GPIB

GPIB-232/485CT-A
User Manual

November 1999 Edition
Part Number 370898A-01

Worldwide Technical Support and Product Information

www.ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
China 0755 3904939, Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,
Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406,
Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456, Mexico (D.F.) 5 280 7625,
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, Norway 32 27 73 00, Poland 48 22 528 94 06,
Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085, Sweden 08 587 895 00,
Switzerland 056 200 51 51, Taiwan 02 2377 1200, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to techpubs@ni.com

Important Information

Warranty

The GPIB-232/485CT-A is warranted against defects in materials and workmanship for a period of two years from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Compliance

FCC/Canada Radio Frequency Interference Compliance*

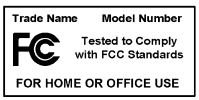
Determining FCC Class

The Federal Communications Commission (FCC) has rules to protect wireless communications from interference. The FCC places digital electronics into two classes. These classes are known as Class A (for use in industrial-commercial locations only) or Class B (for use in residential or commercial locations). Depending on where it is operated, this product could be subject to restrictions in the FCC rules. (In Canada, the Department of Communications (DOC), of Industry Canada, regulates wireless interference in much the same way.)

Digital electronics emit weak signals during normal operation that can affect radio, television, or other wireless products. By examining the product you purchased, you can determine the FCC Class and therefore which of the two FCC/DOC Warnings apply in the following sections. (Some products may not be labelled at all for FCC, if so the reader should then assume these are Class A devices.)

FCC Class A products only display a simple warning statement of one paragraph in length regarding interference and undesired operation. Most of our products are FCC Class A. The FCC rules have restrictions regarding the locations where FCC Class A products can be operated.

FCC Class B products display either a FCC ID code, starting with the letters **EXN**, or the FCC Class B compliance mark that appears as shown here on the right.



The curious reader can consult the FCC web site `http://www.fcc.gov` for more information.

FCC/DOC Warnings

This equipment generates and uses radio frequency energy and, if not installed and used in strict accordance with the instructions in this manual and the CE Mark Declaration of Conformity**, may cause interference to radio and television reception. Classification requirements are the same for the Federal Communications Commission (FCC) and the Canadian Department of Communications (DOC).

Changes or modifications not expressly approved by National Instruments could void the user's authority to operate the equipment under the FCC Rules.

Class A

Federal Communications Commission

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Canadian Department of Communications

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

Class B

Federal Communications Commission

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful

interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Canadian Department of Communications

This Class B digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.

Cet appareil numérique de la classe B respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

European Union - Compliance to EEC Directives

Readers in the EU/EEC/EEA must refer to the Manufacturer's Declaration of Conformity (DoC) for information** pertaining to the CE Mark compliance scheme. The Manufacturer includes a DoC for most every hardware product except for those bought for OEMs, if also available from an original manufacturer that also markets in the EU, or where compliance is not required as for electrically benign apparatus or cables.

* Certain exemptions may apply in the USA, see FCC Rules §15.103 **Exempted devices**, and §15.105(c). Also available in sections of CFR 47.

** The CE Mark Declaration of Conformity will contain important supplementary information and instructions for the user or installer.

Contents

Chapter 6
Programming in G Mode

Figures

Tables

About This Manual

The *GPIB-232/485CT-A User Manual* describes the features, functions, and operation of the GPIB-232CT-A and GPIB-485CT-A. This manual assumes that you have a general knowledge of RS-232 or RS-485 serial communications and the GPIB.

Conventions

The following conventions appear in this manual:

This icon denotes a note, which alerts you to important information.

This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.

bold — Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

GPIB-232/485CT-A — *GPIB-232/485CT-A* refers to either the GPIB-232CT-A or GPIB-485CT-A box.

IEEE 488 and IEEE 488.2 — *IEEE 488* and *IEEE 488.2* refer to the ANSI/IEEE Standard 488.1-1987 and the ANSI/IEEE Standard 488.2-1992, respectively, which define the GPIB.

italic — Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

`monospace` — Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

`monospace bold` — Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

RS-232	*RS-232* refers to the ANSI/EIA-232-C standard.
RS-422	*RS-422* refers to the EIA-422-A standard.
RS-485	*RS-485* refers to the EIA-485 standard.

Related Documentation

The following documents contain information that you might find helpful as you read this manual:

- ANSI/EIA-232-D, *Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*
- EIA-485, *Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*
- EIA/RS-422-A, *Electrical Characteristics of Balanced Voltage Digital Interface Circuits*
- ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*
- ANSI/IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols, and Common Commands*

1

Hardware Overview

This chapter lists what you need to get started and optional equipment you can order, and briefly describes the GPIB-232/485CT-A.

What You Need to Get Started

Before you install your GPIB hardware, make sure you have all of the following items:

- ❑ One of the following controllers, which is included in your kit:
  - GPIB-232CT-A, 100–120 VAC
  - GPIB-232CT-A, 220–240 VAC
  - GPIB-232CT-A, DC
  - GPIB-485CT-A, 100–120 VAC
  - GPIB-485CT-A, 220–240 VAC
  - GPIB-485CT-A, DC
- ❑ If you have an AC version, one of the following power cords, which is included in your kit:
  - U.S. standard power cord
  - Swiss power cord
  - Australian power cord
  - Universal European power cord
  - North American power cord
  - U.K. power cord
- ❑ If you have a DC version, one of the following DC power supplies, which is included in your kit:
  - Wall-mount power supply (100–120 VAC, 9 V, 1 A)
  - Desktop power supply (220–240 VAC, 9 V, 1 A)

- One of the following shielded cables, compatible with IBM PC, which you can purchase from National Instruments:
  - RS-232 DTE-to-DTE cable (1, 2, or 4 m) for the GPIB-232CT-A
  - RS-485 null-modem cable (1, 2, or 4 m) for the GPIB-485CT-A
- Type X2 double-shielded GPIB cable (1, 2, or 4 m), which you can purchase from National Instruments

Caution To meet FCC emission limits for this device, you must use a Type X2 double-shielded GPIB cable. If you operate this equipment with a non-shielded cable, it may interfere with radio and television reception.

GPIB-232CT-A Hardware Overview

The GPIB-232CT-A is a high-performance serial-to-GPIB interface. It enables a computer with an RS-232 serial port to be a Talker, Listener, or Controller on the GPIB. The GPIB-232CT-A is also capable of interfacing RS-232 instruments and peripherals to the GPIB.

The GPIB-232CT-A has all the software and logic required to implement the physical and electrical specifications of the IEEE 488 and RS-232 standards. It can interpret and execute high-level commands that you send to it over the serial port, performing GPIB-to-RS-232 protocol conversions. The GPIB-232CT-A also conforms to all versions of the IEEE 488 standard, including IEEE 488.2. The NAT4882 Controller chip implements all IEEE 488 Talker/Listener/Controller functionality.

GPIB-485CT-A Hardware Overview

The GPIB-485CT-A is a high-performance serial-to-GPIB interface. It enables a computer with an RS-485 or RS-422 serial port to be a Talker, Listener, or Controller on the GPIB. The GPIB-485CT-A is also capable of interfacing RS-485 or RS-422 instruments and peripherals to the GPIB. The GPIB-485CT-A does not support any multidrop protocols.

The GPIB-485CT-A has all the software and logic required to implement the physical and electrical specifications of the IEEE 488, RS-485, and RS-422 standards. It can interpret and execute high-level commands that you send to it over the serial port, performing all GPIB-to-RS-485 and GPIB-to-RS-422 protocol conversions. The GPIB-485CT-A also conforms to all versions of the IEEE 488 standard, including IEEE 488.2. The

NAT4882 Controller chip implements all IEEE 488 Talker/Listener/Controller functionality.

AC Version Front Panel

The power switch, fuse holder, and power cord receptacle are located on the GPIB-232/485CT-A front panel, on the AC version only. Figure 1-1 shows the front panel of the AC version.

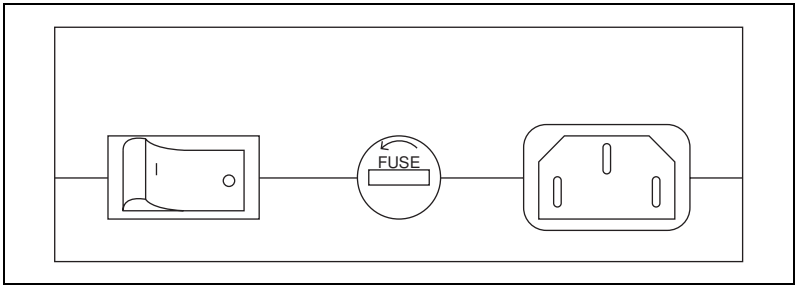


Figure 1-1. The AC Version Front Panel

Top Panel

The six light-emitting diodes (LEDs) are located on the GPIB-232/485CT-A top panel. Figure 1-2 shows the top panel.

Note The following figures show only the GPIB-232CT-A, but the GPIB-485CT-A is similar.

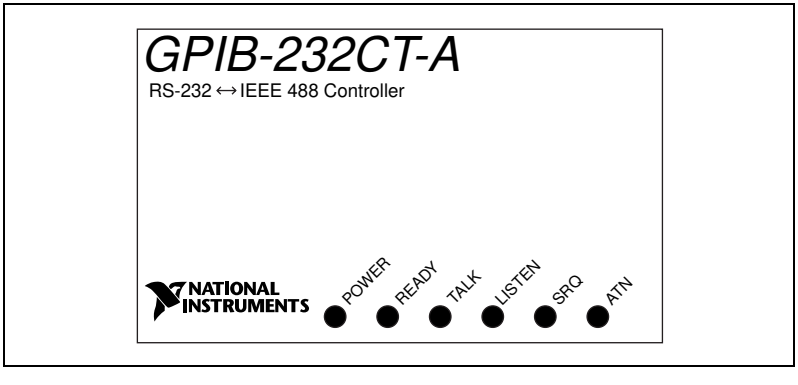


Figure 1-2. The Top Panel

The LEDs show the current status of the GPIB-232/485CT-A at all times. Table 1-1 describes each LED.

Table 1-1. LED Descriptions

LED	Indication
POWER	Indicates that power to the unit has been applied and the ON/OFF switch is in the ON position.
READY	Indicates that the power-on self-test has passed successfully and the unit is ready to operate.
TALK	Indicates that the GPIB-232/485CT-A is addressed as a GPIB Talker.
LISTEN	Indicates that the GPIB-232/485CT-A is addressed as a GPIB Listener.
SRQ	Indicates that the GPIB signal line **SRQ*** is asserted.
ATN	Indicates that the GPIB signal line **ATN*** is asserted.
* indicates that the signal is active low (negative logic or low when asserted).	

