
'
rd$ = ""                           'initialize string to save
```

```
'                                   response from 2714 or 2715
'
IF INSTR(wrt$, "NORM") THEN        'set time limit for reading
'                                   response from 2714 or 2715
   hold.i = 40                     'normalize query takes longer
'                                   to respond
ELSEIF func% = 5 THEN              'binary waveform transfer may
'                                   take longer
   hold.i = 15
ELSE                               '10 seconds will be enough
'                                   for others
   hold.i = 10
END IF
'
i = TIMER                          'initialize counter
'                                   and
DO WHILE TIMER < i + hold.i        'try to read for time alotted
'                                   make sure controller is just
  GOSUB READ.INPUT                 'not too fast for 2714 or 2715
'
  IF end.of.read$ = "Y" THEN
      i = 0
  END IF
LOOP
'
end.of.read$ = "Y"                 'set flag so will avoid
'                                   possible endless loop in
'                                   RS232 calls
RETURN
'
'==========
READ.INPUT:
'==========
'
buffer$ = "N"                      'initialize flag to indicate
'                                   that read something
DO WHILE NOT EOF(1)
     a$ = INPUT$(LOC(1), #1)       'read entire contents of
'                                   buffer
     rd$ = rd$ + a$                'and accumulate response
     buffer$ = "Y"                 'set buffer flag on so know
                                    have read something
LOOP
'
'****************************************************************
'                                                               *
' Following code segment distinguishes between a binary         *
' waveform transfer and any other response from the 2714 or 2715.
     *
' Binary transfers can contain embedded cr/lf characters so     *
' must ignore them until get past the binary data               *
'                                                               *
'*****************************************************************
'
IF func% <> 5 THEN                        'use first cr/lf
'                                          encounter for EOF
   IF INSTR(rd$, CHR$(10)) THEN           'on everything but
'                                          binary files
       IF INSTR(rd$, CHR$(13)) THEN       'insure that read
           end.of.read$ = "Y"             'EOF (cr/lf)
           buffer$ = "Y"                  'then set buffer flag
'                                          on to indicate
       END IF                             'successful read
   END IF
ELSE
   IF LEN(rd$) > 523 THEN                 'insure have read past
'                                          binary data
      IF INSTR(524, rd$, CHR$(13)) THEN   'before checking for
'                                          cr/lf
          IF INSTR(525, rd$, CHR$(10)) THEN
              end.of.read$ = "Y"
              buffer$ = "Y"               'set buffer flag on
          END IF
      END IF
   END IF
END IF
'
RETURN
'
'*****************************************************************
' PROCEDURE to perform a number of tasks:                       *
'     1) format the screen to accept operator input             *
'     2) open communication link with 2714 or 2715
      *
'        (func%=1,call RS232.CALLS)                             *
'     3) accept input                                           *
'     4) link to procedure which sends input to 2714 or 2715
      *
'        (SENDCOMMAND)                                          *
'                                                               *
' Procedure continues looping until the ESCape key is pressed   *
' The ESCape key ends program execution and returns operator    *
' to DOS.                                                       *
'*****************************************************************
```

```
SUB ENTERCOMMAND
'
'====================================================
' fill in the upper portion of screen with help info
'====================================================
'
SCREEN 0
COLOR 0, 0
CLS
COLOR 12, 0
LOCATE 1, 26, 0
PRINT "2714 or 2715 TALK/LISTEN DEMO FOR RS-232";
COLOR 14, 9
FOR i% = 3 TO 8
   LOCATE i%, 6
   PRINT STRING$(70, " ");
NEXT
'
LOCATE 4, 15, 0
PRINT "THIS ROUTINE ACCEPTS SINGLE COMMANDS OR
   QUERIES ONLY.";
LOCATE 6, 23, 0
PRINT "- Press the {Enter} key to send  ";
LOCATE 7, 23, 0
PRINT "- Press the {Esc} key to end ";
'
'===================================
' define a window for operator input
'===================================
'
COLOR 12, 0
LOCATE 9, 8, 0
PRINT "COMMAND INPUT";
COLOR 14, 9
FOR i% = 10 TO 17
   LOCATE i%, 6
   PRINT STRING$(70, " ");
NEXT
'
'========================================
' define a window for instrument response
'========================================
'
COLOR 12, 0
LOCATE 18, 8, 0
PRINT "SPECTRUM ANALYZER RESPONSE";
COLOR 14, 9
```

```
FOR i% = 19 TO 25
   LOCATE i%, 6
   PRINT STRING$(70, " ");
NEXT
'
'======================
'set up linkage to 2714 or 2715
'======================
'
func% = 1
CALL RS232.CALLS
'
'=====================================================
' initialize string which holds communication message
'=====================================================
'
wrt$ = ""
'
'==================================================
' now accept user input which will be sent to 2714 or 2715
'==================================================
'
LOCATE 11, 8, 1
'======
INLOOP:
'======
IN$ = INKEY$
IF IN$ = "" THEN
       GOTO INLOOP
ELSEIF LEN(IN$) <> 1 THEN
       BEEP
       GOTO INLOOP
ELSEIF IN$ = CHR$(27) THEN
       COLOR 7, 0
       CLS
       END
ELSEIF IN$ = CHR$(13) AND LEN(wrt$) = 0 THEN
'                                   carriage return with no
'                                   message
       BEEP                        'is an error
       GOTO INLOOP
ELSEIF IN$ = CHR$(13) THEN         'carriage return is the signal
'                                   to
       CALL SENDCOMMAND            'send user input to 2714 or
2715
       wrt$ = ""                   're-initialize string holding
'                                   input
```

```
       FOR x% = 11 TO 16           're-initialize input area on
'                                   screen
           LOCATE x%, 8: PRINT STRING$(66, " ");
       NEXT
       LOCATE 11, 8, 1             'reposition cursor and
       GOTO INLOOP                 'return to accept more input
ELSEIF IN$ = CHR$(8) AND LEN(wrt$) = 0 THEN
       BEEP                        'backspace without a place
'                                   to go
       GOTO INLOOP                 'is an error
ELSEIF IN$ = CHR$(8) THEN
       GOSUB BACKSPACE
       GOTO INLOOP
ELSE
       wrt$ = wrt$ + IN$           'accumulate input message
'                                   from user
       x = POS(0): y = CSRLIN      'one byte at a time
       LOCATE y, x: PRINT IN$;     'and reposition cursor
       x = x + 1                   'increment position of cursor
       IF x = 74 AND y = 16 THEN   'check cursor position
           FOR index% = 11 TO 16   'if outside realm of input
'                                   window
                 LOCATE index%, 8  'clear window
                 PRINT STRING$(66, " ");
           NEXT
           LOCATE 11, 8, 1         'and relocate cursor
           GOTO INLOOP
       ELSEIF x = 74 THEN          'if cursor beyond line
           CurRow% = CSRLIN        'move it down to next line
           LOCATE CurRow% + 1, 8, 1
           GOTO INLOOP
       ELSE
           GOTO INLOOP             'if cursor position OK just
       END IF                      'return to accept more input

END IF
'
'****************************************************************
' Routine to back up cursor and erase character. This is a      *
' DESTRUCTIVE backspace routine.                                *
'****************************************************************
'=========
BACKSPACE:
'=========
'
x = POS(0)
y = CSRLIN
```

```
'
IF x = 8 AND y = 11 THEN  '
             BEEP                  'cannot backspace beyond
'                                   area of window
             RETURN           '
ELSEIF x = 8 THEN
             y = y - 1             'backspace to end of
'                                   previous line
             LOCATE y, 73          'and locate cursor
ELSE
             LOCATE y, x - 1       'locate cursor one position
'                                   back
END IF
'
PRINT " " + CHR$(29);              'erase the character and then
'                                   back up
wrt$ = LEFT$(wrt$, LEN(wrt$) - 1)  'remove one character from the
'                                   string of saved input
'
RETURN
END SUB
'****************************************************************
' This procedure performs all RS232 calls.  All information     *
' needed has been defined as global variables:                  *
'         rd$ = returned info from RS232 call                   *
'         wrt$ = info to send to device                         *
'         func% = identifies which RS232 function to perform    *
'         wfm% = array to hold waveform                         *
'****************************************************************
SUB RS232.CALLS
'
IF func% = 1 THEN
          GOTO SELECT.DEVICE
ELSEIF func% = 3 THEN
          GOTO SEND.MESSAGE.TO.DEVICE
ELSEIF func% = 5 THEN
          GOTO GET.BINARY.WAVEFORM
END IF
'
'****************************************************************
'  This routine opens COM PORT 1 for communication via RS232,   *
'  opened for random so can handle both input and output        *
'****************************************************************
'  SPEED = 9600 (bits per second-also called BAUD RATE)         *
'  PARITY = NONE (parity bit not used)                          *
'  DATA = 8 (number of data bits per byte)                      *
'  STOP = 1 (number of stop bits)                               *
'  EOL = CRLF (carriage return/line feed as terminator)         *
'  FLOW CONTROL = NONE  (no "handshaking")                      *
'  ECHO = OFF (only needed for terminal emulation)              *
'  VERBOSE = ON (guarantees a response for each communication   *
'****************************************************************
' If the 2714 or 2715 has not been configured to agree with the
above   *
' parameters, the program will display an error message and     *
' force the user to change the 2714 or 2715. Program will not
execute   *
' otherwise.                                                    *
'****************************************************************
'  CS  =  0  (suppress checking the CLEAR TO SEND line)         *
'  DS  =  0  (suppress checking the DATA SET READY line)        *
'  RB  =  2048  (size in bytes of the receive buffer            *
'  TB  =  2048  (size in bytes of the transmit buffer)          *
'****************************************************************
'
SELECT.DEVICE:
'
buffer$ = "N"                      'initialize these two
'                                   indicators
read.error% = 0
'
OPEN "com1:9600,n,8,1,CS,DS,rb 2048,tb 2048" FOR
   RANDOM AS #1
'
' =====================================================
' clear the RS232 buffers and reset STATUS reporting to
' insure no messages pending
=======================================================
'
lc.status = INP(&H3FB)             'get line control register
'                                   (COM1)
lc.status = lc.status OR 64        'set the break bit to on
OUT &H3FB, lc.status               'send the modified
'                                   register contents back
'
SLEEP 1                            'wait 1 second
'
lc.status = INP(&H3FB)             'get line control register
'                                   (COM1)
lc.status = lc.status AND (NOT 64) 'reset the break bit
OUT &H3FB, lc.status               'send the modified register
'                                   contents back
'
========================================================
' now continue with verification of the comm port set up
========================================================
'
PRINT #1, "RS232 VERBOSE:ON"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 ECHO:OFF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 EOL:CRLF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 FLOW:NONE"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
'
' set up the 2714 or 2715 so REQUEST message will NOT display on
analyzer
' and so the 2714 or 2715 will return header information
'
PRINT #1, "RQS Off;HDR ON"
GOSUB READ.FOR.VERBOSE
'EXIT SUB'
'****************************************************************
'  This routine will send the command string and read the       *
'  response                                                     *
'****************************************************************
'
SEND.MESSAGE.TO.DEVICE:
'PRINT #1, wrt$
'
```

```
'=======================================
' begin read loop for response from 2714 or 2715
'=======================================
'
GOSUB READ.FOR.VERBOSE
'
EXIT SUB
'
'****************************************************************
'  This routine will read the 2714 or 2715 response to get a
binary     *
'  waveform.                                                    *
'****************************************************************
'GET.BINARY.WAVEFORM:
'
GOSUB READ.FOR.VERBOSE
'
IF LEN(rd$) >= 14 THEN
      rd$ = LEFT$(rd$, LEN(rd$) - 4)   'strip off checksum,
'                                       semi-colon and cr/lf
      rd$ = RIGHT$(rd$, LEN(rd$) - 9)  'strip off header
'                                       and byte count
'
      FOR x% = 0 TO 511
          wfm%(x%) = ASC(MID$(rd$, x% + 1, 1))
'                                       convert each binary value
'                                       to ascii
      NEXT
END IF
'
EXIT SUB
'
'****************************************************************
'  This routine will display the error message if RS232         *
'  communication cannot be established                          *
'****************************************************************
'
ERROR.DISPLAY:
'
'                                       save cursor coordinates
'
hold.x% = POS(0)
hold.y% = CSRLIN
'=================
'save screen image
'=================
    REDIM before$(7, 41), colr%(7, 41)
```

```
FOR p% = 0 TO 7
   FOR q% = 0 TO 41
    colr%(p%, q%) = SCREEN(p% + 18, q% + 20, 1)
    before$(p%, q%) = CHR$(SCREEN(p% + 18, q% + 20))
   NEXT q%
NEXT p%
'=============================================
' next print the window and the error message
'=============================================
COLOR 14, 6
'
FOR z% = 18 TO 25
  LOCATE z%, 20, 0
  PRINT STRING$(42, CHR$(32));
NEXT
'
LOCATE 18, 30, 0
PRINT "COM(1) PORT PROBLEM";
LOCATE 19, 25
PRINT "Verify RS232 Port Configuration";
LOCATE 20, 31
PRINT "(UTIL MENU 4/0/2)";
LOCATE 21, 22
PRINT "STATUS..... ONLINE  BAUD RATE....  9600";
LOCATE 22, 22
PRINT "DATA BITS.. 8       PARITY........ NONE";
LOCATE 23, 22
PRINT "EOL........ CRLF    FLOW CONTROL.. NONE";
LOCATE 24, 22
PRINT "ECHO....... OFF     VERBOSE....... ON";
'
LOCATE 25, 27, 0
PRINT "Press {any key} to continue";
DO WHILE INKEY$ = ""
LOOP
'=========================
' redisplay original screen
'=========================
hold.min% = 0
max% = 41
min% = 0
FOR index% = 1 TO 8
    FOR p% = 0 TO 7
        FOR q% = min% TO max% STEP 7
            frgrnd% = colr%(p%, q%) MOD 16
            bckgrnd% = (((colr%(p%, q%) – frgrnd%) / 16)
                       MOD 128)
```

```
                COLOR frgrnd%, bckgrnd%
                LOCATE p% + 18, q% + 20
                PRINT before$(p%, q%);
            NEXT q%
        NEXT p%
        IF min% > hold.min% THEN
            min% = min% + 1
            hold.min% = min%
        ELSE
            min% = min% + 1
            hold.min% = 0
        END IF
    NEXT index%
'
LOCATE hold.y%, hold.x%
COLOR 14, 9
rd$ = ""
'
RETURN
'
READ.FOR.VERBOSE:
'
end.of.read$ = "N"
error.query$ = "N"                 'two flags used in reading
'                                   response from 2714 or 2715
COM(1) ON                          'enable event trapping for
'                                   communication activity on
'                                   com port 1
i = TIMER
DO WHILE end.of.read$ = "N"        'Give 60 seconds for response
'                                   from event query.
  IF TIMER > i + 60 THEN           'If no response, assume
'                                   communication
      COM(1) OFF                   'not established, so set
'                                   flag to indicate
      end.of.read$ = "Y"           'this.
      error.query$ = "Y"
   END IF
LOOP
'
'======================================================
' begin checking for error flags after return from read
'======================================================
'
IF error.query$ = "Y" THEN         'no communication
'                                   established
        GOSUB ERROR.DISPLAY        'display error
        read.error% = 1            'set error flag
        CLOSE #1                   'close comm port
ELSEIF buffer$ <> "Y" THEN
        GOSUB ERROR.DISPLAY        'so do the same as above
        read.error% = 1            'set error flag
        CLOSE #1
END IF
'
RETURN
'
END SUB
'

'****************************************************************
' Procedure to SEND COMMAND to the 2714 or 2715 after input from
      *
'   ENTERCOMMAND procedure.                                     *
'                                                               *
' Command passed in the global string "WRT$"                   *
'   NOTE:                                                       *
'      only SINGLE commands can be sent by the user.            *
'****************************************************************
SUB SENDCOMMAND
'
wrt$ = UCASE$(wrt$)
'
FOR IX% = 20 TO 24                 'clear out the response window
   LOCATE IX%, 8
   PRINT STRING$(66, " ");
NEXT
'
LOCATE 20, 8                       'reposition cursor
'
POSIT1 = INSTR(wrt$, "?")          'all queries end with "?"
'
IF POSIT1 = 0 THEN                 'if no question mark
   GOSUB PROCESS.ONE.COMMAND       'then one command entered
ELSE
   GOSUB PROCESS.ONE.QUERY
END IF
'
EXIT SUB
'
'****************************************************************
' sending a waveform to the 2714 or 2715 is a completely differ-
ent      *
' process than all other commands so do it separately          *
```

```
'****************************************************************
'
PROCESS.ONE.COMMAND:
'
IF LEFT$(wrt$, 3) = "CUR" OR LEFT$(wrt$, 5) = "CURVE" THEN
                                   'if a curve command
    GOSUB SEND.WAVE                'then go send it
ELSE
    func% = 3                      'else send all other
    CALL RS232.CALLS               'commands the same
    rd$ = LEFT$(rd$, LEN(rd$) - 2)
    PRINT rd$;
END IF
'
RETURN
'
'****************************************************************
' Since receiving a response from a curve query is different,   *
' the program distinguishes this query and processes it         *
' separately.                                                   *
'****************************************************************
'
PROCESS.ONE.QUERY:
'
IF LEFT$(wrt$, 4) = "CUR?" OR LEFT$(wrt$, 6) = "CURVE?" OR
LEFT$(wrt$, 5) = "CURV?" THEN
    GOSUB ACQUIRE.WAVE
ELSE                               'if length of response
    func% = 3                      'is too large to fit in
    CALL RS232.CALLS               'the window defined for the
    IF LEN(rd$) > 66 THEN          'response, then use this
'                                   routine to display it in
       DO WHILE LEN(rd$) >= 3      'groups small enough to fit.
           GOSUB FRAGMENT.RESPONSE
       LOOP
    ELSE
       PRINT rd$;
    END IF
END IF
'
RETURN
'
'
ACQUIRE.WAVE:
'==================================
' insure a binary waveform transfer
'==================================
hold.wrt$ = wrt$
'
'                                   waveform preamble for binary
'                                   data transfer
'
wrt$ = "WFM ENC:B;"
func% = 3
CALL RS232.CALLS
'
wrt$ = hold.wrt$
'
'                                   validate that curve query is
'                                   properly formatted
'
IF INSTR(wrt$, "CURVE") OR INSTR(wrt$, "CURV") THEN
   IF LEN(wrt$) < 8 THEN
      posit = INSTR(wrt$, "?")
      tem.wrt$ = LEFT$(wrt$, posit)
      wrt$ = tem.wrt$ + " D"
   END IF
ELSEIF LEN(wrt$) < 6 THEN
   posit = INSTR(wrt$, "?")
   tem.wrt$ = LEFT$(wrt$, posit)
   wrt$ = tem.wrt$ + " D"
END IF
'=====================
' send the curve query
'=====================
wrt$ = "HDR ON;" + wrt$            'insure that hdr is on
PRINT #1, wrt$
'=============================
' read and format the response
'=============================
func% = 5
CALL RS232.CALLS
'
'==============
' go display it
'==============
'
GOSUB DISPLAY.WAVE
'
RETURN
'
'****************************************************************
' Here is an example of a hex waveform being sent to the 2714 or
2715.  *
```