Rear Panel

The configuration switches are located on the rear panel of the GPIB-232/485CT-A. Figure 1-3 shows the rear panel of the AC version. Figure 1-4 shows the rear panel of the DC version. The unmarked DIP switches are reserved for future development and should remain in the OFF position.

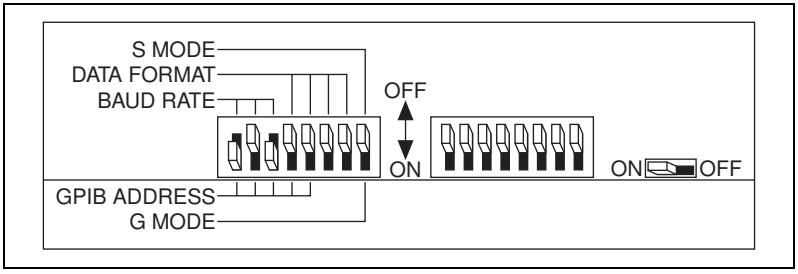


Figure 1-3. The DC Version Rear Panel

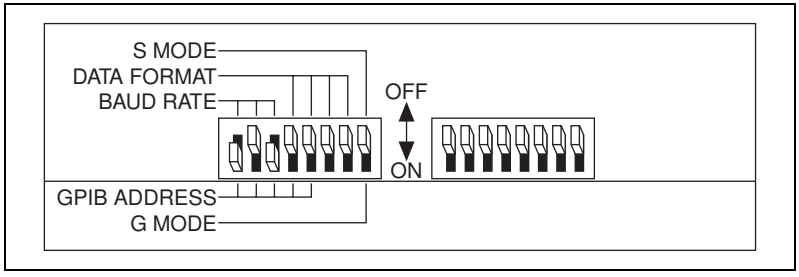


Figure 1-4. The AC Version Rear Panel

Side Panels

The GPIB connector and the serial connector are mounted on opposite side panels. On the DC version, the DC power jack is located next to the serial connector.

Figure 1-5 shows the location of the serial and GPIB connectors and the DC power jack.

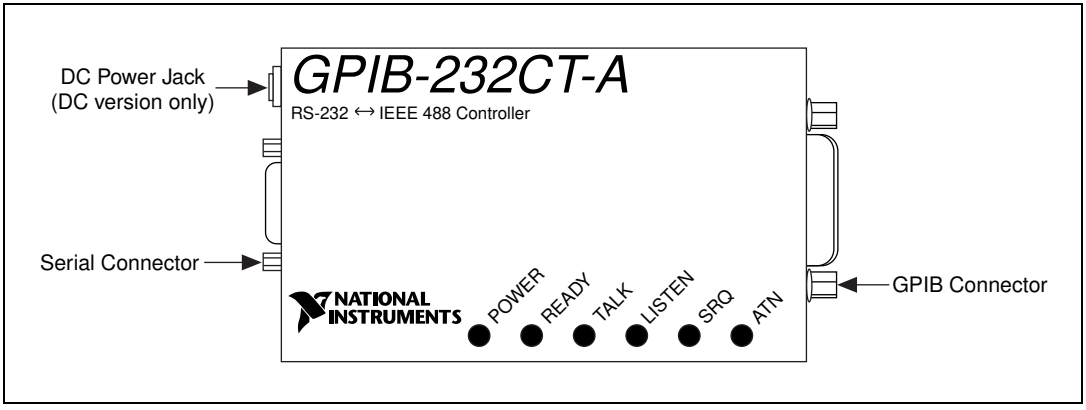


Figure 1-5. Location of the Connectors and the DC Power Jack

RS-232 Connector

The RS-232 port on the GPIB-232CT-A is configured as a DTE (Data Terminal Equipment) and uses a standard 9-pin shielded D-Subminiature male connector with screwlock assemblies. The RS-232 connector accepts standard 9-pin D-Subminiature female connectors. Figure 1-6 shows a diagram of the RS-232 connector and the signals supported. For more information on the RS-232 signals refer to Appendix D, *Interfacing to an RS-232 Device*.

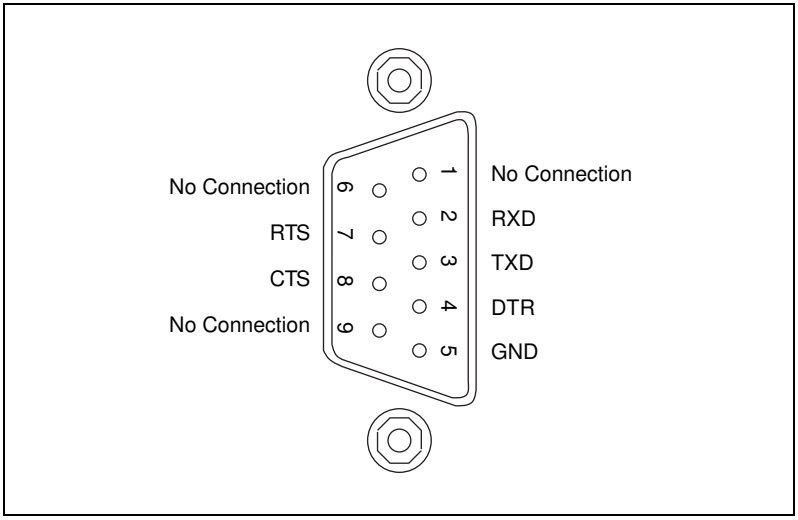


Figure 1-6. The RS-232 Connector and Signal Designations

RS-485 Connector

The RS-485 port on the GPIB-485CT-A uses a standard 9-pin shielded D-Subminiature male connector with screwlock assemblies. The RS-485 connector accepts standard 9-pin D-Subminiature female connectors. Figure 1-7 shows a diagram of the serial connector and the signals supported. For more information on the RS-485 signals refer to Appendix E, *Interfacing to an RS-485 Device*.

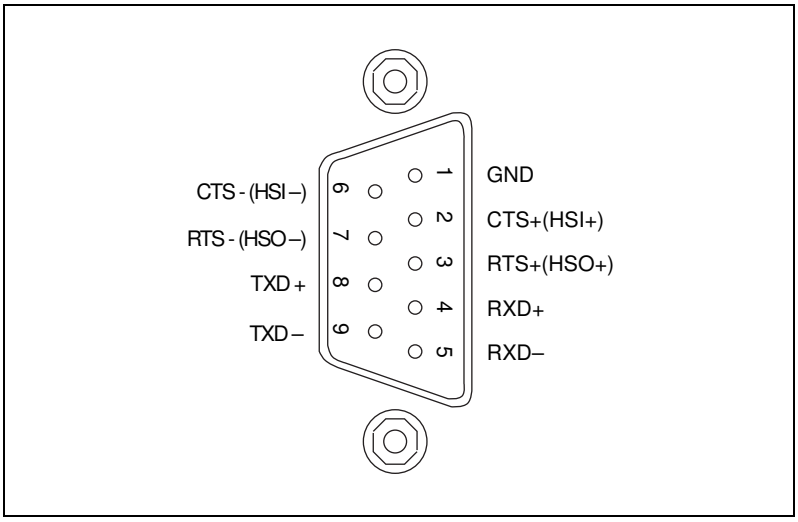


Figure 1-7. The RS-485 Connector and Signal Designations

GPIB Connector

The GPIB connector is a standard 24-pin shielded Champ female connector with metric screwlock hardware. Figure 1-8 shows a diagram of the GPIB connector and the signals supported. A * suffix indicates that the signal is active low. Refer to Appendix F, *GPIB Basics*, for more information about the GPIB signal lines.

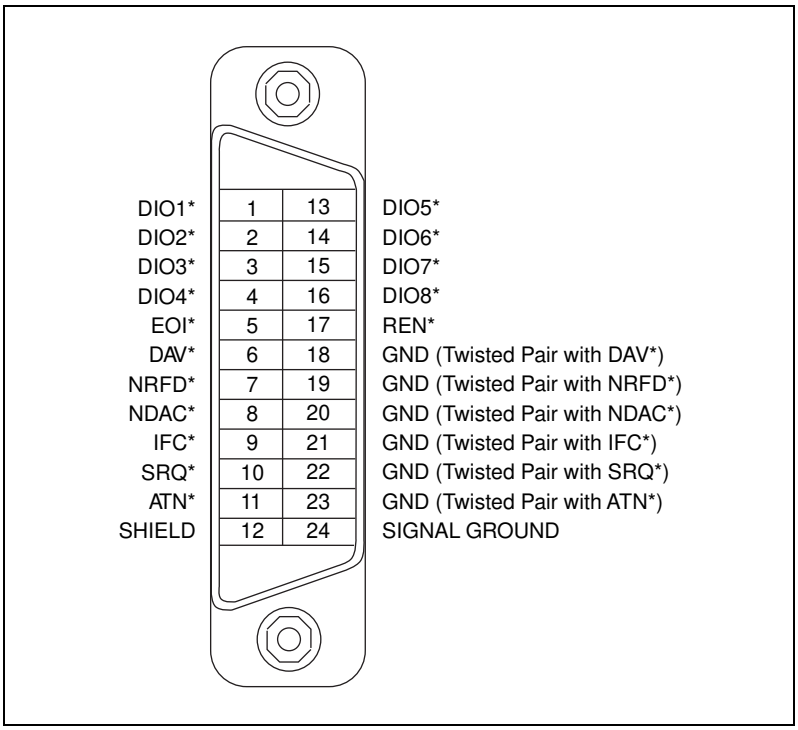


Figure 1-8. The GPIB Connector and Signal Designations

2

Operating in S Mode and G Mode

This chapter helps you determine which mode of operation, S mode or G mode, you should use. It also describes data buffering and handshaking schemes.

Choosing Between S Mode and G Mode

The GPIB-232/485CT-A can be connected to a serial device and one or more GPIB devices. The way you use the serial device in your system setup determines which mode of operation you should use. If the serial device is the Controller, you should use S mode. If the serial device is a Talker/Listener only, and a GPIB device is the Controller, you should use G mode.

Operating in S Mode

The GPIB-232/485CT-A should be configured to operate in S mode if your serial device acts as a Controller in the GPIB system, addressing devices, and performing other GPIB Controller functions. In S mode operation, you can use the NI-488.2 software.

Figure 2-1 shows an example of a setup using S mode. The GPIB-232/485CT-A is connected to a PC, which is controlling a GPIB plotter.