```
' This send function assumes that an acquire waveform function  *
' has been performed previously. This is another arbitrary      *
' decision for demonstration purposes.                          *
' Any combination of characters can be constructed to create    *
' a waveform to be sent to the 2714 or 2715.
      *
' Also the waveform is always being sent to storage location C. *
' This is strictly an arbitrary target location.                *
' A,B or C can be specified to receive the waveform.            *
'****************************************************************
'
SEND.WAVE:
'
'===============================
' will always send to waveform C
'===============================
'
'                                   first make sure save C is on
'
wrt$ = "SAV C:ON;"
func% = 3
CALL RS232.CALLS
'
'                                   next set the preamble to
'                                   point to the target
'                                   waveform (C)
'
wrt$ = "WFM WFI:C;"
func% = 3
CALL RS232.CALLS
'
'                                   next construct the waveform
'                                   to be sent
'
wrt$ = "CURVE #H0"                 'curve command with argument
'                                   indicating hex transfer
wrt$ = wrt$ + HEX$(513)            'byte count
FOR x% = 0 TO 511                  'actual waveform data
   TEMP$ = STR$(wfm%(x%))          'conversion routine from
'                                   decimal to hex
      IF VAL(TEMP$) < 16 THEN
         wrt$ = wrt$ + "0"
      END IF
      wrt$ = wrt$ + HEX$(wfm%(x%))
NEXT
'
'                                   next calculate checksum
'                                    NOTE:
'                                    see explanation of CURVE
'                                    command for explanation of
'                                    calculating checksum
'
check.sum& = 3
'
FOR x% = 0 TO 511
   check.sum& = check.sum& + wfm%(x%)
NEXT
'
check.sum& = check.sum& MOD 256
check.sum& = (256 - check.sum&) MOD 256
'
IF check.sum& < 16 THEN
   wrt$ = wrt$ + "0"
END IF
'
wrt$ = wrt$ + HEX$(check.sum&)
'
'                                    finally the waveform is ready
'                                    to send
'
func% = 3
CALL RS232.CALLS
'
RETURN
'
'                                    display up to 5 lines of
'                                    response, 66 characters
'                                    per line
'
FRAGMENT.RESPONSE:
'
display.line% = 1
DO WHILE display.line% < 6
   rd1$ = MID$(rd$, 1, 66)
   posit = 1
   DO WHILE posit <> 0
      posit = INSTR(rd1$, CHR$(10))
      IF posit <> 0 THEN
         MID$(rd1$, posit, 1) = " "
      END IF
   LOOP
   posit = INSTR(rd1$, CHR$(13))
   IF posit <> 0 THEN
       rd1$ = LEFT$(rd1$, posit - 1)
   END IF
'
   LOCATE display.line% + 19, 8, 0
   PRINT rd1$;
   rd$ = RIGHT$(rd$, LEN(rd$) - LEN(rd1$))
'
   IF LEN(rd$) >= 3 THEN
         display.line% = display.line% + 1
   ELSE
         display.line% = 6
   END IF
LOOP'
IF LEN(rd$) >= 3 THEN
   GOSUB TEMP.STOP
END IF'
RETURN
'
DISPLAY.WAVE:
'
'======================================
' routine to display the data on screen
'======================================
x = 20: y = 8
FOR IX% = 0 TO 511
  LOCATE x, y
  PRINT wfm%(IX%);
  y = y + 5
  IF y > 70 THEN
     y = 8
     x = x + 1
  END IF
  IF x > 24 THEN
     x = 20
     GOSUB TEMP.STOP
  END IF
NEXT
'
RETURN
'
'===============================================
' routine invoked to control display of response
'===============================================
'TEMP.STOP:
'
COLOR 0, 7
LOCATE 25, 9, 0: PRINT "Press {Enter} to continue the list";
DO WHILE INKEY$ <> CHR$(13)LOOPCOLOR 9, 9
```

```
FOR indx% = 20 TO 25
    LOCATE indx%, 8
    PRINT STRING$(66, CHR$(32));
NEXT
COLOR 14, 9
RETURN
END SUB
```

CATV XMOD Command Demo Program

The program in Example 6–2 executes the `CATV XMOD` command, and displays the test data in a single-column format. This example illustrates how to perform the built-in XMOD test routine from a remote location. Simple modifications to this program will make possible remote execution of the 2714 or 2715's remaining CATV test modes.

The program can be revised for GPIB by modifying the device-dependent initialization, error handling, and I/O functions contained in the main module subroutine `READ.BUFFER`, and in the `RS232.CALLS` module.

Example 6–2 performs the following functions:

1. It opens `COM1` and establishes RS-232 communications with the spectrum analyzer using this configuration:

Baud rate:	9600
Data bits:	8
Stop bits:	1
Parity:	None
Verbose:	ON
Echo mode:	OFF
Terminator:	CR LF
Flow control:	NONE

If the 2714 is not properly configured the program displays an error message. Refer to the `RS232` command description in *Command and Query Definitions* and the RS-232 installation procedure in *Introduction To Programming* for more information on RS-232 configuration.

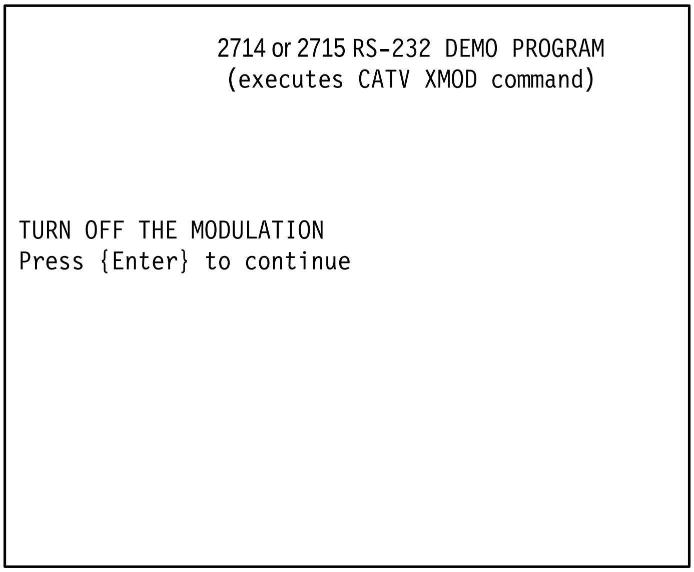
2. The program generates the terminal display shown in Figure 6–2. A screen prompt instructs the operator to turn off the modulation.

3. The operator presses **[Enter]** to begin the CATV XMOD test. The program terminates if the operator presses the **[ESC]** key on the computer keyboard.

4. After the XMOD test completes, a screen prompt on the terminal instructs the operator to turn on the modulation. The terminal display is similar to that shown in Figure 6–2.

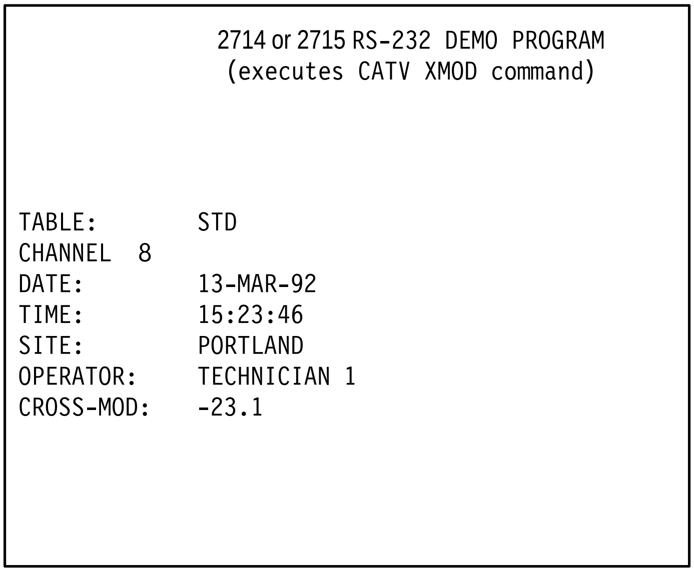
5. Next the program sends the `CATV FULLXMOD` query. The response is arranged in an easy-to-read, single column format (Figure 6–3).

6. A screen prompt instructs the operator to press **[Enter]** on the computer keyboard to run the program again, or **[Esc]** to exit the program.



```
               2714 or 2715 RS-232 DEMO PROGRAM
                (executes CATV XMOD command)



TURN OFF THE MODULATION
Press {Enter} to continue
```

Figure 6–2: Terminal Display Before CATV XMOD Test Begins



```
               2714 or 2715 RS-232 DEMO PROGRAM
                (executes CATV XMOD command)



TABLE:        STD
CHANNEL  8
DATE:         13-MAR-92
TIME:         15:23:46
SITE:         PORTLAND
OPERATOR:     TECHNICIAN 1
CROSS-MOD:    -23.1
```

Figure 6–3: Terminal Display of CATV XMOD Test Results

The program declares the following procedures:

`CROSS.MOD`

This procedure executes the CATV cross-modulation command (`CATV XMOD`), and then sends an `EVEnt?` query. A screen prompt is generated after event 403 is returned. This prompt instructs the operator to press **[Enter]** to resume the test. After the test concludes, the `CATV? FULLXMOD` query is sent, and the query response is displayed in a single column format on the terminal screen.

`RS232.CALLS`

This procedure performs all input/output for RS-232 communications between the controller and the 2714 or 2715. It also checks for communication errors and displays error messages.

***NOTE**. The* `SUB RS232.CALLS` *subroutine in this program is identical to the* `SUB RS232.CALLS` *subroutine used in Example 6–1.*

The 2714 or 2715 must have a spectral display on its screen when this program is executed. If the 2714 or 2715 is displaying a menu when the program is executed, it will not run properly.

Example 6–2 CATV XMOD Command Demo Program

```
'****************************************************************
' This is an example of a program that sends and processes a    *
' command that pauses the operation of the 2714 or 2715. The CATV
XMOD  *
' command is sent which performs the cross-modulation function. *
' The 2714 or 2715 will pause and wait for you to press the
<Enter>     *
' key. When <Enter> is pressed (signifying that the source of   *
' modulation has been removed) the CATV XMOD command continues  *
' execution. Finally, the results of the crossmodulation        *
' measurement are printed on screen.                            *
*****************************************************************
' Communications are established with a baud rate of 9600,      *
' data bits set to 8, parity of NONE, EOL = CRLF, FLOW          *
' CONTROL = NONE, ECHO = OFF, and VERBOSE = ON.                 *
*****************************************************************
'
DECLARE SUB CROSS.MOD ()
DECLARE SUB RS232.CALLS ()
'
'=================================================
' global variables used for communications linkage
'=================================================
'
COMMON SHARED RD$, wrt$, func%, errflg$, end.of.read$, buffer$,
   wfm%()
'
```

```
'==================
' set up ESCape key
'==================
'
KEY(15) OFF
KEY 15, CHR$(&H0) + CHR$(&H1)
ON KEY(15) GOSUB END.PROGRAM
KEY(15) ON
'
'=============================
' set up the communication key
'=============================
'
ON COM(1) GOSUB READ.BUFFER
'
'==============
' select device
'==============
'
func% = 1
CALL RS232.CALLS
'
'========================
' call cross-mod. routine
'========================
CALL CROSS.MOD
'
RERUN.TEST:
LOCATE 20, 10
PRINT "Press {Enter} to rerun program, or {Esc} to end."
    WHILE INKEY$ <> "": WEND
    DO WHILE INKEY$ <> CHR$(13)
    LOOP
    CALL CROSS.MOD
    GOTO RERUN.TEST
'
END.PROGRAM:
COLOR 7, 0
CLS
END
'
'****************************************************************
'                                                               *
'  read routine for RS232 branched from RS232.                  *
'  CALLS subprogram                                             *
'                                                               *
'****************************************************************
```

```
'
'===========
READ.BUFFER:
'===========
'
COM(1) OFF                         'turn automatic branch off
'
RD$ = ""                           'initialize string to save
'                                   response from 2714 or 2715
'
IF INSTR(wrt$, "NORM") THEN        'set time limit for reading
'                                   response from 2714 or 2715
   hold.i = 40                     'normalize query takes
'                                   longer to respond
ELSEIF func% = 5 THEN              'binary waveform transfer
'                                   may take longer
   hold.i = 15
ELSE                                 '10 seconds will be
'                                     enough for others
   hold.i = 10
END IF
'
i = TIMER                            'initialize counter and
DO WHILE TIMER < i + hold.i          'try to read for time
'                                     alloted to make sure
  GOSUB READ.INPUT                   'controller is just not too
  IF end.of.read$ = "Y" THEN         'fast for 2714 or 2715
      i = 0
  END IF
LOOP
'
end.of.read$ = "Y"                   'set flag to avoid possible
'                                     endless loop in RS232 calls
RETURN
'
'==========
READ.INPUT:
'==========
'
buffer$ = "N"                        'initialize flag to indicate
'                                     that read something
'
DO WHILE NOT EOF(1)
     a$ = INPUT$(LOC(1), #1)         'read entire contents
'                                     of buffer
     RD$ = RD$ + a$                  'and accumulate response
     buffer$ = "Y"                   'set buffer flag on
```

```
'                                      so know have read
LOOP                                 'something
'
'*****************************************************************
'                                                                *
'  Following code segment distinguishes between a binary         *
'  waveform transfer and any other response from the 2714 or
2715.      *
'  Binary transfers can contain embedded cr/lf characters so    *
'  must ignore them until get past the binary data.             *
'                                                                *
'*****************************************************************
'
IF func% <> 5 THEN                   'use first cr/lf encounter
'                                     for EOF
   IF INSTR(RD$, CHR$(10)) THEN      'on everything but binary

'                                     files
       IF INSTR(RD$, CHR$(13)) THEN  'insure that read
           end.of.read$ = "Y"        'EOF (cr/lf)
           buffer$ = "Y"             'then set buffer flag ON to
       END IF                        'indicate successful read
   END IF
ELSE
   IF LEN(RD$) > 523 THEN            'insure have read past
'                                     binary data
      IF INSTR(524, RD$, CHR$(13)) THEN
                                     'before checking for cr/lf
          IF INSTR(525, RD$, CHR$(10)) THEN
              end.of.read$ = "Y"
              buffer$ = "Y"          'set buffer flag on
          END IF
      END IF
   END IF
END IF
'
RETURN
'===============================================================
' This submodule executes the CATV cross-modulation command
' (CATV XMOD) and does an EVENT? query. When event 403 is
' received, a message is printed on screen prompting the user
' to press <Enter> to continue. After <Enter> is pressed,
' the CONT command is sent and the CATV XMOD command resumes
' execution. When the command has finished executing, a CATV?
' FULLXMOD query is sent, and the full query response is parsed
' out and formatted on screen.
'===============================================================
'
SUB CROSS.MOD
CLS
COLOR 14
LOCATE 2, 20
PRINT "     2714 or 2715/RS-232 DEMO PROGRAM"
LOCATE 3, 20
PRINT "   (executes CATV XMOD command)"
'

wrt$ = "CATV XMOD;EVE?"                'send CATV XMOD command
func% = 3                              'and query the
CALL RS232.CALLS                       'event
'
IF INSTR(RD$, "EVENT 403") THEN        'when event 403 is received
    LOCATE 10, 10                      'prompt user to
    PRINT "TURN OFF THE MODULATION"; 'turn off modulation
    LOCATE 11, 10
    PRINT "Press {Enter} to continue";
    WHILE INKEY$ <> "": WEND
    DO WHILE INKEY$ <> CHR$(13)
    LOOP
    LOCATE 10, 10                      'Clear
    PRINT STRING$(70, " ")             'the
    LOCATE 11, 10                      'prompt
    PRINT STRING$(70, " ")             'message
    '    LOCATE 10, 10
    PRINT "Running Cross-Modulation Test ..."
    '
    wrt$ = "CONT;CATV? FULLXMOD"       'query the results
    func% = 3                          'from the
    CALL RS232.CALLS                   'cross-modulation
    LOCATE 15, 10
    GOSUB PRINT.RESPONSE
ELSE
    'display error message
END IF
    EXIT SUB
PRINT.RESPONSE:
    LOCATE 10, 10                      'Clear
    PRINT STRING$(70, " ")             'the
    LOCATE 11, 10                      'status
    PRINT STRING$(70, " ")             'message
    '
    LOCATE 9, 1
    START = INSTR(RD$, ":")            'get char position of ";"
    RD$ = MID$(RD$, START + 1)         'remove header from response
    TERM = INSTR(RD$, ",")               'get char pos. of ","
    TABLE$ = MID$(RD$, 2, TERM - 3)      'extract channel table
    IF TABLE$ = "" THEN
        PRINT "     TEST FAILED, CHECK ANALYZER AND RETRY"
        SLEEP 3
        RETURN
    END IF
    PRINT "TABLE:      "; TABLE$
      RD$ = MID$(RD$, TERM + 1)
      TERM = INSTR(RD$, ",")
      CHANNEL$ = MID$(RD$, 1, TERM - 1)
                                        'extract channel #
    PRINT "CHANNEL:    "; CHANNEL$
      RD$ = MID$(RD$, TERM + 1)
      TERM = INSTR(RD$, " ")
      DAT$ = MID$(RD$, 2, TERM - 1)
                                        'extract date
    PRINT "DATE:       "; DAT$
      RD$ = MID$(RD$, TERM + 1)
      TERM = INSTR(RD$, ",")
      TIM$ = MID$(RD$, 1, TERM - 2)
                                        'extract time
    PRINT "TIME:       "; TIM$
      RD$ = MID$(RD$, TERM + 1)
     TERM = INSTR(RD$, ",")
      SITE$ = MID$(RD$, 2, TERM - 3)
                                        'extract site name
    PRINT "SITE:       "; SITE$
      RD$ = MID$(RD$, TERM + 2)
      TERM = INSTR(RD$, ",")
      OPER$ = MID$(RD$, 1, TERM - 2)
                                        'extract operator name
    PRINT "OPERATOR:   "; OPER$
      RD$ = MID$(RD$, TERM + 1)
      TERM = INSTR(RD$, ";")
      XMOD$ = MID$(RD$, 1, TERM - 1)
                                        'extract crossmod value
    PRINT "CROSS-MOD:  "; XMOD$
    '
    RETURN
END SUB
'
'*****************************************************************
'  This procedure performs all RS232 calls. All information      *
'  needed has been defined as global variables:                  *
'          rd$ = returned info from RS232 call                   *
'          wrt$ = info to send to device                         *
'          func% = identifies which RS232 function to perform   *
'          wfm% = array to hold waveform                        *
'****************************************************************
SUB RS232.CALLS
'
IF func% = 1 THEN
          GOTO SELECT.DEVICE
ELSEIF func% = 3 THEN
          GOTO SEND.MESSAGE.TO.DEVICEELSEIF func% = 5 THEN
          GOTO GET.BINARY.WAVEFORM
END IF
'
'****************************************************************
'  This routine opens COM PORT 1 for communication via RS232,   *
'  opened for random to handle both input and output.           *
*'***************************************************************
'  SPEED = 9600 (bits per second-also called BAUD RATE)         *
'  PARITY = NONE (parity bit not used)                          *
'  DATA = 8 (number of data bits per byte)                      *
'  STOP = 1 (number of stop bits)                               *
'  EOL = CRLF (carriage return/line feed as terminator)         *
'  FLOW CONTROL = NONE (no "handshaking")                       *
'  ECHO = OFF (only needed for terminal emulation)              *
'  VERBOSE = ON (guarantees a response for each communication)  *
*'***************************************************************
'  If the 2714 or 2715 has not been configured to agree with the
above  *
'  parameters, the program will display an error message and    *
'  force the user to change the 2714 or 2715. Program will not
      *
'  execute otherwise.                                           *
'****************************************************************
'  CS  =  0  (suppress checking the CLEAR TO SEND line)         *
'  DS  =  0  (suppress checking the DATA SET READY line)        *
'  RB  =  2048  (size in bytes of the receive buffer)           *
'  TB  =  2048  (size in bytes of the transmit buffer)          *
'****************************************************************
'
SELECT.DEVICE:
'
buffer$ = "N"                       'initialize these two
read.error% = 0                     'indicators
'
OPEN "com1:9600,n,8,1,CS,DS,rb 2048,tb 2048" FOR RANDOM AS #1
'
'
==============================================================
' clear the RS232 buffers and reset STATUS reporting to insure
' no messages pending
==============================================================
'
lc.status = INP(&H3FB)              'get line control
'                                    register (COM1)
lc.status = lc.status OR 64         'set the break bit to on
OUT &H3FB, lc.status                'send the modified
'                                    register contents back
'
SLEEP 1                             'wait 1 second
'
lc.status = INP(&H3FB)              'get line control
'                                    register (COM1)
lc.status = lc.status AND (NOT 64)  'reset the break bit
OUT &H3FB, lc.status                'send the modified
'                                    register contents back
'
'
========================================================
' now continue with verification of the comm port set up
========================================================
'
PRINT #1, "RS232 VERBOSE:ON"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "rs232 ECHO:OFF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "RS232 EOL:CRLF"
GOSUB READ.FOR.VERBOSE
'
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
PRINT #1, "rs232 FLOW:NONE"
GOSUB READ.FOR.VERBOSE
'
```

```
IF read.error% = 1 THEN
    GOTO SELECT.DEVICE
END IF
'
'
'                                   set up the 2714 or 2715 so
REQUEST
'                                   message will NOT display on
'                                   analyzer and so the 2714 or
2715 will
'                                   return header information
'
PRINT #1, "RQS Off;HDR ON"
GOSUB READ.FOR.VERBOSE
'
EXIT SUB
'
'*****************************************************************
'  This routine will send the command string and read the        *
'  response                                                      *
'*****************************************************************
'
SEND.MESSAGE.TO.DEVICE:
'
PRINT #1, wrt$
'
'=======================================
' begin read loop for response from 2714 or 2715
'=======================================
'
GOSUB READ.FOR.VERBOSE
'
EXIT SUB
'
'*****************************************************************
'  This routine will read the 2714 or 2715 response to get a
binary     *
'  waveform.                                                     *
'*****************************************************************
'
GET.BINARY.WAVEFORM:
'
GOSUB READ.FOR.VERBOSE
'
IF LEN(RD$) >= 14 THEN
      RD$ = LEFT$(RD$, LEN(RD$) - 4)
                                    'strip off checksum,
'                                    semi-colon and cr/lf
      RD$ = RIGHT$(RD$, LEN(RD$) - 9)
                                    'strip off header and byte
'                                    count
'
      FOR x% = 0 TO 511
          wfm%(x%) = ASC(MID$(RD$, x% + 1, 1))
                                    'convert each binary value to
'                                    ascii
      NEXT
END IF
'
EXIT SUB
'
'****************************************************************
'  This routine will display the error message if RS232         *
'  communication cannot be established.                         *
'****************************************************************
'
ERROR.DISPLAY:
'
'                                    save cursor coordinates
'
hold.x% = POS(0)
hold.y% = CSRLIN
'==================
' save screen image
'==================
    REDIM before$(7, 41), colr%(7, 41)
    FOR p% = 0 TO 7
       FOR q% = 0 TO 41
        colr%(p%, q%) = SCREEN(p% + 18, q% + 20, 1)
        before$(p%, q%) = CHR$(SCREEN(p% + 18, q% + 20))
       NEXT q%
    NEXT p%
    '============================================
    ' next print the window and the error message
    '============================================
    COLOR 14, 6
    '
    FOR z% = 18 TO 25
      LOCATE z%, 20, 0
      PRINT STRING$(42, CHR$(32));
    NEXT
    '
    LOCATE 18, 30, 0
    PRINT "COM(1) PORT PROBLEM";
```

```
LOCATE 19, 25
PRINT "Verify RS232 Port Configuration";
LOCATE 20, 31
PRINT "(UTIL MENU 4/0/2)";
LOCATE 21, 22
PRINT "STATUS..... ONLINE  BAUD RATE..... 9600";
LOCATE 22, 22
PRINT "DATA BITS.. 8       PARITY........ NONE";
LOCATE 23, 22
PRINT "EOL........ CRLF    FLOW CONTROL.. NONE";
LOCATE 24, 22
PRINT "ECHO....... OFF     VERBOSE....... ON";
'
'                                first clear input key buffer
'
WHILE INKEY$ <> "": WEND
LOCATE 25, 21, 0
'
' now wait for user input
'
PRINT "Press {enter} to continue or {esc} to end";
DO WHILE INKEY$ = ""
LOOP
'===========================
' redisplay original screen
'===========================
hold.min% = 0
max% = 41
min% = 0    FOR index% = 1 TO 8
    FOR p% = 0 TO 7
        FOR q% = min% TO max% STEP 7
            frgrnd% = colr%(p%, q%) MOD 16
            bckgrnd% = (((colr%(p%, q%) - frgrnd%) / 16)
                       MOD 128)
            COLOR frgrnd%, bckgrnd%
            LOCATE p% + 18, q% + 20
            PRINT before$(p%, q%);
        NEXT q%
    NEXT p%
    IF min% > hold.min% THEN
        min% = min% + 1
        hold.min% = min%
    ELSE
        min% = min% + 1
        hold.min% = 0
    END IF
NEXT index%
```

```
'
LOCATE hold.y%, hold.x%
COLOR 14, 9
RD$ = ""
'
RETURN'
READ.FOR.VERBOSE:
'
end.of.read$ = "N"                  '
error.query$ = "N"                  'two flags used in reading
'                                    response from 2714 or 2715
COM(1) ON                           'enable event trapping for
'                                    communication
'                                    activity on com port 1
i = TIMER
DO WHILE end.of.read$ = "N"         'give 60 seconds for
'                                    response from event
   IF TIMER > i + 60 THEN           'query. If no response,
'                                    assume communication
      COM(1) OFF                    'not established so
'                                    set flag to indicate
      end.of.read$ = "Y"            'this.
      error.query$ = "Y"
   END IF
LOOP
'
'=======================================================
' begin checking for error flags after return from read
'=======================================================
'
IF error.query$ = "Y" THEN          'no communication
'                                    established
        GOSUB ERROR.DISPLAY         'display error
        read.error% = 1             'set error flag
        CLOSE #1                    'close comm port
ELSEIF buffer$ <> "Y" THEN
        GOSUB ERROR.DISPLAY         'so do the same as above
        read.error% = 1             'set error flag
        CLOSE #1
END IF
'
RETURN
'
END SUB
```

Remote Menu Control

The 2714 or 2715 includes a set of commands and queries for designing menus on the spectrum analyzer screen, and interacting with these menus by pressing keys on the spectrum analyzer Keypad. This feature is designed primarily for making automated measurements with the Tektronix 2402A TekMate™, an external, portable, PC-based controller without a display or keyboard. These features can also be used with a standard PC controller to define a menu remotely. This allows the 2714 or 2715 user to interact with menu functions locally, at the Keypad, and see results on the spectrum analyzer's display.

Remotely-defined menus are controlled by these four commands:

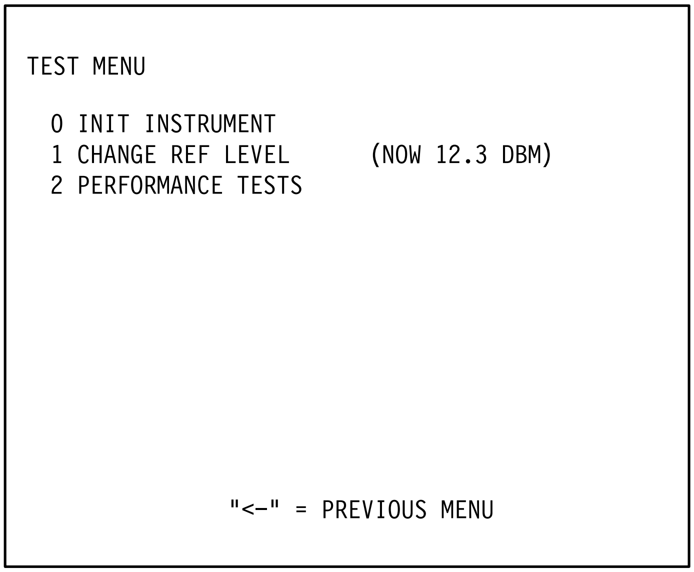
- `DEFMenu`: defines menu lines to print on the display
- `CLRMenu`: clears the current menu from the display
- `KEY?`: returns the identity of the last key pressed
- `CLRKey`: clears pending key presses

The syntax and use of these commands, including side effects, are described in *Command and Query Definitions*. Figure 6–4 shows a test menu that could be defined by a user. This example shows how different types of actions can be implemented by selecting the menu commands from a user-defined menu.

The following sequence of instrument-specific commands are used to create the menu of Figure 6–4:

```
CLRMENU

DEFMENU L1:"TEST MENU"

DEFMENU L3:" 0 INIT INSTRUMENT"

DEFMENU L4:" 1 CHANGE REF LEVEL     (NOW 12.3DBM)"

DEFMENU L5:" 2 PERFORMANCE TESTS"

DEFMENU L16:"     ""<-"" = PREVIOUS MENU"
```

The initial `CLRMenu` command clears every line in the menu so only the lines that contain information must be defined. This command also clears the last key pressed so that subsequent `KEY?` queries return `NULL` until the user presses a key. Any key presses prior to the menu definition are discarded.



```
TEST MENU

  0 INIT INSTRUMENT
  1 CHANGE REF LEVEL      (NOW 12.3 DBM)
  2 PERFORMANCE TESTS











                "<–" = PREVIOUS MENU
```

Figure 6–4: A Remote Menu

Notice the use of paired quotes (“”) to represent a single quote mark on the menu display. This menu could also have been defined using a single DEFMenu command with arguments separated by commas in place of the four individual DEFMenu commands. Refer to *Command and Query Definition* for the syntax of this command.

Execute and Exit

Item 0 on the remote menu of Figure 6–4 is the simplest type of entry. The desired action is instrument initialization and an exit from the menu when the operator presses the **[0]** key on the spectrum analyzer’s front panel Keypad. This is accomplished by the following algorithm executing on either the Tektronix 2402A TekMate or a controller:

- Do KEY? queries until the result is not NULL.
- If MO is returned (key **[0]** has been pressed), execute an INIT followed by a CLRMenu command.

This is a very simple example. The 2402A TekMate (or your PC controller) is capable of performing a complex test sequence in response to a single keystroke.

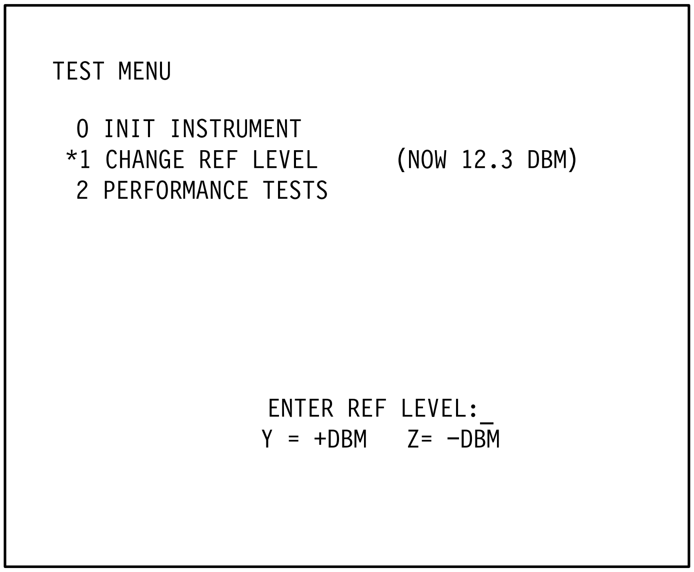
Numeric Entry

Item 1 on the remote menu of Figure 6–5 is a numeric entry item. This is a more complex entry because it requires additional interaction with the operator. When the 2714 or 2715's front panel **[1]** key is pressed, a numeric entry line is displayed at the bottom of the screen (Figure 6–5). Terminator keys for the entry are also defined; **[Y]** = +DBM and **[Z]** = –DBM. The user enters a number and terminator from the spectrum analyzer Keypad in response to this prompt.

The terminator key signals the 2402A TekMate (or PC) when the numeric entry is finished, and also specifies the units. The 2402A TekMate then programs the value into the instrument, or it may use the entry as a parameter for itself, as outlined in the following algorithm:

- Do KEY? queries until the result is not NULL.
- If M1 is returned (key **[1]** has been pressed), execute the following commands:

```
DEFMENU L4:" *1 CHANGE REF LEVEL     (NOW 12.3DBM)"

DEFMENU L15:"      ENTER REF LEVEL:_"

DEFMENU L16:"      Y = +DBM  Z = –DBM"
```



```
TEST MENU

  0 INIT INSTRUMENT
 *1 CHANGE REF LEVEL     (NOW 12.3 DBM)
  2 PERFORMANCE TESTS






                ENTER REF LEVEL:_
               Y = +DBM   Z= –DBM
```

Figure 6–5: Prompting for a Numeric Entry

Next the programmer should use an algorithm similar to the following one to accept the data entry:

- Do KEY? queries until the result is not NULL.
- Use appropriate error checking to ensure only numeric, decimal point, and terminator key entries are accepted.
- Begin building an ASCII string by appending successive key presses to a string variable, say VAL$.
- Change line 15 of the menu with this command:

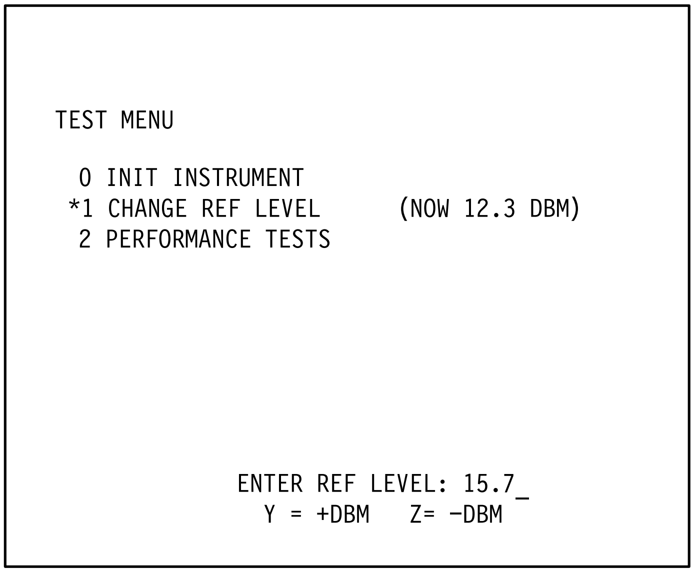
  ```
  DEFMENU L15:"    ENTER REF LEVEL: " +  VAL$ + "_"
  ```

- Loop until the whole number has been entered and a terminator has been detected.

For example, if the keys **[1] [5] [.]** and **[7]** are pressed, the menu display will appear as shown in Figure 6–6. When a terminator key is pressed, the program must respond by setting the reference level using the REF command:

```
REF 15.7 DBM

CLRMENU
```

The CLRMENU command automatically exits the menu. If desired, you may remain in the remote menu by issuing the necessary DEFMENU commands to rewrite the screen.