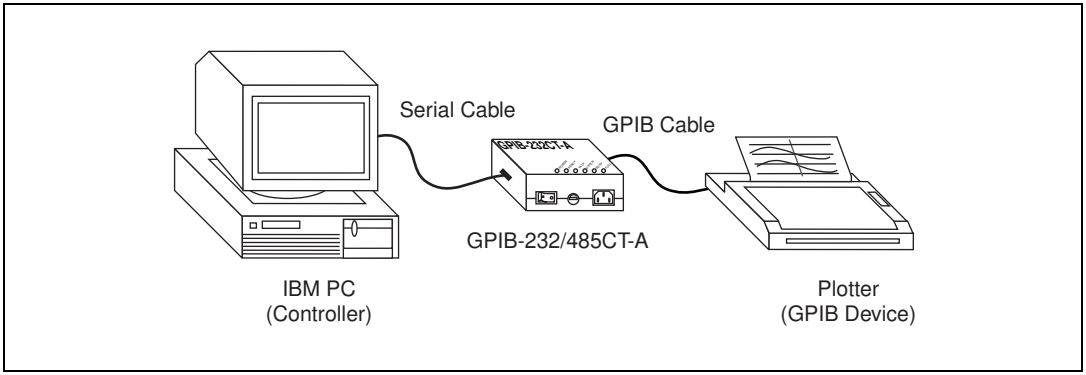


Figure 2-1. Example of S Mode System Setup

Refer to Chapter 3, *Installing and Configuring Your Controller*, for detailed information on setting up your GPIB-232/485CT-A to operate in S mode. Refer to Chapter 4, *Programming in S Mode*, and Chapter 5, *S Mode Functions*, for information on programming the GPIB-232/485CT-A in S mode.

Operating in G Mode

The GPIB-232/485CT-A should be configured to operate in G mode if your serial device acts only as a Talker and/or Listener while a GPIB device manages the system, sending and receiving data to and from the serial device.

Figure 2-2 shows an example of a setup using G mode. The GPIB-232/485CT-A is connected to a serial printer, which is programmed by the GPIB Controller.

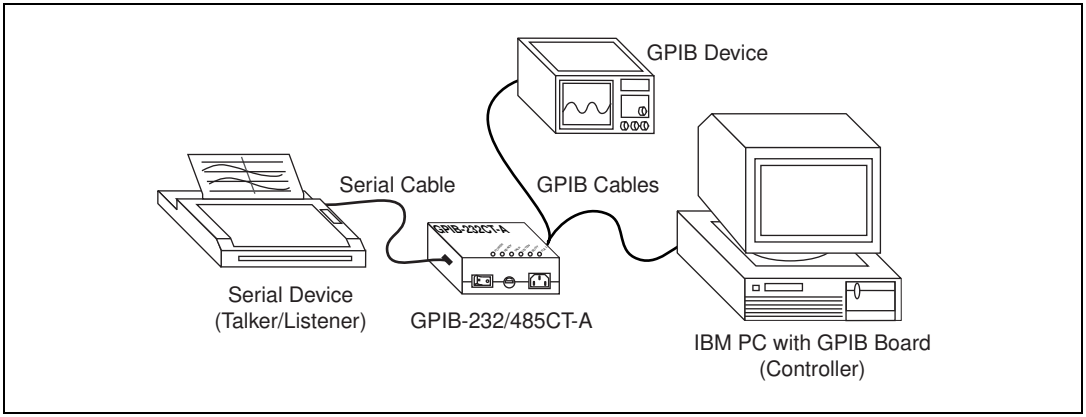


Figure 2-2. Example of G Mode System Setup

Refer to Chapter 3, *Installing and Configuring Your Controller*, for detailed information on setting up your GPIB-232/485CT-A to operate in G mode. Refer to Chapter 6, *Programming in G Mode*, and Chapter 7, *G Mode Functions*, for information on programming in G mode.

Data Buffering and Handshaking Schemes

Two protection mechanisms are used to ensure that the GPIB-232/485CT-A does not lose incoming serial data: data buffering and handshaking.

The GPIB-232/485CT-A has an internal RAM buffer that stores incoming serial data until it can output the data to the GPIB port. The size of this RAM buffer determines how much serial data the GPIB-232/485CT-A can accept before the buffer is completely full.

When its RAM buffer is nearly full, the GPIB-232/485CT-A can handshake with the serial host to stop data transmission. When the buffer is almost empty, the GPIB-232/485CT-A can again handshake with the serial host to start data transmission. The GPIB-232/485CT-A is capable of using both the XON/XOFF software handshaking and the hardware handshaking protocols.

Hardware Handshaking

The hardware handshake function is always active during serial data transfers and uses the Request to Send (RTS on the GPIB-232CT-A, RTS+ and RTS– on the GPIB-485CT-A) and Clear to Send (CTS on the GPIB-232CT-A, CTS+ and CTS– on the GPIB-485CT-A) signal lines. When the GPIB-232/485CT-A is ready to accept serial data, it asserts the RTS line(s). RTS remains asserted until the GPIB-232/485CT-A data buffer is almost full. At this point, the GPIB-232/485CT-A unasserts the RTS line(s), signaling to the serial host that it is no longer ready to accept data. The serial host should monitor the RTS line(s) and suspend data transmission whenever RTS becomes unasserted. The GPIB-232/485CT-A asserts RTS when it is again ready to receive serial data.

The GPIB-232/485CT-A is also able to suspend transmission when the serial device is no longer ready to accept data. The GPIB-232/485CT-A is configured to immediately stop transmission of serial data when CTS becomes unasserted. It resumes transmission when CTS is reasserted.

Because most serial devices use the same form of hardware handshaking as the GPIB-232/485CT-A, you can achieve bidirectional flow control by using a serial cable that connects the GPIB-232/485CT-A RTS signal(s) to the serial device CTS signal(s). In addition, the serial device RTS signal(s) should be connected to the GPIB-232/485CT-A CTS signal(s). This setup allows each device to monitor the RTS signal(s) of the other device and to suspend transmission when necessary to prevent data loss. Refer to or Appendix D, *Interfacing to an RS-232 Device*, or Appendix E, *Interfacing to an RS-485 Device*, for more information on wiring schemes.

XON/XOFF Software Handshaking

If your serial device does not implement or recognize the hardware handshake scheme, your cable does not support the necessary handshake lines, or your application software requires XON/XOFF handshaking, you might need to enable the XON/XOFF handshaking protocol by using the `xon` command. This handshaking protocol performs the same function as the hardware handshake but does so by sending special control codes over the data lines instead of changing logic levels on dedicated control lines.

When you enable the XON/XOFF protocol, the GPIB-232/485CT-A sends the XOFF character (decimal 19 or <CTRL-S>) before the internal buffer overflows. When the GPIB-232/485CT-A is able to start receiving characters again, it sends the XON character (decimal 17 or <CTRL-Q>). Similarly, if the GPIB-232/485CT-A is transmitting data and receives the

XOFF character, it suspends transmission of any further data until it receives the XON character.

If you are transmitting binary data (as opposed to 7-bit ASCII), do not configure the GPIB-232/485CT-A to use XON/XOFF software handshaking. Because the binary data could contain any binary sequence, including decimal 19 (<Ctrl-S>) or decimal 17 (<Ctrl-Q>), the GPIB-232/485CT-A would not be able to distinguish between data values or handshake control codes. If XON/XOFF software handshaking were enabled in this case, the GPIB-232/485CT-A would handshake erratically.

3

Installing and Configuring Your Controller

This chapter contains detailed instructions for connecting and configuring your GPIB-232/485CT-A.

Check the Hardware Configuration

The AC versions of the GPIB-232/485CT-A are shipped with a 100–120 V or 220–240 V internal power supply. The DC versions of the GPIB-232/485CT-A are shipped with a 100–120 V or 220–240 V, wall-mount or desktop power supply. Before configuring your GPIB-232/485CT-A, verify that the voltage marked on the GPIB-232/485CT-A or on the power supply matches the voltage that is supplied in your area.

Caution Do *not* operate your GPIB-232/485CT-A at any voltage other than the one marked on your GPIB-232/485CT-A. Doing so could damage the unit. Replacement fuses for the AC version must be the proper type and size. For fuse specifications, refer to Appendix A, *Specifications*.

The GPIB-232/485CT-A is shipped with the following default settings:

- S mode
- 7 data bits/character
- 1 stop bit/character
- Parity disabled
- Serial port configured to 9600 baud

If you want to change any of the default settings, you must change the DIP switch settings. To change the settings, shut down your system and then refer to the section *Configure the Hardware* later in this chapter. If you plan to use the default settings, continue with the next section.

Connect the Hardware

Complete the following steps to connect the GPIB-232/485CT-A to your system.

Step 1. Power Off Your System

Power off your system, turn off your computer, and unplug the power cord.

Step 2. Verify That You Have a Null-Modem Serial Cable

You must use a null-modem serial cable (also known as a file transfer cable or a DTE-to-DTE cable) to connect your computer to the GPIB-232CT-A. The following National Instruments cables are null-modem serial cables.

Table 3-1. National Instruments Null-Modem Serial Cables

National Instruments Part Number	Cable Type
182238-01	9-pin to 9-pin, 1 m
182238-02	9-pin to 9-pin, 2 m
182238-04	9-pin to 9-pin, 4 m
181074-10	9-pin to 25-pin, 1 m

See Appendix D, *Interfacing to an RS-232 Device*, and Appendix E, *Interfacing to an RS-485 Device*, for more information on cable pinouts.

Step 3. Connect the Cables

1. Connect the serial cable to the GPIB-232/485CT-A serial connector and securely fasten the holding screws. Connect the other end of the cable to your serial device. Be sure to use only shielded serial cables, and follow the appropriate serial cabling restrictions.
2. Connect the GPIB cable to the GPIB connector on the GPIB-232/485CT-A, and tighten the thumb screws on the connector. Connect the other end to your GPIB device(s). Be sure to use only shielded GPIB cables, and follow all IEEE 488 cabling restrictions.
3. If you have an AC version, connect the power cord to the power receptacle on the front panel of the GPIB-232/485CT-A, then plug the supply into an AC outlet of the correct voltage.

If you have a DC version, connect the DC power plug of the DC power supply to the power jack on the serial end of the GPIB-232/485CT-A, then plug the supply into an AC outlet of the correct voltage.

Step 4. Power On Your System and the GPIB-232/485CT-A

1. Plug in the power cords for your computer system and power on all devices.
2. If you have an AC version, use the front panel rocker switch to power on your GPIB-232/485CT-A. If you have a DC version, use the power switch on the rear panel to power on your GPIB-232/485CT-A.

   The **POWER** LED indicator should come on immediately. The **READY** LED indicator should come on after the GPIB-232/485CT-A has passed its power-on self test, indicating the unit is ready for operation. If the **READY** LED does not come on within seven seconds after the unit is powered on, recheck all connections and switch settings and retry the power-on sequence. If the **READY** LED still does not come on, contact National Instruments.

Configure the Hardware

If you want to change the settings of the GPIB-232/485CT-A, power off your system and follow the instructions in the next sections.

Changing the S Mode Characteristics

You can use the DIP switch on the rear panel to configure the serial port characteristics of the GPIB-232/485CT-A in S mode. When switch 8 is set to S mode, switches 1 through 3 set the baud rate, and switches 4 through 7 set the data formatting characteristics. Figure 3-1 shows the DIP switch. The unmarked DIP switches on the rear panel are reserved for future development and should remain in the OFF position.

Note The numbers 1–8 do not appear on the box. They are included in these illustrations as a reference aid.

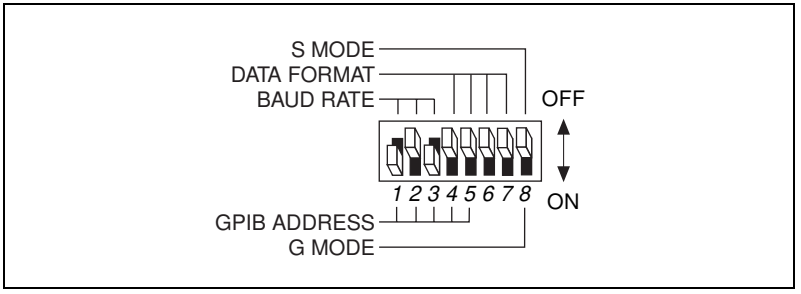


Figure 3-1. Default Setting (S Mode) for DIP Switch

In Figure 3-1, switch 8 is set to S mode, so the labels on top of the switch apply. Switches 1 through 3 are ON, OFF, and ON, respectively, indicating that the serial port is operating at 9600 baud. Switches 4 and 5 are both OFF, which indicates that parity is disabled. Switch 6 is OFF, indicating 1 stop bit/character. Switch 7 is OFF, indicating that the GPIB-232/485CT-A is using 7 bits per character for serial data transfers.

Tables 3-2 and 3-3 show the possible configurations for the baud rate and data format switches when you are using S mode and what each configuration indicates. Default settings are in **bold**.

Table 3-2. S Mode Switch Settings for Serial Port Baud Rate

Switches			Indication
1	**2**	**3**	
OFF	OFF	OFF	300 baud
ON	OFF	OFF	600 baud
OFF	ON	OFF	1200 baud
ON	ON	OFF	2400 baud
OFF	OFF	ON	4800 baud
ON	**OFF**	**ON**	9600 baud
OFF	ON	ON	19200 baud
ON	ON	ON	38400 baud

Table 3-3. S Mode Switch Settings for Data Formatting Characteristics

Switch	Position	Indication
4	**OFF**	odd parity
	ON	even parity
5	**OFF**	parity generation/checking disabled
	ON	parity generation/checking enabled
6	**OFF**	1 stop bit/character
	ON	2 stop bits/character
7	**OFF**	7 bits/character
	ON	8 bits/character
8	**OFF**	operates in S mode
	ON	operates in G mode

Sample Switch Settings for S Mode

To operate in S mode, set switch 8 to OFF. Set the remaining switches to match the characteristics of the terminal or computer you attach to the other end of the serial cable.

Often, you can change the serial port characteristics of the terminal or computer by setting switches or running a utility program, or from within a programming environment. Determine the default characteristics of your computer or terminal's serial port. If you want to change the configuration on that side, do so before attempting to communicate with the GPIB-232/485CT-A. Then set the configuration switch on the GPIB-232/485CT-A to match your serial port characteristics.

IBM PC or Compatibles

If your computer is an IBM PC or compatible, the serial port default characteristics on the PC are as follows:

Baud rate 1200

Parity Even

Data bits 7

Stop bits 1

If these defaults meet the needs of your application, set the GPIB-232/485CT-A switches as shown in Figure 3-2.

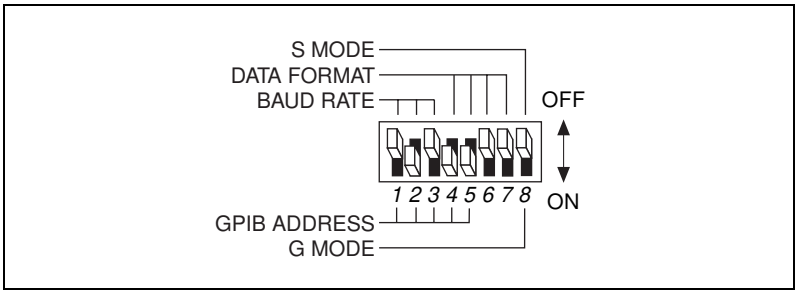


Figure 3-2. Switch Settings to Match IBM PC Defaults

In many cases, you might want to change the default characteristics of the serial port on the IBM PC. You might want to run at a higher baud rate and you might want to send 8-bit data bytes for binary data that are sent to the GPIB device. To change the IBM PC's serial port characteristics to 9600 baud and 8 data bits from within Quick BASIC, place the following statement at the beginning of your application program:

```
OPEN "COM1:9600,,8," AS #1
```

Then set the switches on the GPIB-232/485CT-A as shown in Figure 3-3.

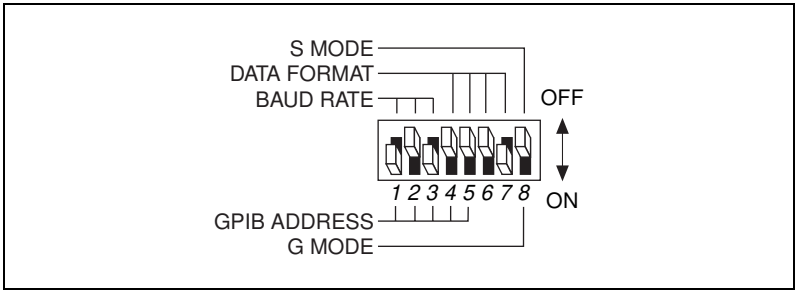


Figure 3-3. Sample Switch Settings with an IBM PC or Compatible

Other Systems

If your computer (or terminal) is other than those described in this chapter, refer to the manual that came with your equipment to learn the default settings of the serial port and how to change them.

Whatever serial port characteristics you decide to use, you must set up both your serial device and your GPIB-232/485CT-A to identical characteristics.

Changing the G Mode Characteristics

You can use the DIP switch on the rear panel to configure the G mode settings. When switch 8 is set to G mode, switches 1 through 5 set the GPIB addresses for the GPIB-232/485CT-A and the serial device connected to it. Figure 3-4 shows a sample G mode setting. The unmarked DIP switches on the rear panel are reserved for future development and should remain in the OFF position.

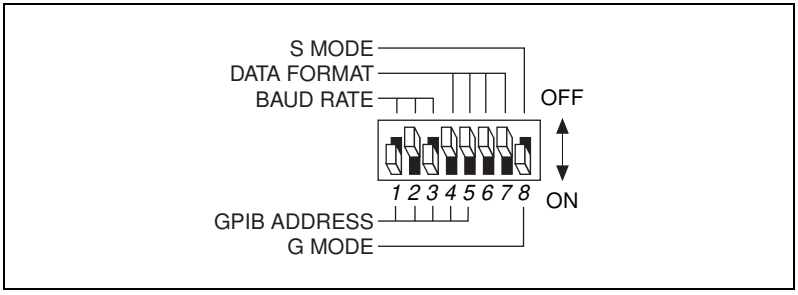


Figure 3-4. Sample G Mode Switch Setting

In Figure 3-4, switches 1 through 5 are ON, OFF, ON, OFF, and OFF, respectively, indicating that the GPIB-232/485CT-A is at GPIB address 5 and the serial device is at GPIB address 6. Switches 6, 7, and 8 are OFF, OFF, and ON, respectively, indicating that the GPIB-232/485CT-A is operating in G mode. Switches 6 and 7 must remain in the OFF position while in G mode.

Choosing GPIB Addresses for G Mode

When you use the GPIB-232/485CT-A in G mode, switches 1 through 5 set the primary GPIB address of the GPIB-232/485CT-A. The primary address plus one is the GPIB address of the serial device connected to the GPIB-232/485CT-A. Before you set the address switches, find two consecutive addresses that are not used by any other GPIB devices in your system. Refer to *Addressing the GPIB-232/485CT-A and Serial Device* in Chapter 6, *Programming in G Mode*, for more information. Table 3-4

shows the switch settings for the first (primary) GPIB address and the corresponding serial device address.

Table 3-4. GPIB Address Switch Settings for G Mode

Switches					GPIB-232/485CT-A Address	Serial Device Address
1	2	3	4	5		
OFF	OFF	OFF	OFF	OFF	0	1
ON	OFF	OFF	OFF	OFF	1	2
OFF	ON	OFF	OFF	OFF	2	3
ON	ON	OFF	OFF	OFF	3	4
OFF	OFF	ON	OFF	OFF	4	5
ON	OFF	ON	OFF	OFF	5	6
OFF	ON	ON	OFF	OFF	6	7
ON	ON	ON	OFF	OFF	7	8
OFF	OFF	OFF	ON	OFF	8	9
ON	OFF	OFF	ON	OFF	9	10
OFF	ON	OFF	ON	OFF	10	11
ON	ON	OFF	ON	OFF	11	12
OFF	OFF	ON	ON	OFF	12	13
ON	OFF	ON	ON	OFF	13	14
OFF	ON	ON	ON	OFF	14	15
ON	ON	ON	ON	OFF	15	16
OFF	OFF	OFF	OFF	ON	16	17
ON	OFF	OFF	OFF	ON	17	18
OFF	ON	OFF	OFF	ON	18	19
ON	ON	OFF	OFF	ON	19	20
OFF	OFF	ON	OFF	ON	20	21
ON	OFF	ON	OFF	ON	21	22
OFF	ON	ON	OFF	ON	22	23

Table 3-4. GPIB Address Switch Settings for G Mode (Continued)

Switches					GPIB-232/485CT-A Address	Serial Device Address
1	2	3	4	5		
ON	ON	ON	OFF	ON	23	24
OFF	OFF	OFF	ON	ON	24	25
ON	OFF	OFF	ON	ON	25	26
OFF	ON	OFF	ON	ON	26	27
ON	ON	OFF	ON	ON	27	28
OFF	OFF	ON	ON	ON	28	29
ON	OFF	ON	ON	ON	29	30
OFF	ON	ON	ON	ON	30	0

4

Programming in S Mode

This chapter describes how to program the GPIB-232/485CT-A in S mode. It explains status information and error handling, programming considerations, programming messages, function arguments, GPIB termination methods, function names, S mode default settings, and the S mode functions.

Choosing Your S Mode Programming Method

When using the GPIB-232/485CT-A in S mode, you can either use the S mode functions listed in the manual, or you can use the high-level NI-488.2 software for the GPIB-232/485CT-A. If you want to use the functions listed in this manual, refer to this chapter and Chapter 5, *S Mode Functions*, for programming information. If you want to use the NI-488.2 software, refer to the NI-488.2 user manual and NI-488.2 function reference manual. Contact National Instruments for ordering information if you do not have the NI-488.2 software package.