```
TEST MENU

  0 INIT INSTRUMENT
 *1 CHANGE REF LEVEL      (NOW 12.3 DBM)
  2 PERFORMANCE TESTS







              ENTER REF LEVEL: 15.7_
                Y = +DBM   Z= -DBM
```

Figure 6–6: Specifying a Numeric Value

Defining a Submenu

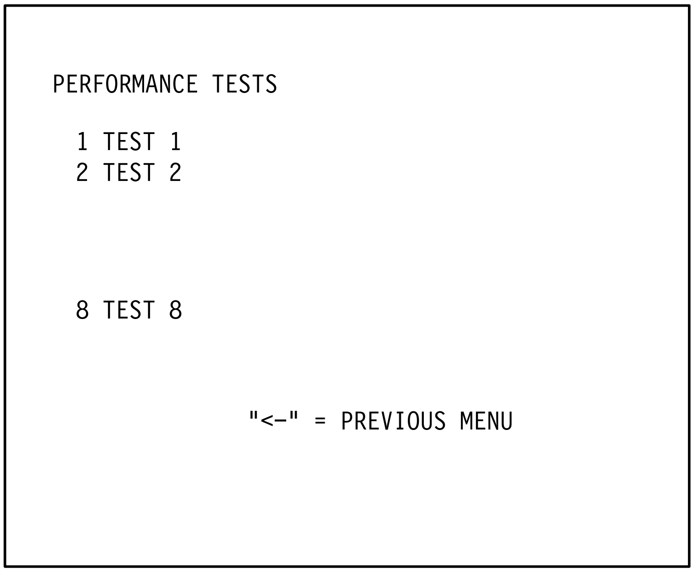
Item 2 on the remote menu of Figure 6–6 shows how submenus might be created. Use the 2402A TekMate to execute the following algorithm for an example of building a submenu:

- Do KEY? queries until the result is not NULL.
- If the key is M2, send the following commands:

```
CLRMENU

DEFMENU L1:"PERFORMANCE TESTS"

DEFMENU L4:" 1 TEST 1"

DEFMENU L5:" 2 TEST 2"

DEFMENU L9:" 8 TEST 8"

DEFMENU L16:"     ""<-"" = PREVIOUS MENU"
```

The resulting menu is shown in Figure 6–7. Note that the 2714 or 2715 does not understand a nested menu structure. The nesting structure is maintained by the application program running on the controller.

The application program also implements the Backspace **[BKSP]** key to return to a higher menu level, as with the standard 2714 or 2715 menus.



```
PERFORMANCE TESTS

  1 TEST 1
  2 TEST 2



  8 TEST 8


               "<-" = PREVIOUS MENU
```

Figure 6–7: A Remote Submenu

Introduction to GPIB Programming

The routines in the *GPIB Sample Subroutines* on page 6–48 can also work with earlier versions of QuickBASIC. However, you must use the appropriate GPIB system software and make the changes to function calls as outlined in the READ-QB.DOC document file from National Instruments. The GPIB system software is that supplied by National Instruments along with their boards.

The devices considered are the National Instruments PCII, PCIIA, and PCII/IIA GPIB boards, supplied by Tektronix or directly available from National Instruments dealers. Consult *Introduction To Programming* on page 1–1 and the material supplied with your board for detailed instructions.

Beginning Your GPIB Program

Before you begin programming, several steps must be performed to ensure that QuickBASIC has the necessary GPIB information. See *Preparing the Software* on page 1–10 for the procedure.

A BASIC or QuickBASIC program usually begins with a list of declarations. This establishes variable types and the names of global variables. Several variable names are used by the National Instruments software. To prevent any confusion, National Instruments supplies a program file that declares these variables for you. Place this line at the beginning of your programs to properly declare the reserved National Instruments variables:

```
REM $INCLUDE 'QBDECL4.BAS'
```

QuickBASIC supports the sub-program module concept, so traditional COMMON statements are replaced with COMMON SHARED statements. This means that the variable names so declared may be shared by all modules. Further, each sub-program module must be declared. The following lines declare the global variables used by our subroutines and demonstration program:

```
COMMON SHARED BD%,BDNAME$,RD$,

WRT$ , EVENT.CODE$

COMMON SHARED NUMBYT%
```

Table 6–1 lists the global variables declared by QBDCL4.BAS and the COMMON SHARED statements, and their purpose.

Table 6–1: Variable Names

Name	Description	Remarks
BD%	Integer value associated with a particular device name by IB-FIND(BDNAME$,BD%).	Other names can be used such as MYDEV%, FRST.DEV%, A1%, etc.
BDNAME$	String variable set to a device name such as TEK_SA. The name is the one used when you set up the device with IBCONF.	Any string variable name can be used such as MYNAME$, DEVNAM$, DN$, etc.
EVENT.CODE$	A string variable argument of IBRD containing the 2714 or 2715 event code.	Any string variable name can be used: (MYNAME$, DEVNAM$, DN$, etc.)
IBCNT%	Integer variable updated by GPIB system software after a read, write or command: the number of bytes transferred.	This name is reserved.
IBERR%	Integer variable returned by GPIB system software when error bit of status word is set. Range: 0 to 7, or 10 to 16.	See your GPIB manual for meanings of IBERR% values. The name is reserved.
IBSTA%	Integer variable updated by all GPIB system software functions.	Refer to *Status Byte* on page *5–11.*
NUMBYT%	Integer argument of DEBLK which indicates the number of bytes converted.	Any integer variable name can be used (MYNUM%, INTEG%, etc.).
RD$	String variable used with the IBDR function to contain the data returned by the 2714 or 2715.	Other names can be used such as RET.DAT$, MYDAT$, S1$, etc.
WRT$	A string variable used with the IBWRT function to contain the data to be sent to the 2714 or 2715.	Other names can be used such as WRT.DAT$, MESSAGE$, B4$, etc.

You may desire to dimension arrays or variables at the beginning of your program, even though they are to be used later. For instance, CUR%() is the integer array used for curve data in our demonstration program. RD$ is a string variable that we use as the destination for a variety of data transfers.

If memory size is not a limiting factor in your controller, you may wish to set up the maximum size of RD$ at the head of the program with a SPACE$ command as in this example:

```
DIM SHARED CUR%(512)

RD$ = SPACE$(<max anticipated size>)
```

The number of bytes in the response is less than 5000 for all queries except PLOT? and FILE?.

The PLOT? query may require as much as 37,000 bytes for HPGL plots and 61,100 for Epson plots. The number of points in a PLOT? response depends on the number of waveforms present, and the acquisition mode.

It is possible for a User-Defined Program (UDP) to occupy all available memory. As a result, UDPxx files larger than 100,000 bytes are possible. However, 5000 bytes are adequate for most UDPs.

Error Trapping

There are three types of errors that can occur in your program: DOS, GPIB System, and instrument-related. Different techniques are used to deal with each type. Errors are typically trapped so corrective action can be taken, or the program is caused to end in a non-destructive manner (gracefully).

DOS Errors

DOS errors may occur on instruments equipped with either the GPIB or RS-232 interface. DOS errors typically happen when trying to access a device that is missing or not ready. They are traditionally handled with the BASIC ON ERROR statement. Following the declarations at the head of the program, you must add this line:

```
ON ERROR GOTO ERR.TRAP
```

Next you should construct a subroutine to deal with the problem. This routine may be used to end the program in a non-destructive manner (gracefully):

```
ERR.TRAP:

    PRINT "DOS ERROR HAS OCCURRED."

    PRINT "CHECK YOUR SYSTEM SETUP."

    END

RESUME NEXT
```

With this routine the program stops if a DOS error is encountered. You simply check your system and restart. Consult a BASIC programming manual for selective error trapping techniques and other approaches.

GPIB System Errors

Instruments with a GPIB interface may encounter GPIB system errors. (Refer to programming examples in *RS-232 Program Examples* on page 6–2 for RS-232 error handling techniques.) These errors are detected by examining IBSTA% following each GPIB function call.

A GPIB error has occurred whenever the value of IBSTA% is equal to or greater than 32768. To determine what type of error has occurred, you can check the value of IBERR%. Once you have verified the type of error, you may take corrective action, proceed without doing anything (if the program continues to run and yields valid results), or end the program gracefully. This subroutine can be called following each call to a GPIB function to determine the type of error and to end the program gracefully:

```
GPIB.ERR

    IF IBSTA%<32768 THEN RETURN

    PRINT "GPIB ERROR HAS OCCURRED"

    PRINT "GPIB ERROR CODE IS ";IBERR%

    END

RETURN
```

Instrument-Related Errors

A third type of error involves those directly related to the 2714 or 2715 or its interface with the bus. Any time the 2714 or 2715 detects an abnormal condition, it issues a service request (SRQ) if RQS is set to ON. The SRQ causes the GPIB board to generate a light pen interrupt.

Abnormal events can be detected and decoded automatically by branching to an interrupt handler whenever the interrupt occurs. To do this, add these lines to your program:

```
ON PEN GOSUB ABNORM.EVE

PEN ON

    :

ABNORM.EVE:

    CALL SERIAL.POLL

    CALL EVENT.FIND

RETURN
```

SERIAL.POLL and EVENT.FIND are sample subroutines discussed in *Status Reporting*.

GPIB System Software

All of the programming examples in this manual make use of the subroutines supplied by National Instruments as part of the GPIB or Tektronix GURU II software. If you are using a board from another manufacturer, equivalent software should have accompanied your board. This software supplies the interface between the programming language and the installed GPIB board.

Table 6–2 lists the National Instruments system subroutines used in this manual. Consult your GPIB documentation for detailed information about these and many other subroutines that are commonly available.

Table 6–2: GPIB System Software Callable Subroutines

Subroutine	Description
DEBLK(CUR%(),CUR%(), CNT%,8,NUMBYT%)	Convert first CNT% elements of array CUR%() from binary block to 2-byte integer format in same array and return the number of bytes converted.
IBFIND(BD%,BDNAME$)	Open device indicated by BDNAME$ and return unit descriptor BD%.
IBRD(BD%,RD$)	Read data from BD% to string RD$.
IBRDF(BD%,FILENAME$)	Read data from device BD% and store to disk in file named <FILENAME$>.
IBRDI(BD%,CUR%(),CNT%)	Read CNT% data bytes from device BD% into integer array CUR%().
IBRSP(BD%,SPR%)	Perform serial poll of device BD%.
IBSRE(BD%,V%)	Enable/disable remote mode in the device indicated by BD%. V%=0 disables.
IBWRT(BD%,WRT$)	Send contents of string variable WRT$ to device indicated by BD%.
IBWRTF(BD%,FILENAME$)	Send disk file named <FILENAME$> to device indicated by BD%.

GPIB Sample Subroutines

This subsection contains a collection of subroutines that illustrate simple approaches to transferring various types of data between the system controller and the spectrum analyzer. You may wish to incorporate them into your own software or modify them so they are more appropriate to your needs.

The header statements in Example 6–3 can be placed at the beginning of a program, and be used as a basis for the subroutines in this section. If you modify the subroutines, or add new ones, the statements may no longer be adequate.

***NOTE**. The header statements in Example 6–3 and the sample subroutines in this section do not make provisions for error trapping and event reporting.*

See *Status Reporting* and the demonstration program at the end of this section for error and event reporting routines.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–3 GPIB Program Header Statements

```
REM $INCLUDE: 'QBDECL4.BAS'

COMMON SHARED BD%,BDNAME$,RD$,WRT$

RD$=SPACE$(5000)

BDNAME$ = "TEK_SA"

CALL IBFIND (BDNAME$, BD%)
```

Curve Transfers

Curves (waveforms) transferred from the spectrum analyzer to the controller are important for data analysis, archiving, and reporting purposes. Curves transferred to the spectrum analyzer can be used for comparison or to establish references. The `CURve` command transfers a block of data from the controller to the spectrum analyzer and the `CURve?` query returns a block of data from the spectrum analyzer to the controller.

The data block represents the 512 points in a 2714 or 2715 waveform (see *Command and Query Definitions* for `CURve?` response formats). Before curve data is transferred you must specify which digital display register (A, B, C, or D) the curve is coming from or going to, and the type of data encoding to be used. The encoding (ASCII-encoded decimal, ASCII-encoded hexadecimal, or binary) is determined by the waveform preamble (see the `WFMpre` command).

The destination of the transmitted data or origin of the returned data is also determined by the preamble unless the `A`, `B`, `C`, or `D` argument is used with the `CURve?` query. In these cases, the origin is specified by the argument. For example, `CURve? C` returns data from the C register.

Example 6–4 shows two QuickBASIC subroutines that can be used with the National Instruments board and software to send and return ASCII-encoded curve data. The returned data is displayed on the controller screen, and will resemble the ASCII example in the `CURve` command as described in *Command and Query Definitions*. The `WFMpre` command determines the encoding and source or destination registers in both subroutines. An alternate approach to transferring curve data (as packed integers) is illustrated in the GPIB demonstration program at the end of this section.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–4 Subroutines to Return or Transmit Curve Data

```
   GET.CURVE:
'ESTABLISH SOURCE REGISTER AND ENCODING, ENSURE
'HEADER IS TURNED ON
   WRT$ = "WFMPRE WFID:D,ENCDG:ASCII;HDR ON"
      CALL IBWRT(BD%,WRT$)
'RESERVE SPACE FOR THE DATA
   RD$ = SPACE$(2056)
'RETURN THE CURVE DATA
   WRT$ = "CURVE?"
      CALL IBWRT(BD%,WRT$)
      CALL IBRD(BD%,RD$)
'TRIM AND DISPLAY THE DATA ON SCREEN
   PRINT MID$(RD$,1,IBCNT%)
RETURN
   :
   PUT.CURVE:
'ESTABLISH ENCODING AND DESTINATION REGISTER
'SAVE THE REGISTER
WRT$ ="WFMPRE WFID:A,ENCDG:ASCII;SAVE A:ON"
      CALL IBWRT(BD%,WRT$)
'TRIM CURVE DATA AND SEND IT
'CURVE COMMAND HEADER IS INCLUDED IN RD$
   WRT$ = MID$(RD$,1,IBCNT%)
      CALL IBWRT(BD%,WRT$)
RETURN
```

The curves transferred to the spectrum analyzer in these examples are the previously returned waveforms, but you can also send artificially generated curves. Such curves can be generated in ASCII format using a spreadsheet. Curves should always be transferred to a saved register to ensure they are not immediately overwritten by the next spectrum analyzer sweep.

Whenever a curve is generated, the `WRT$ = MID...` line in the `PUT.CURVE` subroutine must be replaced by a statement such as this one:

```
   WRT$ = "CURve "+IND$+BC$+DATA$+CK$
```

where:

IND$ = Null for ASCII, #H for hex, % for binary

BC$ = Null for ASCII, 0201 for hex, CHR$(2)+CHR$(1) for binary (Byte Count)

DATA$ = 512 data points, appropriately encoded, representing the curve

CK$ = Null for ASCII , HEX$(chksum) for hex, “0”+HEX$(chksum) if chksum < 16, CHR$(chksum) for binary (Checksum)

Transferring Files

The FILE command/query transfers named data files between the spectrum analyzer and controller. FILE and FILE? enable data from one instrument to be returned for disk storage and subsequent transmission to another instrument, or perhaps, to the same instrument in the event of NVRAM failure. They are not intended or well-suited for viewing curves or editing settings.

Permissible file names are established by the spectrum analyzer, and are listed under the FILE command discussion in *Command and Query Definitions*. The files are created within the spectrum analyzer’s memory only as required. That is, a BSET03 file exists only if the B-register settings have been previously saved in the third storage location. The currently created files can be viewed by pressing the key sequence **[UTIL MENU] [4] [6]**.

NOTE*. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–5 Subroutines to Return or Transmit Data Files

```
   GET.FILE:
'ENTER THE 2714 or 2715 FILENAME FOR THE FILE TO UPLOAD
'SEE TABLE 4-3 FOR A LIST OF THE NAMES
   FILE2714 or 2715$ = <2714 or 2715 FILENAME>
'TURN ON HEADER, REQUEST FILE
   WRT$ = "HDR ON;FILE? "+FILE2714 or 2715$
      CALL IBWRT(BD%,WRT$)
'READ FILE INTO STRING VARIABLE RD$
      CALL IBRD(BD%,RD$)
'TRIM STRING VARIABLE TO NUMBER
'OF CHARACTERS TRANSFERRED
   FILEDAT$ = MID$(RD$,1,IBCNT%)
   RETURN
      :
   PUT.FILE:
'SEND THE FILE IN MEMORY BACK TO THE 2714 or 2715
```

```
WRT$ = FILEDAT$
   CALL IBWRT(BD%,WRT$)
RETURN
```

Example 6–5 lists QuickBASIC subroutines that can be used with the National Instruments GPIB board and software to transmit or return data files. You must supply the name of the file to be returned or transmitted.

The response header is turned on when the file is returned to the controller. This is how the returned data string (FILEDAT$) will appear:

```
FILEDAT$ = FILE "<filename>",<data block>
```

This is exactly the form required for the FILE command, so upon transmission you need only send FILEDAT$. Alternatively, you can set HDR OFF in GET.FILE and then set WRT$ = "FILE "+FILEDAT$ in PUT.FILE.

Similar files may be sent to different locations, but you cannot transfer one type of file to another type. That is, you can return BSET04 and send it to CSET05, but you cannot return ACF2 and rename it UDP07.

The file name is embedded in the response, so this subroutine transmits the file to the same location that it came from. This limitation can be circumvented by inserting this line at the beginning of PUT.FILE:

```
MID$(FILEDAT$,1) = "FILE <new filename>"
```

These subroutines do not store or retrieve the files to and from disk. This can be done with the usual BASIC OPEN, PRINT # and INPUT # statements. Alternatively, you can use the National Instruments IBWRTF() and IBRDF() calls to transfer data directly between the spectrum analyzer and disk. Example 6–5 is an example of this approach.

Plotting Spectrum Analyzer Screen Data

The PLOT? query enables the transfer of data representing an image of the 2714 or 2715 display screen from the spectrum analyzer to a plotter or printer. It performs a function similar to the front panel **[PLOT]** key. The printer or plotter must speak the HPGL language or be compatible with Epson FX codes, and the appropriate printer type must be specified either locally or using the PTYpe command.

It is possible for the spectrum analyzer to send data directly to a plotter. To do this, the GPIB ATN line must be held high while the spectrum analyzer is addressed as a talker and the plotter as a listener, and then ATN is set low. EOS ON and WAIt for an end-of-sweep SRQ must also be set. This approach requires no computer memory.

An alternative approach to plotting the spectrum analyzer screen data enables you to create independent input and output subroutines for use with specific devices, and to share those devices with other instruments on the GPIB. This approach involves returning the screen plot data to the controller, then sending it to the designated output device. The program does not proceed until the plot data is received, so it is not necessary to `WAIt` for an end-of-sweep SRQ. Further, this approach enables you to do the printing or plotting at a more convenient time or use an entirely different controller and output device.

Figure 6–8 and Figure 6–9 show how you might return and print or plot instrument data. Each “Do it” in the diagrams can represent a separate subroutine for a specific device.

Example 6–6 shows a subroutine for obtaining screen plot data from the 2714 or 2715 using the `PLOT?` query. It does not store the data on disk (you can add that capability if desired), but holds the data in memory as the string variable `PLOT.DAT$`.

The length of the string required depends on the display acquisition mode, the plotter type, and the number of registers displayed. 12 kbyte of memory is enough for a single sweep in PEAK acquisition mode if an HPGL plotter is used. If an Epson FX printer is used, as many as 61.1 Kbyte may be required for four sweeps in MAX/MIN mode. Substitute `61100` for `12000` and change `HPGL4` to `EPSON` if using an Epson FX printer.

Specify `HPGL2` if you have a 2-pen plotter. Data may be sent to a parallel Epson-type printer on the controller’s parallel port, but to send it over the GPIB, the printer must be equipped with a GPIB board (an unlikely but possible configuration).

The `SEND.PLOT` subroutine transmits `PLOT.DAT$` to an HPGL 4-pen plotter matching the type specified in `GET.PLOT`. Note that the controller timeout is disabled, and execution following the plot is restarted by pressing any key.

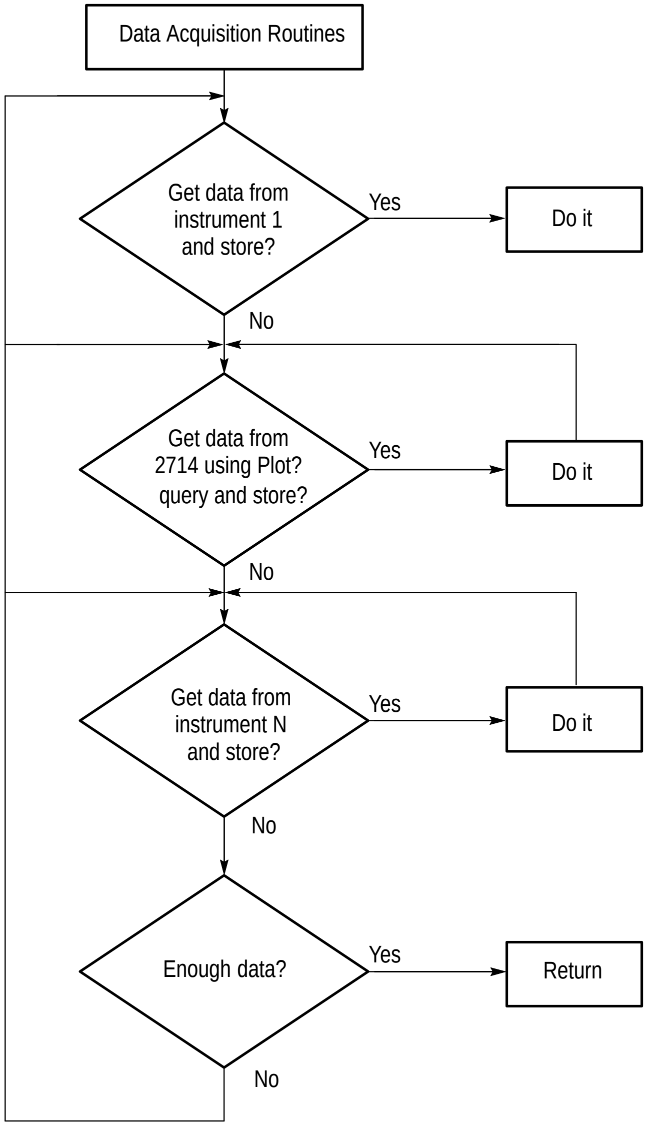

Data Acquisition Routines
Get data from instrument 1 and store?
Yes
Do it
No
Get data from 2714 using Plot? query and store?
Yes
Do it
No
Get data from instrument N and store?
Yes
Do it
No
Enough data?
Yes
Return
No

Figure 6–8: Possible Data Acquisition Scheme

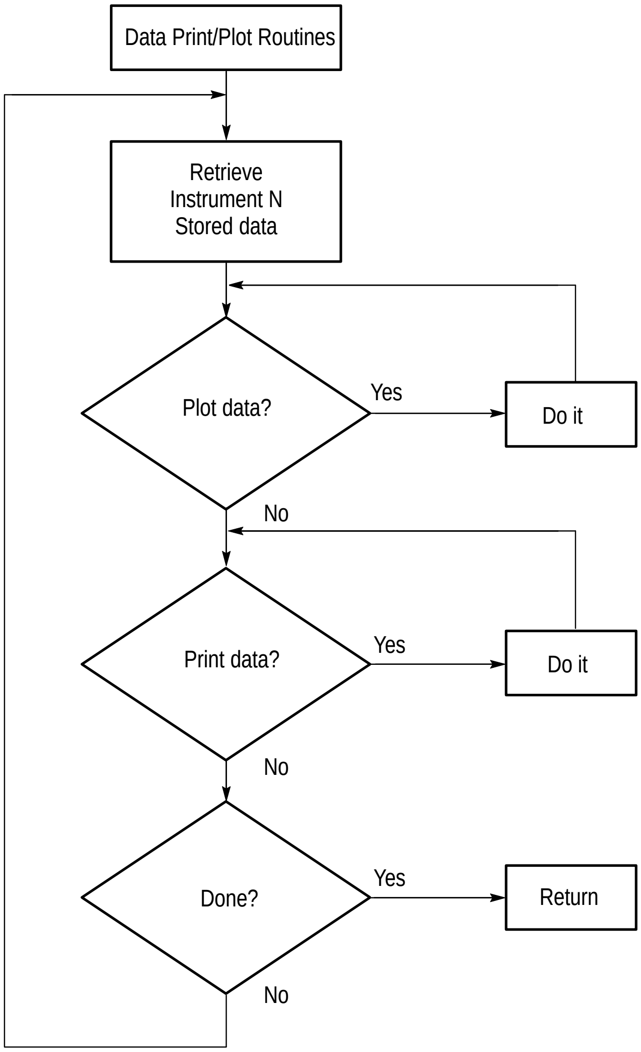


Figure 6–9: Possible Data Print/Plot Scheme

Returning the On-screen Readouts

The 2714 or 2715's on-screen readouts provide a summary of the important operational instrument parameters. The `PRDouts?` query enables returns most of these readouts to the computer. This query does not return the 2714 or 2715's general purpose message line, the GPIB status line, or the user-defined DISPLAY MESSAGE line.

The `PRDouts?` query does return the `PRDOUTS` header (if `HDR` is on) and up to 14 arguments, depending on the spectrum analyzer's mode of operation and its current status. The arguments are listed in *Command and Query Definitions* under `PRDouts?` on page 4–59. The response is ASCII-encoded and can be read and displayed on the controller screen using the subroutine in Example 6–7.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–6 Subroutines to Return or Send Screen Plot Data

```
 GET.PLOT:
'RESERVE SPACE FOR SCREEN DATA
      PLOT.DAT$ = SPACE$(12000)
'SET PLOTTER TYPE AND REQUEST SCREEN DATA
      WRT$ = "PTYPE HPGL4;PLOT?"
         CALL IBWRT (BD%,WRT$)
'GET SCREEN DATA
         CALL IBRD (BD%,PLOT.DAT$)
'TRIM DATA TO NUMBER OF BYTES TRANSFERRED
      PLOT.DAT$=MID$(PLOT.DAT$,1,IBCNT%)
   RETURN
      :
   SEND.PLOT:
'DISABLES TIME OUT TO GIVE PLOTTER TIME TO FINISH
         CALL IBTMO(BD%,0)
      CLS
'SEND SCREEN DATA TO PLOTTER
         PLOTTER$ = "HC100"
         CALL IBFIND(PL%,PLOTTER$)
         CALL IBWRT(PL%,PLOT.DAT$)
'PRESS A KEY AFTER PLOTTER FINISHES
      PRINT "PRESS ANY KEY TO CONTINUE"
      DO WHILE INKEY$ = ""
         LOOP
'REESTABLISHES 30-SECOND TIME OUT
         CALL IBTMO (BD%,14)
   RETURN
```

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–7 Subroutine for Returning On-Screen Readouts

```
   READOUTS:
'RESERVE SPACE FOR THE READOUTS
      READOUT.DAT$ = SPACE$(448)
'REQUEST THE READOUTS
      WRT$ = "PRDOUTS?"
         CALL IBWRT (BD%,WRT$)
'RETURN THE READOUTS
         CALL IBRD (BD%,READOUT.DAT$)
'TRIM DATA TO NUMBER OF BYTES RETURNED
      READOUT.DAT$ = MID$(READOUT.DAT$,1,IBCNT%)
'DISPLAY THE READOUTS
      PRINT READOUT.DAT$
   RETURN
```

Saving and Restoring Equipment Settings

The SET? query can be used to return the current spectrum analyzer control settings. The settings are returned in a message format containing the command headers and arguments necessary to place the 2714 or 2715 in its current configuration.

Settings can be saved in a controller disk file for later transfer to the same 2714 or 2715, or to another 2714 or 2715, when restoring the same operating environment. Because the SET header is always suppressed in the response, the response can be transmitted as received.

The SET? query is generally used in preference to settings file transfers for several reasons:

- The same command and format can be used with a variety of Tektronix instruments for the same purpose.
- The 2714 or 2715 implements the retransmitted settings as soon as received rather than requiring a separate RECall command.
- The returned settings are in ASCII format and can be easily read if necessary.

See the SET? query in *Command and Query Definitions* for details.

Figure 6–8 shows QuickBASIC subroutines for storing and reestablishing the settings group. The routine is suitable for use with the National Instruments GPIB board and software.

***NOTE**. You must substitute the disk filename in which settings are to be stored in place of* “filename.”

NOTE. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.

Example 6–8 Subroutines to Save and Restore Settings Groups

```
 GET.SET:
'REQUEST THE SETTINGS
      WRT$ = "SET?"
         CALL IBWRT (BD%,WRT$)
'READ THE SETTINGS FROM THE 2714 or 2715
         CALL IBRD (BD%,RD$)
'TRIM THE SETTINGS TO THE NUMBER OF BYTES RETURNED
      SETTINGS$ = MID$(RD$,1,IBCNT%)
'DISPLAY THE SETTINGS FOR VERIFICATION PURPOSES
      PRINT SETTINGS$
'SAVE SETTINGS ON DISK --SUBSTITUTE DISKFILE NAME
'THAT YOU WANT TO STORE SETTINGS UNDER FOR FILENAME
            OPEN "O",#1,"FILENAME"
      PRINT #1, SETTINGS$
      CLOSE #1
   RETURN
         :
   PUT.SET:
'OPEN THE DISK FILE AND READ IN THE STORED SETTINGS
      OPEN "I",#1,FILENAME$
      INPUT #1,SETTINGS$
      CLOSE #1
'DISPLAY THE SETTINGS FOR VERIFICATION PURPOSES
      PRINT SETTINGS$
'SEND THE SETTINGS TO THE 2714 or 2715
      WRT$ = SETTINGS$
         CALL IBWRT (BD%,WRT$)
   RETURN
```

Waiting for Results

A number of 2714 or 2715 functions require a wait period between the time they are requested or begin execution, and the time at which the results of the operation are available. These functions are listed below:

- Delta marker readouts
- Counter readouts
- Signal searches
- User-Defined Programs

- Marker readouts
- Ensemble averages
- Plots
- Normalizations

In Example 6–9 we have used the M,C MINUS SAVE A mode as an example of how to program for these events. This is only one example of many different possibilities when the `WAIt` command can be used. The `WAIt` command is typically used together with the single sweep mode to maintain synchronization between the controller and 2714 or 2715 during program execution.

***NOTE**. The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–9 Subroutine to Demonstrate the WAIt Command

```
'==============================================================
' program segment to demonstrate the use of the WAIT command
'==============================================================
'
'========================
' set up for auto polling
'========================
'
PEN OFF
ON PEN GOSUB serial.poll
PEN ON
'
'==============================================================
'use only waveform A and place instrument in single sweep mode
'==============================================================
'
wrt$ = "VIEW A:ON;VIEW B:OFF;VIEW C:OFF;VIEW D:OFF;SIGSWP"
CALL IBWRT(bd%, wrt$)
'
'==============================================================
' turn on end of sweep indicator, trigger a single sweep, and
' WAIT until end of sweep
'==============================================================
'
wrt$ = "EOS ON;SIGSWP;WAIT"
CALL IBWRT(bd%, wrt$)
'
```

```
'==============================================================
' halt program execution until EOS is indicated this guarantees
' that what is saved in A register is correct
'==============================================================
'
END.OF.SWEEP$ = "N"
DO WHILE END.OF.SWEEP$ = "N"
LOOP
'
'==============================================================
' DO NOT turn on Save A until at least one sweep has been made
'==============================================================
'
wrt$ = "SAVE A:ON"
CALL IBWRT(bd%, wrt$)
'
'==============================================================
' now use the waveform saved in A to subtract from what is
' displayed in C
'==============================================================
'
wrt$ = "VIEW A:ON;VIEW B:OFF;VIEW C:ON;VIEW D:OFF;VIEW MINU-
SA:ON;"
CALL IBWRT(bd%, wrt$)
'
wrt$ = "SIGSWP;WAIT"
CALL IBWRT(bd%, wrt$)
'
'================================================
' wait here until end of sweep before continuing
'================================================
'
END.OF.SWEEP$ = "N"
DO WHILE END.OF.SWEEP$ = "N"
LOOP
'
'==============================================================
' now reset the analyzer and view the C register which contains
' the difference between what is saved in A and what is
' actively displayed
'==============================================================
'
wrt$ = "EOS OFF;VIEW A:OFF;VIEW B:OFF;VIEW C:ON;VIEW D:OFF;VIEW
MINUSA:ON;"
CALL IBWRT(bd%, wrt$)
'
'=======================
' terminate auto polling
'======================
'
PEN OFF
'
'===============
' exit module
'===============
'
END SUB
```

Sample GPIB Controller

The following `COMM2714 or 2715` program is a simple utility for communicating with the 2714 or 2715 spectrum analyzer over the GPIB. It contains some of the subroutines (or elements of them) discussed earlier in this section, in addition to some new material. This simple program shows how to command and interrogate the 2714 or 2715 in a manner that enables several useful operations to be performed.

NOTE. *The 2714 or 2715 must have a spectral display on its screen when any RS-232 or GPIB program is executed. If the 2714 or 2715 is displaying a menu when a program is executed, the program will not run properly.*

Example 6–10 Sample GPIB Controller Program

```
'                COMM2714 or 2715

'
' program to communicate with a Tektronix 2714 or 2715 Spectrum
Analyzer
' via the GPIB
'
' declare GPIB system software reserved variables
    REM $INCLUDE: 'qbdecl4.bas'
'
' declare common global variables
    COMMON SHARED bd%,BDNAME$,RD$,wrt$
    COMMON SHARED event.code$,NUMBYT%
'
' dimension max size of returned data string
    RD$ = SPACE$(5000)
'
' dimension an integer array for packed integer CURve? response
    DIM SHARED CUR%(512)
```

```
'
' obtain bus device unit descriptor (BD%).
' BDNAME$ must match name
' established for the 2714 or 2715 with the IBCONF program
    BDNAME$ = "TEK_SA"
        CALL IBFIND(BDNAME$, bd%)
'
' establish link to abnormal event handler; enable interrupt line
    ON PEN GOSUB ABNORM.EVE
    PEN ON
'
' enables SRQ generation in case of abnormal event
        CALL ibwrt(bd%, "RQS ON")
' trap DOS errors
    ON ERROR GOTO ERR.TRAP
'
' generate the menu
MENU:
    CLS
    PRINT "F1    SEND COMMAND OR QUERY"
    PRINT
    PRINT "F2    SAVE CURRENT SETTINGS TO FILE"
    PRINT
    PRINT "F3    RESTORE SETTINGS"
    PRINT
    PRINT "F4    SAVE AN INSTRUMENT FILE"
    PRINT
    PRINT "F5    RESTORE AN INSTRUMENT FILE"
    PRINT
    PRINT "F6    ACQUIRE CURVE DATA"
    PRINT
    PRINT "F10   EXIT"
    PRINT
    PRINT "PRESS F1-F6 OR F10 TO MAKE SELECTION"
    PRINT
'
' chk keyboard for keypress, decode,
' and branch to correct subroutine
KYBD.CHK:
    IN$ = INKEY$
    IF IN$ = "" GOTO KYBD.CHK
    IF LEN(IN$) = 1 THEN BEEP: GOTO KYBD.CHK
    IN.KEY = ASC(MID$(IN$, LEN(IN$), 1))
' decoding complete
        SELECT CASE IN.KEY          'branch to subroutine
            CASE 59                 'F1 pressed
                GOSUB SEND.RCV
                GOTO MENU
            CASE 60                 'F2 pressed
                GOSUB SAVE.SET
                GOTO MENU
            CASE 61                 'F3 pressed
                GOSUB RES.SET
                GOTO MENU
            CASE 62                 'F4 pressed
                GOSUB SAVE.FILE
                GOTO MENU
            CASE 63                 'F5 pressed
                GOSUB RES.FILE
                GOTO MENU
            CASE 64                 'F6 pressed
                GOSUB INT.CUR
                GOTO MENU
            CASE 68                 'F10 pressed
                SYSTEM              'returns to dos
        END SELECT
    GOTO MENU                     'regenerates the menu
'
' subroutine to send a command or query
' and receive the response
SEND.RCV:
    CLS
    PRINT : PRINT "ENTER MESSAGE TO SEND"
    PRINT
    INPUT wrt$
        CALL ibwrt(bd%, wrt$)
        GOSUB GPIB.ERR
    hold.time = TIMER               'slight delay; srq check
    DO WHILE TIMER < hold.time + 1
        LOOP
    QUES = INSTR(1, wrt$, "?")
' if ques=0,there is no response;
    IF QUES = 0 THEN GOTO SEND.RCV
'
' if message contains "?" get response and print it
    CALL IBRD(bd%, RD$)
    GOSUB GPIB.ERR
    PRINT : PRINT "THE RESPONSE IS:"
    PRINT : PRINT MID$(RD$, 1, IBCNT%)
    PRINT : PRINT
    INPUT "SEND MORE (ENTER Y OR N)?"; Y$
    IF Y$ = "Y" THEN GOTO SEND.RCV
RETURN
'
```