Status Information and Error Handling Characteristics

The function descriptions in Chapter 5, *S Mode Functions*, explain that the GPIB-232/485CT-A *records* specific status and error information. This means that it stores that information in its memory so that it is available when you request it.

The function descriptions also explain that the GPIB-232/485CT-A *returns* certain information to you. This means that the GPIB-232/485CT-A sends information to you over the serial port.

The GPIB-232/485CT-A continuously monitors the serial port for transmission errors. If it encounters an error in the serial data, the GPIB-232/485CT-A records the error. You can program the GPIB-232/485CT-A to ignore serial port errors using the `spign` function.

Programming Considerations

- The programming examples for each function description are in Microsoft QuickBASIC Version 4.5. Although the examples in this manual are written in BASIC, you can program the GPIB-232/485CT-A using any programming language that has access to a serial port.
- In the function syntax descriptions, arguments enclosed in square brackets (`[]`) are optional. Do not enter the brackets as part of your argument.
- For all programming examples, the communications port has been assigned to file number 1 (`#1`) by the BASIC `OPEN "COM..."` statement.
- The I/O and high-level bus management functions are the most frequently used and should meet most of your needs. In the descriptions that follow, these functions are marked with an asterisk (*).
- You can use function name abbreviations, which include only as many characters as necessary to distinguish them from other functions. The abbreviated forms are indicated by **bold** text in the syntax description of each function.

Programming Messages

You can program the GPIB-232/485CT-A by sending programming messages (ASCII strings) and data strings to its serial port.

Programming Message Format

Each programming message is terminated with a carriage return (<CR>), a linefeed (<LF>), or a carriage return followed by a linefeed (<CR><LF>). Message termination is indicated by a <CR> in the syntax portions of the function descriptions. In the programming examples, the BASIC `PRINT #` statement automatically sends a carriage return at the end of the string, so a carriage return is not needed as part of the code.

You can enter programming messages in any combination of uppercase and lowercase letters.

Programming Message Example 1

The following line of code is an example programming message in BASIC.

```
PRINT #1,"clr 3,4"
```

This programming message contains the function name `clr` and the arguments `3` and `4`. It tells the GPIB-232/485CT-A to clear the devices at GPIB addresses 3 and 4. `PRINT #1` is the BASIC command to send characters to the serial port after the serial port has been opened with the `"OPEN COM..."` statement. In this example, BASIC automatically sends a <CR>, so it is not necessary to include it in the code.

Programming Message Example 2

To send more than one programming message with one `PRINT` statement, you can embed a <CR> (denoted by `CHR$(13)`) or a <LF> (denoted by `CHR$(10)`) in the statement. For example, to send the two programming messages "send interface clear" (`sic`) and "send remote enable" (`sre`), you could use either of these two sequences:

```
PRINT #1,"sic"
PRINT #1,"sre 1"
```

or

```
PRINT #1,"sic"+CHR$(13)+"sre 1"
```

Programming Message Example with Data String

The following line of BASIC code is an example of a programming message with a data string.

```
PRINT #1, "wrt 2"
PRINT #1, "IN;CI;"
```

This programming message contains the function name `wrt`, the argument 2, and the data string `"IN;CI;"`. It tells the GPIB-232/485CT-A to write to the device at primary address 2. `"IN;CI;"` is the data string that contains the data `wrt` sends out on the GPIB. In this case, a <CR> is automatically sent by BASIC following each print string, so it is not necessary to include it in the code.

Both the `cmd` and `wrt` programming messages are followed by a data string that can contain 7- or 8-bit data.

How Messages are Processed

The GPIB-232/485CT-A processes a programming message on a line-by-line basis. When the GPIB-232/485CT-A receives a message, it buffers the entire message, interprets the function name and arguments, then executes the message. The data portions of the `wrt` and `cmd` functions are processed differently. The data immediately following a `wrt` and a `cmd` function are sent directly to the GPIB.

The GPIB-232/485CT-A recognizes <CTRL-H> (hex 8) in a programming message as a backspace and erases the previous character. The GPIB-232/485CT-A recognizes <CTRL-H> in a data string as a data byte and does not erase the previous character.

Function Arguments

When specifying a function, separate the first argument from the function name with at least one space. Separate each additional argument with at least one space or a comma.

In the syntax portions of the function descriptions in Chapter 5, *S Mode Functions*, the information within the square brackets (`[]`) is optional. If you want to include optional information, do not include the brackets as part of your argument.

Abbreviations for Arguments

The function descriptions in Chapter 5, *S Mode Functions*, use abbreviations for some arguments. The abbreviations are as follows:

`addr`	a GPIB address
`alist`	one or more `addrs`
`bool`	a boolean value (1 = true, on, or enable or 0 = false, off, or disable)

GPIB Address

One argument used with most functions is the GPIB address. Each device on the GPIB has a GPIB address. The GPIB-232/485CT-A address is 0 at power on, but you can change it using the `caddr` function. Refer to the manuals that came with your GPIB devices to learn their addresses. You should know the addresses when you program the GPIB-232/485CT-A.

Only the lower five bits of each GPIB address are significant. These bits can range from 0 through 30 for both the primary and the secondary address.

For example, the binary value 01100010 (decimal 98) is interpreted as decimal 2.

Each of the following GPIB addresses specifies a primary address of 0 and a secondary address of 2. A plus sign (+) separates the primary address from the secondary address.

`0+2` or `0+98` or `32+98` or `0+\x62`

Lists of GPIB Addresses

When a function requires a list of one or more GPIB addresses, the maximum number of addresses that you can specify is 14. If you specify more than 14 addresses, the GPIB-232/485CT-A records the EARG error. This limit exists because the IEEE 488 specification allows no more than 15 GPIB devices to coexist on any GPIB.

Numeric String Arguments

Another type of argument is a numeric string. A numeric string represents an integer which you can express using decimal, octal, or hexadecimal digits. To specify an octal integer, precede the numeric string with a backslash (`\`). To specify a hexadecimal integer, precede the numeric string with a backslash X (`\x`).

Each of the following numeric strings represents the decimal value 112:

decimal	octal	hexadecimal
`112`	`\160`	`\x70`

GPIB Read and Write Termination Methods (END and EOS)

The IEEE 488 specification defines two ways that GPIB Talkers and Listeners can identify the last byte of data messages: END and EOS. The two methods permit a Talker to send data messages of any length without the Listener(s) knowing the number of transmission bytes in advance. END and EOS can be used individually or in combination, but the Listener must be configured to detect the end of a transmission.

END message — The Talker asserts the **EOI*** (End Or Identify) signal while the last data byte is being transmitted. The Listener stops reading when it detects a data

byte accompanied by **EOI***, regardless of the value of the byte.

EOS character — The Talker sends an EOS (end-of-string) character at the end of its data string. The Listener stops receiving data when it detects the EOS character. Either a 7-bit ASCII character or a full 8-bit binary byte can be used.

The GPIB-232/485CT-A always terminates GPIB `rd` operations on the END message. You can use the `eos` and `eot` functions to change GPIB read and write termination methods.

Function Names

Each function name has been selected to indicate the function's purpose and make your programs easier to understand. However, if you want to reduce some overhead in your program, you can use an abbreviation of the name that includes only as much of the function name as is necessary to distinguish it from other functions. This abbreviated form of the function name is shown in **bold** text in the function tables and in the syntax portions of the function descriptions.

S Mode Default Settings and Related Functions

Tables 4-1 and 4-2 list power-on characteristics of the GPIB-232/485CT-A in S mode and the functions you can use to change those characteristics.

Table 4-1. S Mode Serial Port Characteristics

Characteristic	Default Setting	Related Function
Echo bytes to serial port	no	**ec**ho
Ignore serial port errors	yes	**spi**gn
Send XON/XOFF	no	**x**on
Recognize XON/XOFF	no	**x**on

Table 4-2. GPIB Characteristics in S Mode

Characteristic	Default Setting	Related Function
Primary/secondary address	pad = 0, sad = none	**cad**dr
IEEE 488 parallel poll subset	PP1 (remote)	**co**nf
End-of-string modes	none	**eos**
Send END on writes	yes	**eot**
IST bit setting	0	**is**t
GPIB-232/485CT-A is System Controller	yes	**rsc**
I/O timeout	10 s	**tm**o
Serial poll timeout	0.1 s	**tm**o

List of S Mode Functions by Group

The GPIB-232/485CT-A S mode functions are divided into three main groups: GPIB functions, serial port functions, and general use functions. For more information about the S mode functions, refer to the alphabetical list of functions at the end of this chapter, or to Chapter 5, *S Mode Functions*.

GPIB Functions

The GPIB functions manage the GPIB port of the GPIB-232/485CT-A. The GPIB function subgroups are listed with the most frequently used groups first. Often, the I/O and high-level bus management functions are the only ones you need.

I/O Functions—read data from and write to GPIB devices.

- **rd**
- **wr**t

High-Level Bus Management Functions—send frequently used bus management instructions to GPIB devices.

- **cl**r
- **l**oc
- **tr**g

GPIB Initialization Functions—initialize various configurations of the GPIB port.

- **cad**dr
- **eos**
- **eot**
- **o**nl
- **rsc**
- **tm**o

Serial Poll Functions—conduct and respond to GPIB serial polls.

- **rsp**
- **rsv**

Low-Level Bus Management Functions—give you precise control over the GPIB. Use them when the I/O and high-level bus management functions do not meet the needs of your GPIB device.

- **ca**c
- **cm**d
- **g**ts
- **li**nes
- **ln**
- **pc**t
- **si**c
- **sre**

Parallel Poll Functions—conduct and respond to GPIB parallel polls.

- **is**t
- **ppc**
- **ppu**
- **rp**p

Serial Port Functions

The serial port functions initialize and manage the serial port of the GPIB-232/485CT-A.

- **ec**ho
- **spi**gn
- **x**on

General Use Functions

The general use functions are used for general operations that are not provided by the GPIB functions or serial port functions.

- **co**nf
- **id**
- **st**at
- **wa**it

Alphabetical List of S Mode Functions

Table 4-3 lists all of the S mode functions in alphabetical order.