```
' subroutine to fetch current instrument settings
' from 2714 or 2715 and save to disk
SAVE.SET:
    CLS : PRINT
    PRINT "ENTER  NAME FOR SETTINGS FILE"
    PRINT "USE PATH IF NOT IN CURRENT DIRECTORY"
    INPUT FILENAME$
    WRT$ = "SET?"
        CALL IBWRT(BD%, WRT$)       'request settings
        GOSUB GPIB.ERR
        CALL IBRD(bd%, RD$)         'read settings
        GOSUB GPIB.ERR
        SETTINGS$ = MID$(RD$, 1, IBCNT%)
'
' eliminate extra characters and print
    PRINT : PRINT SETTINGS$
    PRINT : PRINT
    INPUT "OK TO STORE (ENTER Y OR N)? "; Y$
    IF Y$ = "N" THEN RETURN         'store if everything
    OPEN "O", #1, FILENAME$, IBCNT% 'looks OK
    PRINT #1, SETTINGS$
    CLOSE #1
RETURN
' subroutine to restore a group of instrument
' settings from disk to the 2714 or 2715
RES.SET:
    CLS : PRINT
    PRINT "ENTER NAME OF SETTINGS FILE"
    PRINT
    INPUT FILENAME$
    OPEN "I", #1, FILENAME$         'read settings file
    INPUT #1, SETTINGS$
    CLOSE #1
    PRINT : PRINT SETTINGS$
    INPUT "OK TO RESTORE (ENTER Y OR N)? "; Y$
    IF Y$ = "N" THEN RETURN         'if displayed settings
    WRT$ = SETTINGS$                'OK, then restore
        CALL IBWRT(BD%, WRT$)       'to 2714 or 2715
        GOSUB GPIB.ERR
RETURN
'
' subroutine fetches file from 2714 or 2715, stores on disk
SAVE.FILE:
    CLS : PRINT
    PRINT "ENTER NAME OF 2714 or 2715 FILE TO STORE"
    INPUT FILE2714 or 2715$
' see FILE command for 2714 or 2715 file names
```

```
    PRINT
    PRINT "ENTER NAME OF DISK FILE for STORing"
    INPUT FILENAME$
    FILE2714 or 2715$ = UCASE$(FILE2714 or 2715$)
    WRT$="HDR ON;FILE? "+CHR$(34)+FILE2714 or 2715$+CHR$(34)
        CALL IBWRT(BD%, WRT$)
'
' request file transfer
        GOSUB GPIB.ERR
        CALL IBRDF(bd%, FILENAME$)  'read and store
        GOSUB GPIB.ERR              '2714 or 2715 file to disk
RETURN                              'as FILENAME$
' subroutine restores 2714 or 2715 file from disk to the 2714 or
2715
RES.FILE:
    CLS : PRINT
    PRINT "ENTER DISK FILE TO RESTORE TO 2714 or 2715"
    INPUT FILENAME$                 'note: the file named
        CALL IBWRTF(bd%, FILENAME$) 'FILENAME$
        GOSUB GPIB.ERR              'contains the name of
RETURN                              'the 2714 or 2715 file to be
restored
'
' subroutine to fetch curve data in packed binary
' form and convert it to 2-byte integer format
INT.CUR:
    PRINT
    PRINT "GET CURVE FROM WHICH REGISTER?"
    INPUT "     (ENTER A, B, C, OR D)   "; REG$
    PRINT
' ensure response header is on
        CALL IBWRT(BD%, "HDR ON")
        GOSUB GPIB.ERR
' tell 2714 or 2715 which register and encoding to use
    WRT$ = "WFMPRE WFID:" + REG$ + ",ENCDG:BINBLK"
        CALL IBWRT(BD%, WRT$)
        GOSUB GPIB.ERR
        CALL IBWRT(BD%, "CUR?")
        CALL IBRDI(BD%, CUR%(), 9)  'fetch curve
        GOSUB GPIB.ERR              'data in packed binary;
        CALL IBrdi(Bd%, Cur%(), 512)
' write over header characters (1st 9) with data
        CALL DEBLK(CUR%(), CUR%(), 512, 8, NUMBYT%)
        PRINT "# OF BYTES CONVERTED = "; NUMBYT%
RETURN
' DEBLK unpacks binary data and restores as 2-byte integers in
' same array. NUMBYT$ always equals 512 subroutine to find and
' display event code following an SRQ created by an abnormal
' event
ABNORM.EVE:
    CLS
    PRINT "AN ABNORMAL EVENT HAS OCCURRED."
    PRINT
        GOSUB SERIAL.POLL           'call subroutines to poll
        GOSUB EVENT.FIND            '2714 or 2715 and find event
code
    PRINT
    PRINT "R TO RESTART; ANY OTHER KEY TO END"
    INPUT KEY$                      'pressing R returns to
    IF KEY$ <> "R" THEN END         'menu; variables
        GOTO MENU                   'are not erased
RETURN
'
' subroutine to serially poll the 2714 or 2715, read the status
byte,
' and print it used as part of abnormal event handler, but
' can be used any time to obtain status byte; see your GPIB
' documentation for more information
SERIAL.POLL:
'
' read and print status byte; reset SRQ
        CALL IBRSP(BD%, SPR%)
    PRINT "STATUS BYTE = "; SPR%
RETURN
'
' Subroutine to find event code using the EVE? query;
' result valid only after serial poll if RQS is ON
EVENT.FIND:
    PRINT "EVENT CODE(S) IS:";
    WRT$ = "HDR OFF;EVE?"           'turn off header and
    event.code$ = SPACE$(5)         'request event code
        CALL IBWRT(BD%, WRT$)       'send command
        CALL IBRD(BD%, event.code$) 'read code
    PRINT event.code$               'print code
        GOSUB GPIB.ERR
RETURN
' subroutine to print the GPIB error code if a GPIB error occurs,
' and end the program gracefully
GPIB.ERR:
    IF IBSTA% < 32768 THEN RETURN   'no GPIB error
    CLS : PRINT                     'if GPIB status word<32768
    PRINT "GPIB ERROR HAS OCCURRED."
    PRINT : PRINT "GPIB ERROR CODE IS "; IBERR%
    PRINT : PRINT
    PRINT "CHECK YOUR SYSTEM AND RESTART."
    END
RETURN
'
' subroutine to end program gracefully on DOS error
ERR.TRAP:
    CLS : PRINT
    PRINT "DOS ERROR HAS OCCURRED."
    PRINT : PRINT
    PRINT "CHECK YOUR SYSTEM AND RESTART."
    END
RESUME NEXT
```

Appendices

Appendix A: RS-232 Concepts

The first part of this appendix, *Introduction to RS-232 Communications,* introduces RS-232 communications concepts to users with no experience using the RS-232 interface. The second part, *Implementation of the RS-232 Interface*, describes implementation of the RS-232 interface on the 2714 or 2715 Spectrum Analyzer. Other sections in this manual contain the detailed instructions needed to operate over the RS-232. For example, *Introduction to Programming,* contains the information needed to configure the 2714 or 2715 for most applications. *Programming* includes an example of an interactive control program for the 2714 or 2715 Spectrum Analyzer. This program uses the RS-232 interface and a Personal Computer (PC) controller. If additional RS-232 communications information is needed, refer to the documents listed at the end of this appendix.

Introduction To RS-232 Communications

RS-232 communications follow a set of electrical, mechanical, and protocol standards that are similar to the more familiar IEEE 488.1 GPIB standards. Many types of devices are designed to communicate according to specifications contained in standard EIA Std RS-232-C.

The RS-232 interface is NOT a bus (GPIB is a bus). Therefore, only one device can be connected at a time. RS-232 uses an asynchronous serial data flow instead of 8-bit parallel with byte-by-byte handshaking. RS-232 does not support device addresses or serial polling.

The DCE (controller) and the DTE (terminal) must be configured the same way for communications to occur successfully. To meet this requirement, communications parameters for the controller and terminal must be set independently.

RS-232 Signal Components

Figure A–1 is a symbolic representation of how a character of data can be transmitted over an RS-232 interface. It shows the encoding of the 7-bit, serial character as it goes out the interface.

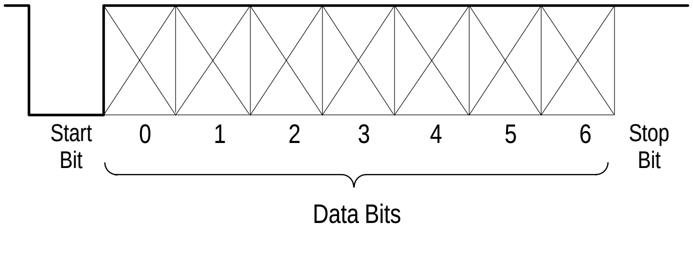


Figure A–1: RS-232 Representation of a Character

The example in Figure A–1 contains two bits called the start bit and stop bit. These bits are added by the interface to the 7-bit character (the *data bits*) as data is sent. The start bit, stop bit, number of data bits, and transmission speed are the most important RS-232 configuration parameters.

The rate of speed that bits are sent through the interface is called the *baud rate*. The RS-232 interface clocks incoming and outgoing bits based upon the specified baud rate. The sending and receiving baud rates must be identical for communications to be established; if they differ the received data will be garbled, causing a framing error or a parity error. These errors are discussed later in this section.

The start bit and stop bit signal the beginning and end of the character, respectively. The receiving interface synchronizes on the start bit shown at the left of Figure A–1. It then uses the baud rate setting and internal clock to time and sample the incoming bits. A framing error occurs if, after detecting a start bit and clocking data bits (and parity bit, if enabled), the receiving interface does not detect the stop bit.

The sending interface can also add one other optional bit before the stop bit, called the parity bit. The parity bit, if used, provides another check for transmission errors. If ODD parity is used, the interface sets this bit to 1 if the 8-bit character consists of an even number of 1 bits. Otherwise, the parity bit would be set to 0. If parity is EVEN, the interface sets the parity bit to 0 if the 8-bit character consists of an even number of 1 bits. Otherwise, the parity bit would be set to 1. The receiving interface uses the parity bit to check incoming data for correct parity.

Implementation of the RS-232 Interface

The 2714 or 2715 follows EIA Standard RS-232-C. This standard establishes electrical levels, connector configuration, and signal protocols for communication between two devices called the DCE (data circuit-terminating equipment) and the DTE (data terminal equipment). The 2714 or 2715 Spectrum Analyzer implements the DTE end of the interface.

The 2714 or 2715’s RS-232 interface is made available through rear panel connector J104. This is a DB9P male connector as shown in Figure A–2. Refer to Table A–1 for a listing of the connector pin definitions.

The RS-232 connector on the 2714 or 2715 (J104) is a PC/AT type RS-232 9-pin male interface connector. The RS-232 connectors used on the PC interface card are not consistent. Three types of connector are used. The three types are 25-pin male, 25-pin female, and 9-pin male.

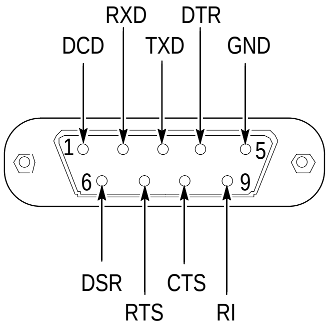


Figure A–2: Rear Panel RS-232 Connector

Table A–1: Back Panel RS-232 Connections

Pin	Use	Description	Direction
1	DCD	Carrier Detect	Input
2	RXD	Received data	Input
3	TXD	Transmitted data	Output
4	DTR	Data terminal ready	Output
5	GND	Signal ground	Common
6	DSR	Data set ready	Input
7	RTS	Request to send	Output
8	CTS	Clear to send	Input
9	(not used)		

Check which type of connector your PC interface card has and refer to the following tables and illustrations to determine which type of cable you need:

- 9-pin Female to 9-pin Female Null Modem Cable

  Table A–2 and Figure A–3 show the wiring configuration for this type of cable. A cable with this configuration is available as an optional accessory. Refer to the *2714* or *2715 Spectrum Analyzer User Manual* for the part number.

NOTE*. Pins 1 and 8 are connected together on each 9-pin female connector. Both pins connect to pin 7 on the opposite 9-pin connector.*

Table A–2: 9-pin Female to 9-pin Female Null-Modem Cable

9-pin Female	to	9-pin Female
Pin (signal)		**Pin (signal)**
1,8 (DCD, CTS)	to	7 (RTS)
2 (RXD)	to	3 (TXD)
3 (TXD)	to	2 (RXD)
4 (DTR)	to	6 (DSR)
5 (GND)	to	5 (GND)
6 (DSR)	to	4 (DTR)
7 (RTS)	to	1,8 (DCD, CTS)
8 (see pin 1)		
9 (not used)		

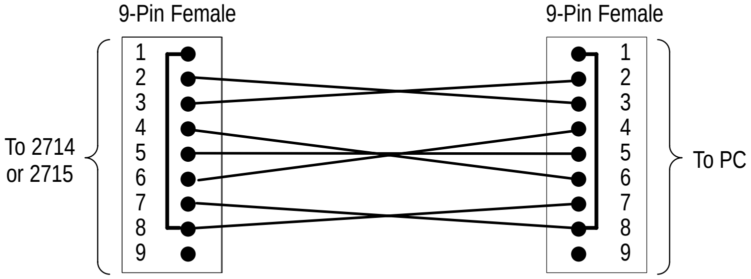


Figure A–3: 9-pin Female to 9-pin Female Null-Modem Cable