Table 4-3. Alphabetical List of S Mode Functions

Function	Purpose
cac mode	Become Active Controller
caddr address	Change the GPIB address of the GPIB-232/485CT-A
clr address list	Clear specified device(s)
cmd count commands	Send GPIB commands
conf option value	Read/change configuration
echo on/off	Echo characters received from serial port
eos modes, eoschar	Change or disable GPIB end-of-string termination mode
eot on/off	Enable or disable END termination message on GPIB write operations
gts mode	Go from Active Controller to Standby

Table 4-3. Alphabetical List of S Mode Functions (Continued)

Function	Purpose
id	Identify system
ist set/clear	Set or clear individual status bit for use in GPIB-232/485CT-A response to parallel polls
lines	Determine state of GPIB control lines
ln address list	Check for listening devices
loc address list	Go to Local
onl on/off	Place the GPIB-232/485CT-A online/offline
pct address	Pass Control
ppc values	Parallel Poll Configure
ppu address list	Parallel Poll Unconfigure
rd count, address	Read data
rpp	Conduct (request) a Parallel Poll
rsc on/off	Request System Control
rsp address list	Conduct (request) a serial poll
rsv status byte	Request service and/or set or change the serial poll status byte
sic time	Send Interface Clear
spign on/off	Ignore serial port errors
sre on/off	Set remote enable
stat modes	Return GPIB-232/485CT-A status
tmo values	Change or disable time limits
trg address list	Trigger selected device(s)
wait mask	Wait for selected event(s)
wrt count, address list, data	Write data
xon modes	Change serial port XON/XOFF protocol

For more detailed information on each function, refer to Chapter 5, *S Mode Functions*.

5

S Mode Functions

This chapter contains descriptions of the S mode functions you can use to program the GPIB-232/485CT-A. These functions are in alphabetical order for easy reference.

The I/O and high-level bus management functions are the most frequently used and should meet most of your needs. In the descriptions that follow, these functions are marked with an asterisk (*).

For general information about using S mode functions, refer to Chapter 4, *Programming in S Mode*.

cac

Become Active Controller

Type

Low-level bus management function

Syntax

cac [bool]<CR>

Purpose

You can use cac to change the GPIB-232/485CT-A from Standby Controller to Active Controller when the I/O and high-level bus management functions do not meet the needs of your device. cac gives you more precise control over the GPIB than the I/O and high-level bus management functions.

Remarks

If the argument bool is 0, the GPIB-232/485CT-A takes control immediately—that is, it takes control asynchronously. If the argument bool is 1, the GPIB-232/485CT-A takes control after any handshake that is in progress completes–that is, it takes control synchronously.

If you call cac without an argument, the GPIB-232/485CT-A returns the current Controller status. The Controller status is 0 if the GPIB-232/485CT-A is not the Active Controller and 1 if the GPIB-232/485CT-A is the Active Controller.

The GPIB-232/485CT-A must be CIC when you call cac with an argument. If you call cac with an argument and the GPIB-232/485CT-A is not CIC, the GPIB-232/485CT-A records the ECIC error.

The power-on Controller status of the GPIB-232/485CT-A is Idle Controller.

See Also

gts and sic.

Example 1

```
PRINT #1,"cac 0"          'Take control immediately.
```

Example 2

```
PRINT #1,"cac 1"          'Take control synchronously.
```

Example 3

```
PRINT #1,"CAC"              'Are we the Active Controller?
response: 1<CR><LF> (...yes, we are CAC)
```

caddr

Change the GPIB Address of GPIB-232/485CT-A

Type

Initialization function

Syntax

caddr [addr]<CR>

Purpose

You can use caddr at the beginning of your program to change the GPIB address of the GPIB-232/485CT-A.

Remarks

The argument addr is a device address that specifies the new GPIB address for the GPIB-232/485CT-A. addr consists of a primary address and an optional secondary address. The secondary address is separated from the primary address by a plus sign (+). Both addresses are expressed as numeric strings.

If you specify a primary address without a secondary address, secondary addressing is disabled.

If you call caddr without an argument, the GPIB-232/485CT-A returns its current GPIB address.

The address assigned by this function remains in effect until you call caddr again, call onl, or turn off the GPIB-232/485CT-A.

The power-on default is 0 with secondary addressing disabled.

Example 1

```
PRINT #1,"caddr 0+22"      'Give GPIB-232/485CT-A a primary
                           'address of 0 and a secondary
                           'address of 22.
```

Example 2

```
PRINT #1,"CADDR 1"         'Change GPIB-232/485CT-A primary
                           'address to 1 and disable secondary
                           'addressing.
```

Example 3

```
PRINT #1,"CADDR"           'Return current GPIB-232/485CT-A address.
response: 1<CR><LF> (current GPIB-232/485CT-A address is 1)
```

clr

Clear Specified Device*

Type

High-level bus management function

Syntax

clr [alist]<CR>

Purpose

You can use clr to reset the internal or device functions of the specified GPIB devices. For example, a multimeter might require that you send it either the GPIB Device Clear or Selected Device Clear command to change its function, range, and trigger mode back to its default setting.

Remarks

The argument alist is a list of addrs separated by commas or spaces. addrs specify the GPIB addresses you want to clear.

If you call clr with alist, the GPIB-232/485CT-A clears only the devices specified in alist (Selected Device Clear). If you call clr without alist, the GPIB-232/485CT-A clears all devices (Device Clear).

If clr is the first function you call that requires gpib Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is recorded.

See Also

Appendix F, *GPIB Basics*, for more information about clearing devices, and Appendix C, *Status and Error Message Information*, for more error information.

Example 1

```
PRINT #1,"clr 14+30,16+12,18,3+26,6"
                            'Selectively clear 5 devices.
```

Example 2

```
PRINT #1,"CLR"              'Issue Device Clear to all devices.
```

cmd

Send GPIB Commands

Type

Low-level bus management function

Syntax

cmd [#count]<LF>
commands<CR>

Purpose

You can use cmd when the I/O and high-level bus management functions do not meet the needs of your device. cmd gives you precise control over the GPIB. For example, in applications that require command sequences not sent by other functions, you can use cmd to transmit any sequence of interface messages (commands) over the GPIB.

Remarks

The argument commands is a list of GPIB commands. These commands are represented by their ASCII character equivalents. For example, the GPIB Untalk (UNT) command is the ASCII character underscore (_). The GPIB commands, or interface messages, are listed in Appendix B, *Multiline Interface Messages*. They include device talk and listen addresses, secondary addresses, device clear and trigger messages, and other management messages.

The argument count is a numeric string preceded by a number sign (#). count specifies the number of GPIB command bytes (interface messages) to send, and can range from 1 to 4294967295. count must not include the <LF> or <CR> characters, which are used to terminate the programming message or string of GPIB command bytes.

If you call cmd without count, the GPIB-232/485CT-A recognizes the end of the command string when it sees a <CR> or an <LF>. Consequently, count is required only if the command string contains a <CR> or <LF> character.

Do not terminate the cmd programming message with the <CR> character alone. Use either <LF> or <CR><LF> to terminate the programming message. If you use <CR> alone and the first character of the command string is <LF>, the GPIB-232/485CT-A discards the <LF> as the second character of a <CR><LF> termination.

Do not use cmd to send programming instructions to devices. Use rd and wrt to send or receive device programming instructions and other device-dependent information.

If the GPIB-232/485CT-A is CIC but not Active Controller, it takes control and asserts **ATN*** before sending the command bytes. It remains Active Controller afterward.

The cmd operation terminates when:

- The GPIB-232/485CT-A successfully transfers all commands.
- The GPIB-232/485CT-A is not Controller-In-Charge. The ECIC error is recorded.
- The I/O time limit is exceeded. The EABO error is recorded.
- There is no device on the bus receiving the command bytes. The ENOL error is recorded.
- A serial port error occurs and is not ignored (see spign) .

After cmd terminates, the GPIB-232/485CT-A records the number of command bytes it actually sent. If one of the errors described above occurs, the count might be less than expected.

If an error occurs and the GPIB-232/485CT-A is unable to transmit the entire command string, the GPIB-232/485CT-A reads in and discards the remaining bytes of the command string.

See Also

spign and Appendix B, *Multiline Interface Messages*.

Example 1

```
PRINT #1,"CMD"+CHR$(10)+"+@"
                          'Program device at address 11 to listen
                          'and GPIB-232/485CT-A at address 0 to
                          'talk. Terminate the programming message
                          'with <LF> (ASCII 10). The device listen
                          'address is 43 (ASCII +) and GPIB-
                          '232/485CT-A talk address is 64 (ASCII @).
                          'The <CR> sent automatically at the end
                          'of the PRINT# statement is used to
                          'terminate the command string.
PRINT #1,"WRT"+CHR$(10)+"ABCDE"
                          'Write the string "ABCDE" to the device at
                          'address 11.
```

Example 2

```
PRINT #1,"cmd #4"+CHR$(10)+"_?W"+CHR$(9)
                          'Pass control to device at GPIB address
                          '23. CHR$(9)=TCT command.
```

conf

Read/Change GPIB-232/485CT-A Configuration

Type

General use function

Syntax

conf option [value]<CR>

Purpose

You can use conf to read or change certain configuration parameters of the GPIB-232/485CT-A.

Remarks

The argument option indicates which configuration parameter to read or change. The argument value indicates the new setting of the configuration parameter indicated by option. The value argument is optional. If you do not specify value, the current setting for the specified option is returned.

When option is 0, the PP2 option is indicated. The PP2 option is used to indicate what type of parallel poll configurations the GPIB-232/485CT-A should accept. If the PP2 value is 0, the GPIB circuitry of the GPIB-232/485CT-A uses the IEEE 488 Parallel Poll (PP) interface function subset PP1 (remote configuration from an external Controller). If the PP2 value is 1, the GPIB-232/485CT-A uses PP subset PP2 (local configuration via the ppc and ppu functions). When PP subset PP2 is used, the GPIB-232/485CT-A ignores remote parallel poll configurations. The default value of the PP2 option is 0.

The values assigned by this function remain in effect until you call conf again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

ppc, ppu, and Appendix H, *Parallel Polling*.

The following examples show commands as you would enter them at a terminal.

Example 1

```
PRINT #1 "conf 0 1"        'Enable the PP2 option.
```

Example 2

```
PRINT #1 "conf 0"          'Return the current setting for
                           'the PP2 option.
```

response: 1<CR><LF> (PP2 mode selected)

echo

Echo Characters Received from Serial Port

Type

Serial port function

Syntax

echo [bool]<CR>

Purpose

You can use echo when a terminal is connected to the GPIB-232/485CT-A and you want everything you type to display on the terminal screen.

Remarks

If the argument bool is 1, characters received from the serial port are echoed back to the serial port. If the argument bool is 0, characters are not echoed. If the argument bool is 1 and echoing was previously disabled, characters are not echoed until this command has been completely processed–that is, the *next* programming message is echoed.

If you call echo without an argument, the GPIB-232/485CT-A returns the current setting.

The assignment made by echo remains in effect until you call echo again, call onl, or you turn off the GPIB-232/485CT-A.

The following examples show commands as you would enter them at a terminal.

Example 1

```
echo 1<CR>                  'Turn on character echoing.
```

Example 2

```
ECHO 0<CR>                  'Disable character echoing.
```

Example 3

```
echo<CR>                    'What is the current echo status?
response: 0<CR><LF> (character echo is disabled)
```

eos

Change/Disable GPIB EOS Termination Mode

Type

Initialization function

Syntax

eos [R[B] eoschar]<CR>
or
eos X[B] eoschar<CR>
or
eos D <CR>

Purpose

You can use eos at the beginning of your program if you want to use an eos mode when you transfer data to and from the GPIB. eos tells the GPIB-232/485CT-A when to stop reading information from the GPIB. eos also enables the GPIB-232/485CT-A to tell other devices that it is finished writing information to the GPIB. eos defines a specific character, end-of-string (EOS), to be recognized as a string terminator.

Remarks

The arguments R, X, B, and D specify GPIB termination methods. They enable or disable the corresponding eos mode. If a particular letter is specified, the corresponding eos mode is enabled. If it is not specified, the corresponding eos mode is disabled. By default, all eos modes are disabled. Table 5-1 lists the termination methods and corresponding letters.

Table 5-1. Data Transfer Termination Methods in S Mode

Description	Letter
REOS—terminate read when EOS is detected	R
XEOS—set **EOI*** with EOS on write functions	X
BIN—compare all 8 bits of EOS byte rather than low 7 bits (all read and write functions)	B
DISABLE—disable all eos modes	D

Methods R and B determine how GPIB read operations (rd) performed by the GPIB-232/485CT-A terminate. If method R alone is chosen, reads terminate when the low seven bits of the byte that is read match the low seven bits of the EOS character. If both methods R and B are chosen, a full 8-bit comparison is used.

Methods X and B together determine when GPIB write operations (wrt) performed by the GPIB-232/485CT-A send the END message. If method X alone is chosen, the END message is sent automatically with the EOS byte when the low seven bits of that byte match the low seven bits of the EOS character. If methods X and B are chosen, a full 8-bit comparison is used.

eoschar is a numeric string that represents a single ASCII character. For example, decimal 10 represents the ASCII linefeed character. Refer to Appendix B, *Multiline Interface Messages*, for more ASCII codes.

If you call eos without an argument, the GPIB-232/485CT-A returns the current eos settings.

If you call eos with B alone as an argument, the GPIB-232/485CT-A records the EARG error.

Note Defining an EOS byte for the GPIB-232/485CT-A does not cause the GPIB-232/485CT-A to insert that byte into the data string when performing GPIB writes. To send the EOS byte, you must include it in the data string that you send following the wrt programming message.

The assignment made by this function remains in effect until you call eos again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

The *GPIB Read and Write Termination Methods (END and EOS)* section in Chapter 4, *Programming in S Mode*.

Example 1

```
PRINT #1,"eos R,B,10"      'Terminate read when <LF> is detected;
                           'compare all 8 bits; do not send EOI*
                           'with <LF>.
PRINT #1,"rd #10 5"        'Read 10 bytes from device 5 into serial
                           'port buffer.
RESP$=INPUT$(10,#1)        'Input 10 bytes from serial port buffer.
LINE INPUT #1,COUNT$       'Input string that indicates number of
                           'bytes actually read from GPIB.
PRINT COUNT$;" bytes were read from GPIB"
                           'Print number of bytes that were read
                           'from the GPIB.
```

Example 2

```
PRINT #1,"EOS X,13"          'Send EOI* with <CR> on wrt; do not
                             'terminate when <CR> is detected
                             'on rd; compare 7 bits.
PRINT #1,"wrt #10 5"+CHR$(10)+"012345678"
                             'GPIB-232/485CT-A sends EOI* with <CR>
                             '(CHR$(13)) to tell Listeners that this
                             'is the last byte of data.
```

Example 3

```
PRINT #1,"eos"               'What are the current EOS settings?
```

response: X,13<CR><LF> (Send EOI* with <CR>)

eot

Enable/Disable END Message on GPIB Writes

Type

Initialization function

Syntax

eot [bool]<CR>

Purpose

You can use eot at the beginning of your program if you want to change how the GPIB-232/485CT-A terminates GPIB writes. Using eot, you tell the GPIB-232/485CT-A to automatically send or not send the GPIB END message with the last byte that it writes to the GPIB.

Remarks

If the argument bool is 1, the GPIB-232/485CT-A automatically sends the END message with the last byte of each wrt. If the argument bool is 0, END is not sent. The power-on default is 1.

The GPIB-232/485CT-A sends the END message by asserting the GPIB **EOI*** signal during the last byte of a data transfer. eot is used primarily to send variable length data.

If you call eot without an argument, the GPIB-232/485CT-A returns a 1 to indicate END termination is currently enabled, or a 0 to indicate END termination is currently disabled.

The assignment made by eot remains in effect until you call eot again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

The *GPIB Read and Write Termination Methods (END and EOS)* section in Chapter 4, *Programming in S Mode*.

Example 1

```
PRINT #1,"eot 0"          'Disable END termination.
```

Example 2

```
PRINT #1,"EOT 1"           'Send END with last byte.
PRINT #1,"WRT 3"+CHR$(10)+ABCDE"
                           'Write data to device at address 3.
                           'The EOI* line is automatically
                           'asserted when the last byte (the letter
                           'E) is sent to tell the Listeners it is
                           'the last byte of data.
```

Example 3

```
PRINT #1,"eot"             'What is the current EOT setting?
```

response: 1<CR><LF> (END termination is currently enabled)

gts

Go from Active Controller to Standby

Type

Low-level bus management function

Syntax

gts [bool]<CR>

Purpose

You can use gts to change the GPIB-232/485CT-A from Active Controller to Standby Controller. Use gts when the I/O and high-level bus management functions do not meet the needs of your device.

gts permits GPIB devices to transfer data without the GPIB-232/485CT-A participating in the transfer. For example, you can use gts if you want to let two external devices talk to each other directly. The GPIB-232/485CT-A can selectively participate in the handshake of the data transfer and hold off the handshake when it detects the END message. The GPIB-232/485CT-A can then take control synchronously without possibly corrupting the transfer.

Remarks

If the argument bool is 1, shadow handshaking is enabled. If the argument bool is 0, shadow handshaking is not performed.

gts causes the GPIB-232/485CT-A to go to the Controller Standby state and to unassert the **ATN*** signal if it is initially the Active Controller.

If you enable shadow handshaking, the GPIB-232/485CT-A participates in the data handshake as an Acceptor without actually reading the data. It monitors the transfers for the END (**EOI*** or end-of-string character) message and holds off subsequent transfers. This mechanism allows the GPIB-232/485CT-A to take control synchronously on a subsequent operation such as cmd or rpp.

Before performing a gts with a shadow handshake, you should call eos to establish the proper end-of-string character or to disable the EOS detection if the end-of-string character used by the Talker is not known.

If you call `gts` without an argument, the GPIB-232/485CT-A returns the current Controller status, as follows:

- `CSB,0` if the GPIB-232/485CT-A is in Standby without shadow handshaking
- `CSB,1` if the GPIB-232/485CT-A is in Standby with shadow handshaking
- `CAC` if the GPIB-232/485CT-A is CIC but is not in Standby—that is, it is the Active Controller
- `CIDLE` if the GPIB-232/485CT-A is not the CIC—that is, if it is an Idle Controller

The GPIB-232/485CT-A must be CIC when you call `gts` with an argument. If you call `gts` with an argument and the GPIB-232/485CT-A is not CIC, the GPIB-232/485CT-A records the ECIC error.

See Also

cac.

Example 1

```
PRINT #1,"gts 0"          'GTS without shadow handshaking.
```

Example 2

```
PRINT #1,"GTS 1"          'GTS with shadow handshaking.
```

Example 3

```
PRINT #1,"gts"            'What is standby status?
response: CSB,1<CR><LF>   (GPIB-232/485CT-A is in standby
                          status with shadow handshaking)
```

id

Identify System

Type

General use function

Syntax

id<CR>

Purpose

You can use `id` if you want to know the revision level of your software or how much RAM is installed in your GPIB-232/485CT-A.

Remarks

The identification is returned in three strings. The first string identifies the company product model and software revision level. The second string is a copyright notice. The third string identifies the number of bytes of RAM in the GPIB-232/485CT-A.

The following example shows the current identification string at the time of this printing. The general format will be as shown; however, version-specific information such as revision levels and copyright dates change as needed.

Example

```
PRINT #1,"id"              'Get system identification.

response:      GPIB-232/485CT-A Rev. B.3<CR><LF>
               (c)1995 National Instruments<CR><LF>
               256K bytes RAM<CR><LF>
```

ist

Set or Clear Individual Status Bit

Type

Parallel poll function

Syntax

ist [bool]<CR>

Purpose

You can use ist when the GPIB-232/485CT-A participates in a parallel poll that is conducted by another device that is Active Controller.

Remarks

If the argument bool is 1, the GPIB-232/485CT-A's individual status bit is set to 1. If the argument bool is 0, the GPIB-232/485CT-A's individual status bit is cleared. The power-on default is 0.

If you call ist without an argument, the GPIB-232/485CT-A returns the value of its individual status bit.

The assignment made by ist remains in effect until you call ist again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

ppc and Appendix H, *Parallel Polling*.

Example 1

```
PRINT #1,"ist 1"          'Set ist to 1.
```

Example 2

```
PRINT #1,"IST 0"          'Clear ist to 0.
```

Example 3

```
PRINT #1,"ist"            'What is ist set to?
```

response: 0<CR><LF> (ist is currently 0)

lines

Return State of GPIB Control Lines

Type

Low-level bus management function

Syntax

lines<CR>

Purpose

You can use `lines` to determine the state of the GPIB control lines.

Remarks

This command returns two numeric strings of information. The numeric strings are separated by a comma and terminated with <CR><LF>. The state of the eight GPIB control lines is returned in the first number. Each bit in this number corresponds to a GPIB control line as follows:

7	6	5	4	3	2	1	0
EOI*	**ATN***	**SRQ***	**REN***	**IFC***	**NRFD***	**NDAC***	**DAV***

The second number contains mask bits in the same order as above, indicating which lines are actually being reported, and which are indeterminable. If a particular mask bit is 1, then the corresponding bit in the first number indicates the state of the line. If the mask bit is 0, then the corresponding bit in the first number should be disregarded.

See Also

Appendix F, *GPIB Basics*.

Example

```
PRINT #1,"lines"            'Determine state of GPIB
                            'control lines.
response: 96,249<CR><LF>    (All control lines except NRFD*
                            and NDAC* can be determined. Of
                            those, only ATN* and SRQ* are
                            currently asserted.)
```

In

Check for Listening Devices

Type

Low-level bus management function

Syntax

ln alist<CR>

Purpose

You can use ln to determine whether or not there are listening devices at the specified GPIB addresses.

Remarks

The argument alist is a list of addrs which are separated by commas or spaces. There should be at least one addr in the list. addrs specify the GPIB addresses of the devices you want to check.