- 9-pin Female to 25-pin Female Null-Modem Cable

  Table A–3 and Figure A–4 show the wiring configuration for this type of cable. A cable with this configuration is available as an optional accessory. Refer to the *2714* or *2715 Spectrum Analyzer User Manual* for the part number.

NOTE*. Pins 1 and 8 are connected together on the 9-pin female connector; these connect to pin 4 on the 25-pin female connector. Pins 5 and 8 are connected together on the 25-pin female connector; these connect to pin 7 on the 9-pin female connector.*

Table A–3: 9-pin Female to 25-pin Female Null-Modem Cable

9-pin Female	to	25-pin Female
Pin (signal)		**Pin (signal)**
1,8 (DCD, CTS)	to	4 (RTS)
2 (RXD)	to	2 (TXD)
3 (TXD)	to	3 (RXD)
4 (DTR)	to	6 (DSR)
5 (GND)	to	7 (GND)
6 (DSR)	to	20 (DTR)
7 (RTS)	to	5, 8 (CTS, DCD)
8 (see pin 1)		
9 (not used)		

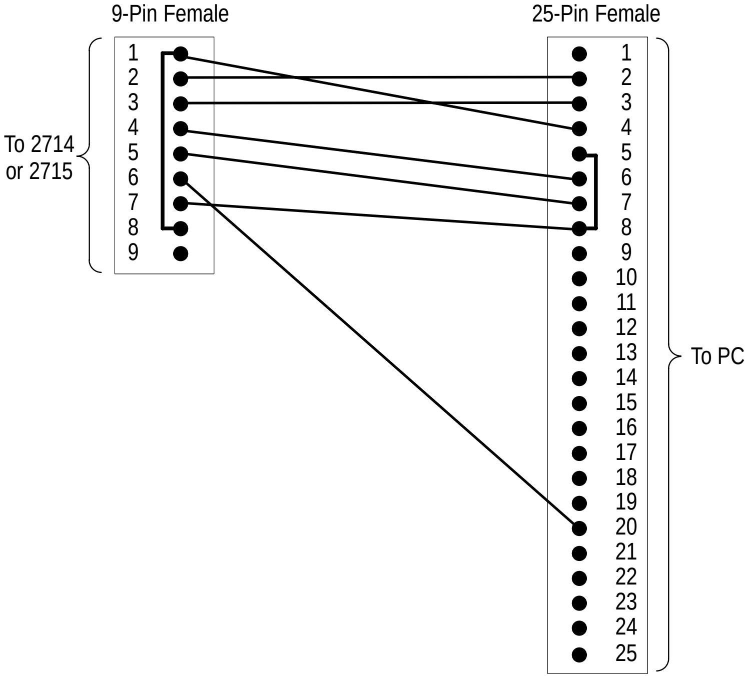


Figure A–4: 9-pin Female to 25-pin Female Null-Modem Cable

- 9-pin Female to 25-pin Male Extension Cable

  Table A–4 and Figure A–5 show the connections for this type of cable. A cable with this configuration is available as an optional accessory. Refer to the *2714* or *2715 Spectrum Analyzer User Manual* for the part number.

Table A–4: 9-pin Female to 25-pin Male Extension Cable

9-pin Female	to	25-pin Male
Pin (signal)		**Pin (signal)**
1 (DCD)	to	8 (DCD)
2 (RXD)	to	3 (RXD)
3 (TXD)	to	2 (TXD)
4 (DTR)	to	20 (DTR)
5 (GND)	to	7 (GND)
6 (DSR)	to	6 (DSR)
7 (RTS)	to	4 (RTS)
8 (CTS)	to	5 (CTS)
9 (not used)		

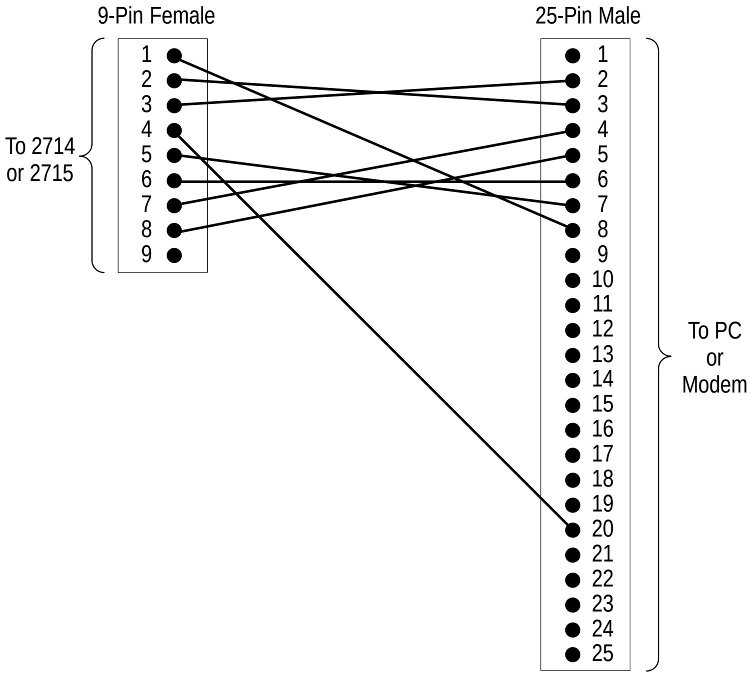


Figure A–5: 9-pin Female to 25-pin Male Extension Cable

Related Documentation

The following documents include the RS-232 standard and *Mastering Serial Communications*, written primarily for programmers.

EIA Standard RS-232-C, August 1969

EIA Standard RS-232-D, January 1987

Mastering Serial Communications by Peter W. Gofton

RS-232 Made Easy by Martin D. Seyer

Appendix B: GPIB System Concepts

The General Purpose Interface Bus (GPIB) is a digital control bus that allows efficient communications between self-contained instruments or devices connected in an instrumentation system. The GPIB is an interface system independent of the stimulus or measurement functions incorporated in any instrument.

Instruments or devices designed to operate on the GPIB digital control bus must be developed according to the specifications contained in IEEE Std 488.1-1978, *IEEE Standard Digital Interface for Programmable Instrumentation*. The IEEE 488.1 digital interface is commonly known as the General Purpose Interface Bus (GPIB). This section discusses the basic concepts of the GPIB. For complete specifications, refer to the IEEE Std 488.1-1978 standard, published by the Institute of Electrical and Electronics Engineers, Inc.

The GPIB has four elements: mechanical, electrical, functional, and operational. Of these four, only the last is device-dependent. Operational elements state the way in which each instrument reacts to a signal on the bus.

Mechanical Elements

The IEEE Std 488.1 defines the GPIB connector and cable assembly as the mechanical elements of the instrumentation system. Standardizing the connector and cable assembly ensures that GPIB-compatible instruments can be physically linked together with complete pin compatibility. The connector has 24 pins: sixteen active signal lines, seven interlaced grounds, and 1 shield connection. Standard connector pin arrangement and nomenclature for the digital control signals are illustrated in Figure B–1.

The cable that attaches to the GPIB connector must be no longer than 20 meters with no more than fifteen peripheral devices (including a GPIB controller) connected at one time. The interconnecting cable assembly, which is offered as an optional accessory to the spectrum analyzer, is provided with a plug and receptacle connector type at each end of the cable to allow either a star or linear bus structure. Contact your local Tektronix Field Office or representative for cable ordering information. Connectors may be rigidly stacked, using standard counter-bored captive screws.

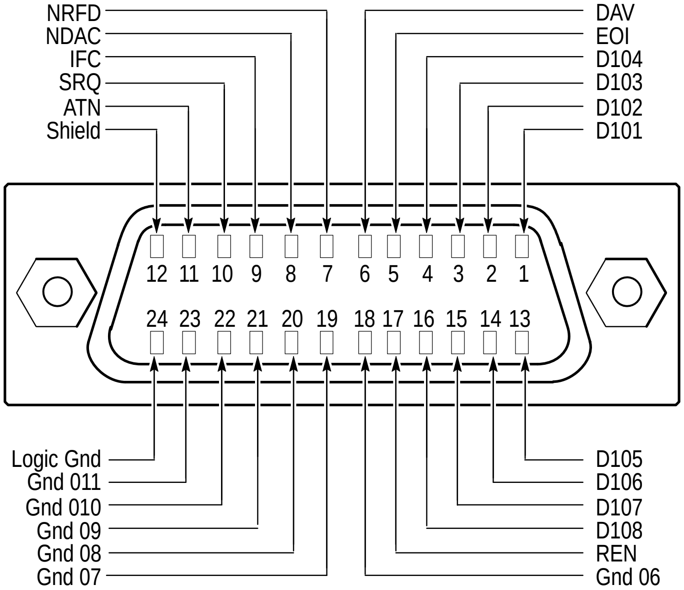


Figure B–1: IEEE Std 488.1 (GPIB) Connector

Electrical Elements

The voltage and current values required at the connector nodes on the bus are based on TTL technology. The power source is not to exceed +5.25 V referenced to logic ground. The standard defines the logic levels as follows:

- Logical 1 is a true state:
  low voltage level ($\leq$+0.8 V), signal line is asserted.
- Logical 0 is a false state:
  high voltage level ($\geq$+2.0 V), signal line not asserted.

Messages can be sent over the GPIB as either active-true or passive-true signals. Passive-true signals occur at a high voltage level and must be carried on a signal line using open-collector devices. Active-true signals occur at a low voltage level.

Functional Elements

The functional elements of the GPIB cover three areas:

- The ten major interface functions of the GPIB are listed in Table B–1. Each function is a system element that provides the basic operational facility by which an instrument can receive, process, and send messages over the GPIB.
- The second functional element is the specific protocol by which the interface functions send and receive their limited set of messages.
- The logical and timing relationships between allowable states for all interface functions is the third area covered.

Table B–1: Major GPIB Interface Functions

Interface Function	Symbol
Source Handshake	SH
Acceptor Handshake	AH
Talker or Extended Talker	T or TE
Listener or Extended Listener	L or LE
Service Request	SR
Remote-Local	RL
Parallel Poll	PP
Device Clear	DC
Device Trigger	DT
Controller	C

Note that while the IEEE Std 488.1 standard defines the ten interface functions, the specific protocol, and timing relationships, not every instrument on the bus will have all ten interface functions incorporated. Only those functions important to a particular instrument's purpose need to be implemented.

A Typical GPIB System

A typical GPIB instrumentation system is shown in Figure B–2, and it includes the nomenclature for the sixteen active signal lines. Only four instruments are shown in this example, but the GPIB can support up to fifteen instruments connected directly to the bus. However, more than fifteen devices can be supported by a single bus if they are interfaced through a primary device. Such a scheme can be used for programmable plug-ins housed in a mainframe where the mainframe is addressed with a primary address code and the plug-ins are addressed with a secondary address code.

To maintain the electrical characteristics of the bus, a device load should be connected for each two meters of cable length. Although instruments are usually spaced no more than two meters apart, they can be separated farther apart if the required number of device loads are lumped at any given point. For proper operation, at least two-thirds of the instruments connected directly to the bus must be in the power-on state.

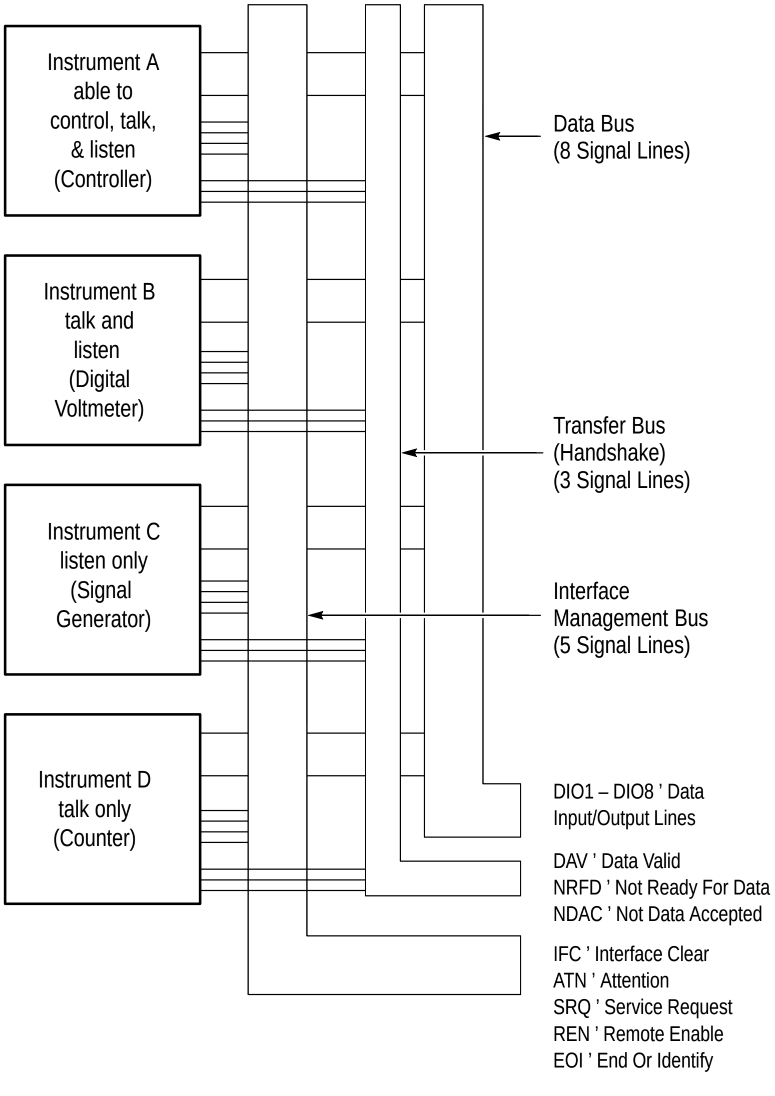


Figure B–2: Typical GPIB System

Talkers, Listeners, and Controllers

A talker is an instrument that can send messages and data over the bus. A listener is an instrument that can accept messages and data from the bus. An instrument can be a talker only, listener only, or be both a talker and a listener. Unless a device is in the talk-only or listen-only mode, it can only communicate with other devices on the bus when it is enabled to do so by the controller in charge of the instrumentation system.

A controller is an instrument that determines, by software routines, which instrument will talk and which instruments will listen during any given time interval. The controller has the ability to assign itself as a talker or a listener whenever the program routine requires it. In addition to designating the current talker and listeners for a particular communication sequence, the controller is assigned the task of sending special codes and commands (called interface control messages) to any or all instruments on the bus. A complete operating system may contain more than one controller. The IEEE standard has provisions for a system controller that operates with another controller in charge of the bus. The controller that is in charge of the bus can take control only when it is directed to do so by the system controller. The system controller may be, but is not necessarily, the controller in charge of the bus.

Interface Control Messages

The two types of interface control messages are multi-line messages sent over the data bus and uni-line messages. A message that shares a group of signal lines with other messages, in some mutually exclusive set, is called a multi-line message. Only one multi-line message **[message byte]** can be sent at one time. A message sent over a single line is called a uni-line message. Two or more of these messages can be sent concurrently.

Only multi-line messages are discussed here; uni-line messages are discussed later in this section. See *GPIB Signal Line Definitions*.

The interface control messages (Table B–4) are sent and received over the data bus only when the ATN (attention) line is asserted (true). Interface message coding can be related to the ISO (International Standards Organization) 7-bit code by relating data bus lines DIO1 through DIO7 to bits B1 through B7, respectively, in the Bits column in Table B–4.

Interface control messages (Table B–2 and Table B–3) include the primary talk and listen addresses for instruments on the bus, addressed commands, universal commands, and secondary addresses for devices interfaced through a primary instrument. Only instruments previously addressed to listen will respond to addressed commands. All instruments, whether they have been addressed or not, will respond to universal commands.

Parallel Poll Enable (PPE) messages are derived from the characters in the first column under Lower Case letters in Table B–4 (decimal coded characters 96 through 111). The standard recommends the use of decimal code 112 (lower case letter p) for the Parallel Poll Disable (PPD) command. All parallel poll-configured instruments respond with status information at the same time when the EOI line is asserted and ATN is true.

Device-Dependent Messages

The IEEE standard does not specify the coding of device-dependent messages that control the internal operating functions of a device. After addressing a talker and the required number of listeners with interface control messages, the controller unasserts the ATN line (false or high) on the bus. When ATN becomes false (high), any commonly understood 8-bit binary code may be used to represent a device-dependent message.

The standard recommends that the alphanumeric codes associated with the numbers, symbols, and uppercase characters (decimal 32 to decimal 94) in the ASCII Code Chart (Table B–4) be used to compose device-dependent messages. One example of a device-dependent message could be the following ASCII character string that controls the signal generator from Figure B–2:

```
MODE V;VOLTS 2.5E–3;FREQ 1.0E3
```

The ASCII character string from this example, sent when the ATN line is unasserted, tells the signal generator to set its front panel controls to the voltage mode (`MODE V;VOLTS`) and produce a 2.5 mV signal (`2.5E-3;`) at a frequency of 1000 Hz (`FREQ 1.0E3`).

When 8-bit binary codes other than the ISO 7-bit are used for device-dependent messages, the most significant bit should be on data line DI08 (for bit 8).

To summarize the difference between interface control messages and device-dependent messages on the data bus, remember that any message sent or received when the ATN line is asserted (low) is an interface control message. Any message (data bytes) sent or received when the ATN line is unasserted (high) is a device-dependent message.

Table B–2: Interface Messages and Functions: Remote Messages Received

Mnemonic	Message	Function
ATN	Attention	AH,C,L,LE,P P,SH,T,TE
DAC	Data Accepted	SH
DAV	Data Valid	AH
DCL*	Device Clear	DC
GET*	Group Execute Trigger	DT
GTL*	Go To Local	RL
IFC	Interface Clear	C,L,LE,T,TE
LLO*	Local Lockout	RL
MSA*	My Secondary Address	LE,TE
MTA*	My Talk Address	T,TE
PPC*	Parallel Poll Configure	PP
PPD*	Parallel Poll Disable	PP
PPE*	Parallel Poll Enable	PP
PPU*	Parallel Poll Unconfigure	PP
REN	Remote Enable	RL
RFD	Ready For Data	SH
SDC*	Selected Device Clear	DC
SPD*	Serial Poll Disable	T,TE
SPE*	Serial Poll Enable	T,TE
SRQ	Service Request	(via C)
TCT*	Take Control	C
UNL*	Unlisten	L,LE

*** Multiline messages.**

Table B–3: Interface Messages and Functions: Remote Messages Sent

Mnemonic	Message	Function
ATN	Attention	C
DAC	Data Accepted	AH
DAV	Data Valid	SH
DCL*	Device Clear	(via C)
GET*	Group Execute Trigger	(via C)
GTL*	Go To Local	(via C)
IFC	Interface Clear	C
LLO*	Local Lockout	(via C)
MSA*	My Secondary Address	(via C)
MTA*	My Talk Address	(via C)
PPC*	Parallel Poll Configure	(via C)
PPD*	Parallel Poll Disable	(via C)
PPE*	Parallel Poll Enable	(via C)
PPU*	Parallel Poll Unconfigure	(via C)
REN	Remote Enable	C
RFD	Ready For Data	AH
SDC*	Selected Device Clear	(via C)
SPD*	Serial Poll Disable	(via C)
SPE*	Serial Poll Enable	(via C)
SRQ	Service Request	SR
TCT*	Take Control	(via C)
UNL*	Unlisten	(via C)
UNT*	Untalk	(via C)

*** Multiline messages.**

Table B–4: ASCII and GPIB Code Chart

B7 B6 B5 BITS	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	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
0 0 0 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
0 0 1 0	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
0 0 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
0 1 0 0	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
0 1 0 1	5 *PPC* ENQ 5 5	25 *PPU* NAK 15 21	45 *5* % 25 37	65 *21* 5 35 53	105 *5* E 55 69	125 *21* U 55 85	145 *5* e 65 101	165 *21* u 75 117
0 1 1 0	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
0 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 0 0 0	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 0 0 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 0 1 0	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 0 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 0 0	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 0 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 0	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 488.1-1987
ISO STD 646-2973

GPIB Signal Line Definitions

Figure B–2 shows how the sixteen active signal lines on the GPIB are functionally divided into three component buses: an 8-line data bus, a 3-line data byte transfer control (handshake) bus, and a 5-line general interface management bus.

The data bus contains eight bidirectional signal lines, DI01 through DI08. Information in the form of data bytes is transferred over this bus. A handshake timing sequence between the enabled talker and the enabled listeners on the three-line data transfer control bus transfers one data byte (eight bits) at a time. These data bytes are sent and received in a byte-serial, bit-parallel fashion.

Since the handshake sequence is an asynchronous operation (no clock signal on the bus), the data transfer rate is only as fast as the slowest instrument involved in a data byte transfer.

A talker cannot place data bytes on the bus faster than the slowest listener can accept them.

Figure B–3 illustrates the flow of data bytes on the bus when a typical controller sends ASCII data to an assigned listener. The first data byte (decimal 44) enables an instrument at address 12 as a primary listener. The second data byte (decimal 108) is optional; for example, enabling a plug-in device at secondary address 12 as the final destination of the data to follow. The data is the two ASCII characters A and B (decimal 65 and decimal 66). Note that the ATN line is asserted for the first two data bytes and unasserted for the device-dependent character to indicate the last data byte in the message.

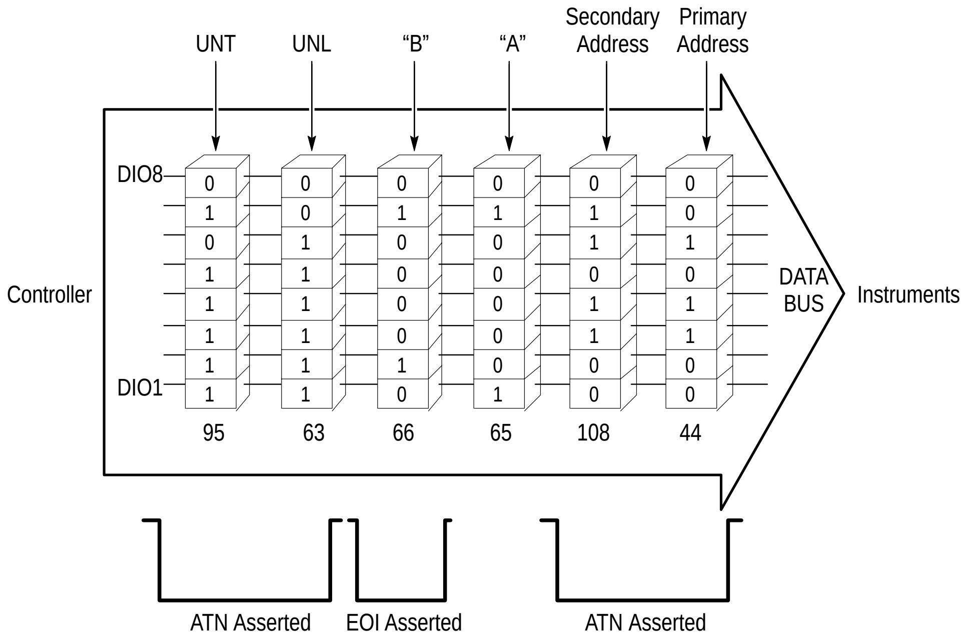


Figure B–3: Example of Data Byte Traffic

To complete the sequence, the controller activates the ATN line again and sends the universal Unlisten (UNL) and Untalk (UNT) commands to clear the bus. Six handshake cycles on the data transfer control bus are required to send the six data bytes.

Transfer Bus (Handshake)

Each time a data byte is transferred over the data bus, an enabled talker and all enabled listeners execute a handshake sequence via signal lines DAV, NRFD, and NDAC (see Figure B–4 — the ATN line is shown to illustrate the controller’s role in the process).

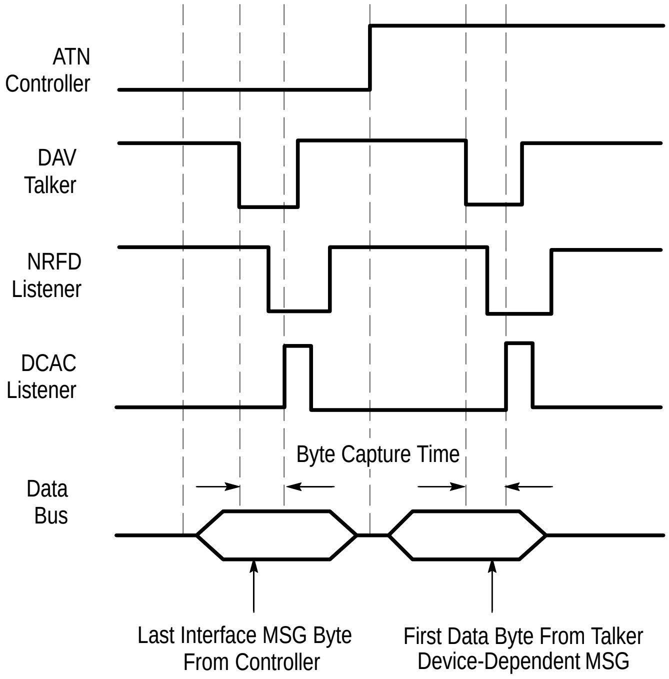


Figure B–4: Handshake Timing Sequence, Idealized

DAV (Data Valid). The DAV signal line is asserted by the talker after the talker places a data byte on the data bus. When asserted (low), DAV tells each assigned listener that a new data byte is on the bus. The talker is inhibited from asserting DAV as long as any listener holds the NRFD signal line asserted.

NRFD (Not Ready For Data). An asserted NRFD signal line indicates one or more of the assigned listeners are not ready to receive the next data byte from the talker. When all of the assigned listeners for a particular data byte transfer have released NRFD, the NRFD line becomes unasserted (high). When NRFD goes high, the RFD message (Ready For Data) tells the talker it may place the next data byte on the data bus.

NDAC (Not Data Accepted). Each assigned listener holds the NDAC signal line asserted until the listener accepts the data byte currently on the bus. When all assigned listeners have accepted the current data byte, the NDAC signal line becomes unasserted (high) telling the talker to remove the data byte from the bus. The DAC message (Data Accepted) tells the talker that all assigned listeners have accepted the current data byte.

Note that one handshake cycle transfers one data byte. The listeners must then reset the NRFD line high and the NDAC line low before the talker asserts DAV for the next data byte transfer. An invalid state on the bus is indicated when NRFD and NDAC are high at the same time.

Management Bus

The management bus is a group of five signal lines that are used to control the operation of the IEEE Std 488.1 (GPIB) Digital Interface.

IFC (Interface Clear)

The system controller is the only instrument on the bus allowed to assert IFC. IFC is asserted for greater-than 100 μs to place all instruments in a predetermined state. While IFC is being sent, only the DCL (Device Clear), LLO (Local Lockout), PPU (Parallel Poll Unconfigure), and REN (Remote Enable) interface messages (universal commands) will be recognized.

ATN (Attention)

The controller in charge is the only instrument on the bus allowed to assert ATN. ATN is asserted when an instrument connected to the bus is being enabled as a talker or listener, or when sending other interface control messages. As long as the ATN line is asserted (low), only instrument address codes and interface control messages are sent over the bus. When the ATN line is unasserted, only those instruments enabled as a talker and listener can send and receive data over the bus.

SRQ (Service Request)

Any instrument connected to the bus can request the controller’s attention by asserting the SRQ line. The controller responds by asserting ATN and executing a serial poll routine to determine which instrument is requesting service. The instrument requesting service responds with a device-dependent status byte with bit seven asserted.

When the instrument requesting service is found, program control is transferred to a service routine for that instrument. When the service routine is completed, program control returns to the main program. The controller does not have to see the SRQ line asserted to perform a polling routine; it may do so whenever a program requires it.

REN (Remote Enable)

The system controller asserts the REN signal line whenever the interface system operates under remote program control. The REN signal causes an instrument on the bus to select between two alternate sources of programming data. It is used with other interface control messages such as LLO (Local Lockout) or GTL (Go To Local). A remote-local interface function indicates to an instrument that the instrument will use either information input from the interface (remote) or information input by the operator via the front panel controls (local).

EOI (End Or Identify)

A talker can use the EOI signal line to indicate the end of a data transfer sequence. The talker asserts EOI as the last byte of data is transmitted. In this case, the EOI line is essentially a ninth data bit and must observe the same settling time as the data on the data bus.

When an instrument controller is listening, it assumes that a data byte sent with EOI asserted is the last data byte in the complete message. When the instrument controller is talking, it may assert the EOI signal line as the last data byte is transferred. The EOI line is also asserted when the ATN line is true if the controller conducts a parallel polling sequence on the bus. The EOI line is not used for a serial polling sequence.

Interface Functions and Messages

The ten major interface functions listed in Table B–1 provide a variety of capabilities and options for an instrumentation system. These functions may be implemented in, or for, any particular instrument with instrument hardware or with a programming routine (software).

Only those functions necessary for an instrument’s purpose must be implemented by the instrument’s designer. An instrument will seldom have all ten interface functions. For example, an instrument generally does not need to implement the Parallel Poll (PP) function if the instrument can respond to a serial polling sequence from the controller in charge of the GPIB system.

The interface functions and their relationship to the interface control messages in Table B–4 are discussed below. All interface control messages discussed are sent and received over the GPIB when the ATN line is asserted (low).

RL (Remote-Local Function)

The RL function provides an instrument with the capability to select between two sources of input information. This function indicates to the instrument that its internal device-dependent functions are to respond to information input from the front panel (Local), or to corresponding programming information from the GPIB (Remote). Only the system controller is permitted to assert the REN (Remote Enable) line, whether or not it is the controller in charge at the time.

When the system controller asserts the REN line, an instrument on the GPIB goes to a remote mode when it is addressed as a listener with its listen address. An instrument remains in a remote mode until the REN line is released (high), or an optional front-panel switch on the instrument is activated to request the local mode, or a GTL (Go To Local) command is received while the instrument is enabled as a listener.

The controller can also disable the instrument’s front-panel “return to local” switch(es) by sending a LLO (Local Lockout) command. The LLO command must be preceded or followed by a listen address (MLA) to cause the instrument to go to a remote mode with front-panel lockout. The UNL (Unlisten) command does not return an instrument to the local mode.

When the REN line goes false it must be recognized by all instruments on the bus, and they must go to the local mode within 100 μs. If data bytes are still being placed on the bus when REN goes false, the system program should ensure that the data bytes are sent and received with the knowledge that the system is in a local mode as opposed to remote mode.

T/TE and L/LE (Talker and Listener Functions)

The T/TE and L/LE functions are independent of each other, although they are discussed under one heading.

The T (Talker) and TE (Talker Extended) functions provide an instrument and its secondary devices, if any, with the capability to send device-dependent data over the GPIB (or, in case of a controller, the capability to send device-dependent program data) over the GPIB. The Talker (T) function is a normal function for a talker and uses only a one-byte primary address code called MTA (My Talk Address). The Talker Extended (TE) function requires a two-byte address code, an MTA code followed by the second byte called MSA (My Secondary Address).

Only one instrument in the GPIB system can be in the active talker state at one time. A non-controller begins talking when ATN is released and continues its talker status until an Interface Clear (IFC) message occurs or an Untalk (UNT) command is received from the controller in charge. The instrument will stop talking and listen any time the controller in charge asserts ATN.

One or more instruments on the bus can be programmed for the L (Listener) function by using their specific primary listen address (called MLA). Some of the instruments interfaced to the bus may be programmed for the LE (Listener Extended) function, if implemented. The LE function requires a two-byte address code. No L or LE function is active during the time that ATN is asserted.

An instrument may be a talker only, a listener only, or implement all functions. All talker and listener functions must respond to ATN within 200 ns. They must also respond to IFC in less than 100 μs. In any case, its address code has the form X10TTTTT for a talker and X01LLLLL for a listener. For instruments with both T and L functions, the T-bit binary values are usually equal to the binary value of the L bits. Before applying power to the system, the system operator sets these five least significant bits by means of an address switch on each instrument. The controller's address code may be implemented in software.

The system program, run from the controller, designates the primary talker and primary listener status of the desired instruments by coding data bits 6 and 7. These bits are set to 1 and 0, respectively, for a talker and 0 and 1, respectively, for a listener. Secondary talk and listen addresses (or commands) are represented by the controller sending both data bits (6 and 7) as a logical 1. The controller may listen to bus traffic without actually addressing itself over the bus.

SH and AH (Source and Acceptor Handshake Functions)

The SH and AH functions are independent of each other, although they are discussed under one heading.

The SH (Source Handshake) function guarantees proper transmission of data, while the AH (Acceptor Handshake) function guarantees proper reception of data. The interlocked handshake sequence between these two functions guarantees asynchronous transfer of each data byte. The handshake sequence is performed via the NRFD, DAV, and NDAC signal lines on the bus (see Figure B–4). Both functions must respond to ATN within 200 ns.

The SH function must wait for the RFD (Ready For Data) message plus a minimum additional delay of 2 μs before asserting DAV. This delay allows the data to settle on the data bus. If three-state drivers are used, the settling time is reduced to RFD plus 1.1 μs. Faster settling times are allowed under special conditions; refer to warning notes in the IEEE 488.1 standard. The time required for the AH function to accept an interface message byte depends upon the implementation of the function.

DC (Device Clear Function)

The DCL (Device Clear) function allows the controller in charge to “clear” any or all instruments on the bus. The controller (under program direction) asserts ATN and sends either the universal DCL (Device Clear) command or the SDC (Selected Device Clear) command.

When the DCL message is received, all instruments on the bus must clear or initialize their internal device functions. When the controller sends the SDC command, only those instruments that have been previously addressed to listen must respond. The IEEE 488.1 standard does not specify the settings an instrument must go to as a result of receiving the DCL or SDC command. In general, these commands are used only to clear the GPIB interface circuits within an instrument.

DT (Device Trigger Function)

The DT (Device Trigger) function allows the controller in charge to start the basic operation specified for an instrument or group of instruments on the bus. The IEEE 488.1 standard does not specify the basic operation an instrument is to perform when it receives the GET (Group Execute Trigger) command. To issue the GET command, the controller asserts ATN, sends the listen addresses of the instruments that are to respond to the trigger, and then sends the GET message.

Once an instrument starts its basic operation in response to GET, the instrument must not respond to subsequent trigger-state transitions until the current operation is complete. Only after completing the operation can the instrument repeat its basic operation in response to the next GET message. Thus, the basic operating time is the major factor that determines how fast the instrument(s) can be repeatedly “triggered” by commands from the bus.

C, SR, and PP (Controller, Service Request, and Parallel Poll Functions)

The C (Controller) function provides the capability to send primary talk and listen addresses, secondary addresses, universal commands, and addressed commands to all instruments on the bus. The Controller function also provides the capability to respond to a service request message (SRQ) from an instrument, or to conduct a parallel poll routine to determine the status of any instruments on the bus that have the Parallel Poll (PP) function implemented. If an instrumentation system has more than one controller, only the system controller is allowed to assert the IFC (Interface Clear) and REN (Remote Enable) lines during system operation, whether or not it is the controller in charge at the time.

If one controller requests system control from another controller, and it receives a message from another controller to send REN, the system controller must verify that the REN line remains unasserted (false) for at least 100 μs before asserting REN. The time interval that REN is asserted depends on the remote programming sequence and will vary with the program. The IFC line must be asserted for at least 100 μs.

The Controller function has specified time intervals for certain operations. For example, the execution time for parallel polling instruments on the bus cannot be less than 2 μs. If the controller is in the controller active wait state and does not receive an internal message to conduct a parallel poll, it must wait for at least 1.5 μs before going to the controller active state. This delay gives the NRFD, NDAC, and EOI lines sufficient time to assume their valid states.

The controller also requires a delay of at least 2 μs (1.1 μs for tri-state drivers) so the instruments can detect that the ATN line is asserted before the controller places the first data byte on the bus.

Taking Control (Asynchronous or Synchronous)

All data bytes transmitted over the GPIB with the ATN line asserted are interpreted as system control information. Asserting ATN directly at any moment is an asynchronous operation with respect to the bus and may cause loss of data if a handshake cycle is in progress. To prevent loss of data, a controller can take control synchronously by monitoring the Transfer Bus and only asserting ATN when DAV is unasserted (false).

As a controller in charge, the system controller (program) may pass control to any other instrument in the system capable of acting as a controller. The controller in charge first addresses the other controller as a talker and then sends the TCT (Take Control) command. The other controller then becomes the controller in charge when ATN is released.

Performing a Serial Poll

The controller-in-charge may conduct a serial poll at any time, whether or not an instrument on the bus has asserted the SRQ line. Most instruments (but not all) have the Service Request (SR) function.

To perform a serial poll, the controller first asserts ATN and issues the Untalk (UNT) and Unlisten (UNL) commands. The controller then sends the Serial Poll Enable (SPE) command, followed by the talk address of the first instrument to be polled. Next the controller releases ATN and the addressed talker responds by sending its status byte over the bus. If the addressed talker has requested service, it must assert bit seven of the status byte and encode the remaining seven bits to indicate the reason for asserting SRQ.

Status bytes are device-dependent and are not specified in the IEEE 488.1 standard. An addressed instrument will release its SRQ line when serially polled, but other instruments may still hold SRQ asserted. When the controller has read the status byte of an addressed instrument, it reasserts ATN and addresses the next instrument to talk, then releases ATN and receives the instrument's status byte. The routine continues until the controller no longer detects the SRQ line asserted. At this time the controller should send the Serial Poll Disable (SPD) message and, optionally, send the Untalk (UNT) message to release the last active talker.

Performing a Parallel Poll

The Parallel Poll (PP) function provides an instrument with the capability to present one, and only one, bit of status information to the controller without being previously addressed to talk. The parallel polling capability requires a commitment by the system program to periodically conduct a parallel poll sequence.

When an instrument responds to a parallel poll, the single data bit presented to the controller may or may not indicate a need for service. If the data bit is used as a service request indication, the controller should perform a serial poll in order to obtain a complete status byte with more information (if the device has the SR function implemented).

Before an instrument can respond to a parallel poll, the GPIB system must first be configured. In a typical sequence, the controller first sends an Unlisten (UNL) command to clear the bus of listeners, then the listen address of the device to be configured. Following this, the controller sends the PPC (Parallel Poll Configure) command followed by a PPE (Parallel Poll Enable) message. The PPE message contains coded information that tells the selected instrument which data line will carry the PP status bit for that device. This entire sequence is repeated for each instrument to be configured.

The PPE message(s) sent by the controller has the form of X110SPPP. Bit four (S) is called the sense bit and the three least significant bits (PPP) represent an octal number (0 through 7) that corresponds to a specific line on the data bus. An instrument must assert this data bus line when its internal status has the same value as the sense bit (S may equal 1 or 0).

The actual parallel poll takes place after each instrument has been completely configured. The concept is to have the controller receive one data byte that contains status information on all of the addressed instruments. To receive this status byte, the controller asserts EOI and the ATN line. The assertion of EOI may coincide with ATN or occur later, so long as both are asserted. This may occur any time after the last PPE message. The controller then reads data bus lines while ATN and EOI are asserted to interpret the status of all selected instruments.

To conclude the parallel poll, the controller releases EOI and then ATN. The instrument(s) does not need to be configured for each subsequent parallel poll. The PPU (Parallel Poll Unconfigure) command will clear all device configurations and prevent them from responding to future polls. The PPD (Parallel Poll Disable) command accomplishes essentially the same results, except that the PP function remains in the “configured” state. PPU is a universal command (all instruments) while PPD is used with PPC and becomes an addressed command (only those devices selected with PPC will accept PPD).

Index

Index