When specifying the addr parameters, a value of 255 (hex FF) can be used as a secondary address. When this special value is used, the GPIB-232/485CT-A checks all of the secondary addresses for the specified primary address.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

ln returns the list of addresses found to be listening on the GPIB bus. The addresses of this list are separated by commas and terminated with <CR><LF>. If none of the specified addresses were found to be listening, a <CR><LF> alone is returned.

Example

```
PRINT #1,"ln 2,5+\xFF,3+20,7"
                              'Look for listening devices.
response: 5+10,5+11,5+12,7<CR><LF> (Four listening devices
                                          found.)
```

loc

Go to Local*

Type

High-level bus management function

Syntax

`loc [alist]`<CR>

Purpose

You can use `loc` to put a GPIB device in local program mode. In this mode you can program the device from its front panel. Because a device must usually be placed in remote program mode before it can be programmed from the GPIB, the GPIB-232/485CT-A automatically puts the device in remote program mode. You then use `loc` to return devices to local program mode.

Remarks

The argument `alist` is a list of `addrs` separated by commas or spaces. `addrs` specify the GPIB addresses of the devices you want to return to local mode.

If you call `loc` with `alist`, the GPIB-232/485CT-A places the specified device(s) in local mode using the Go To Local (GTL) command.

If you call `loc` without `alist`, and the GPIB-232/485CT-A is System Controller, the GPIB-232/485CT-A returns all devices to local mode by unasserting **REN*** and asserting it again. If you call `loc` without `alist` and the GPIB-232/485CT-A is not System Controller, the GPIB-232/485CT-A records the ESAC error.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with `rsc`, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

If only the special value of 255 (hex FF) is entered as a parameter, the GPIB-232/485CT-A configures itself for local program mode by pulsing its internal rtl (return to local) message. The GPIB-232/485CT-A does not require GPIB Controller capability to configure itself for local program mode.

See Also

Appendix C, *Status and Error Message Information*.

Example 1

```
PRINT #1,"loc 6+22,4+23,7"
                              'Put 3 devices in local mode.
```

Example 2

```
PRINT #1,"LOC"                'Put all devices in local mode.
```

Example 3

```
PRINT #1,"LOC 255"            'Put the GPIB-232/485CT-A in local
                              'program mode.
```

onl

Place the GPIB-232/485CT-A Online/Offline

Type

Initialization function

Syntax

`onl [bool]`<CR>

Purpose

You can use `onl` to disable communications between the GPIB-232/485CT-A and the GPIB, or to reinitialize the GPIB-232/485CT-A characteristics to their default values.

Remarks

If the argument `bool` is 1, the GPIB-232/485CT-A places itself online. If the argument `bool` is 0, the GPIB-232/485CT-A takes itself offline. By default, the GPIB-232/485CT-A powers up online, is in the Idle Controller state, and configures itself to be the System Controller.

Placing the GPIB-232/485CT-A offline can be thought of as disconnecting its GPIB cable from the other GPIB devices.

Placing the GPIB-232/485CT-A online allows the GPIB-232/485CT-A to communicate over the GPIB and restores all settings to their power-on values.

If you call `onl` without an argument, the GPIB-232/485CT-A returns the current state, which is 0 if the GPIB-232/485CT-A is offline and 1 if the GPIB-232/485CT-A is online.

See Also

The *S Mode Default Settings and Related Functions* section in Chapter 4, *Programming in S Mode*.

Example 1

```
PRINT #1,"onl 1"          'Put the GPIB-232/485CT-A online and
                          'restore its power-on settings.
```

Example 2

```
PRINT #1,"ONL 0"          'Put the GPIB-232/485CT-A offline to
                          'prevent it from communicating with the
                          'GPIB.
```

pct

Pass Control

Type

Low-level bus management function

Syntax

pct addr<CR>

Purpose

You can use pct to pass Controller-In-Charge (CIC) authority from the GPIB-232/485CT-A to some other GPIB device.

Remarks

The argument addr is the address of the device you want to pass control to. addr consists of a primary address and an optional secondary address.

pct passes CIC authority from the GPIB-232/485CT-A to the device specified by addr. The GPIB-232/485CT-A automatically goes to Idle Controller State. It is assumed that the target device has Controller capability.

If you call pct with an argument and the GPIB-232/485CT-A is not CIC, it records the ECIC error.

If you call pct without an argument, the GPIB-232/485CT-A records the EARG error.

Example

```
PRINT #1,"pct 7+18"        'Pass control to device with primary
                           'address 7 and secondary address 18.
```

ppc

Parallel Poll Configure

Type

Parallel poll function

Syntax

ppc addr,ppr,s [addr,ppr,s]...<CR>

Purpose

You can use ppc to configure specified GPIB devices to respond to parallel polls in a certain manner.

Remarks

The argument addr specifies the GPIB address of the device to be enabled for parallel polls. addr consists of a primary address and an optional secondary address.

The argument ppr is an integer string between 1 and 8 which specifies the data line on which to respond.

The argument s is either 0 or 1 and is interpreted along with the value of the device's individual status bit to determine whether to drive the line true or false.

Each group of addr,ppr,s can be separated by either a comma or space, just as any list of arguments.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

The GPIB-232/485CT-A takes the arguments ppr and s and constructs the appropriate parallel poll enable (PPE) message for each addr specified.

If you call ppc without an argument, the GPIB-232/485CT-A records the EARG error.

If only one addr, ppr, s is specified, and addr has the special value of 255 (hex FF), the GPIB-232/485CT-A configures itself for parallel polls. To do this, the GPIB-232/485CT-A must be using IEEE 488 Parallel Poll (PP) interface function subset PP2. You can do this by setting the PP2 option using the conf function. If the PP2 option is not set, the

GPIB-232/485CT-A records the ECAP error. The GPIB-232/485CT-A does not require GPIB Controller capability to configure itself for parallel polls.

See Also

conf, ist, ppu, rpp, and Appendix H, *Parallel Polling*.

Example 1

```
PRINT #1,"PPC 18+23,8,0 23+10,7,1"
                          'Configure 2 devices for parallel poll.
PRINT #1,"RPP"            'Conduct a Parallel poll of 2 devices
                          'configured above.

response:192<CR><LF> (both devices responded positively)

INPUT #1,PPR%             'Assign parallel poll response to
                          'integer variable.
```

Example 2

```
PRINT #1,"PPC 255,7,1"
                          'Configure the GPIB-232/485CT-A to
                          'respond to parallel polls on data line
                          '7 if its individual status bit is 1.
```

ppu

Parallel Poll Unconfigure

Type

Parallel poll function

Syntax

ppu [alist]<CR>

Purpose

You can use ppu if you are performing parallel polls and you want to prevent certain GPIB devices from responding.

Remarks

The argument alist is a list of addrs that are separated by commas or spaces. addrs specify the GPIB addresses of the devices to be disabled from parallel polls.

If you call ppu with alist, the GPIB-232/485CT-A unconfigures from parallel polls only those devices specified in alist. If you call ppu without alist, the GPIB-232/485CT-A unconfigures all devices from parallel polls.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

If only addr is specified and it has the special value of 255 (hex FF), the GPIB-232/485CT-A disables itself from responding to parallel polls. To do this, the GPIB-232/485CT-A must be using IEEE 488 Parallel Poll (PP) interface function subset PP2. You can do this by setting the PP2 option using the conf function. If the PP2 option is not set, the GPIB-232/485CT-A records the ECAP error. The GPIB-232/485CT-A does not require GPIB Controller capability to disable itself from responding to parallel polls.

See Also

conf, ist, ppc, rpp, and Appendix H, *Parallel Polling*.

Example 1

```
PRINT #1,"ppu 14"          'Send the PPD command to device 14.
```

Example 2

```
PRINT #1,"PPU"              'Send the PPU command to all devices.
```

Example 3

```
PRINT #1,"PPU 255"          'Disable the GPIB-232/485CT-A from
                            'responding to parallel polls.
```

rd

Read Data*

Type

I/O function

Syntax

rd #count [addr]<CR>

Purpose

You can use rd to read data from the GPIB.

Remarks

The argument count is a numeric string preceded by a number sign (#). count specifies the number of GPIB data bytes to read, and can range from 1 to 4294967295.

The argument addr specifies the GPIB address of the device to be addressed as a Talker. addr consists of a primary address and an optional secondary address.

The GPIB-232/485CT-A reads data from the GPIB until one of the following events occurs:

- The GPIB-232/485CT-A successfully transfers all data.
- The GPIB END message is received with a data byte.
- The EOS byte is received.
- The I/O time limit is exceeded. The EABO error is recorded.
- The GPIB-232/485CT-A receives a Device Clear. The EABO error is recorded.
- The addr argument is specified and the requested GPIB addressing bytes cannot be sent. The EBUS error is recorded.

Because you might not know the number of bytes actually read from the GPIB, the GPIB-232/485CT-A returns the received GPIB data to you in the following manner. First, the GPIB-232/485CT-A returns all bytes it read from the GPIB. Next, if the number of bytes read is less than count, the GPIB-232/485CT-A sends null bytes (decimal 0) until the total number of bytes returned to you matches the number specified in count. Finally, it returns a numeric string representing the number of bytes that it actually read from the GPIB.

For example, if you send the GPIB-232/485CT-A the programming message "rd #10"<CR>, it reads data from the GPIB until it receives 10 bytes of data, the END message, or an eos byte. If the GPIB-232/485CT-A received END with the fourth data byte, it would return the 4 data bytes. Then it would send 6 null bytes followed by an ASCII 4 and <CR><LF>.

You should always read back `count` bytes of data from the serial port, then look at the numeric string to determine how many bytes were read from the GPIB.

If you call `rd` with the `addr` argument, the GPIB-232/485CT-A must be CIC to perform the addressing. If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with `rsc`, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you call `rd` with the `addr` argument, and you previously passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is recorded.

When performing the addressing for a specified `addr`, the GPIB-232/485CT-A sends out its own listen address as well as the talk address of the specified device. It then places itself in Standby Controller state with **ATN*** unasserted, and remains there after the read operation is complete.

If the `addr` argument is not specified, the GPIB-232/485CT-A assumes that it has already been addressed to listen by the Controller. If the GPIB-232/485CT-A is the Controller, and did not address itself to listen before calling `rd`, the EADR error is recorded and no data bytes are transferred.

If you call `rd` without an argument, the GPIB-232/485CT-A records EARG.

See Also

`eos`, `eot`, and `tmo`.

Example

```
PRINT #1,"rd #10 3"          'Read up to 10 bytes from the GPIB
                             'device at address 3.
RESP$=INPUT$(10,#1)          'Input 10 bytes from serial port buffer.
INPUT #1,COUNT%              'Input ASCII string representing number
                             'of bytes read from the GPIB. COUNT% is
                             'number of bytes read from GPIB;
                             'remaining bytes in RESP$ can be 'ignored.
```

rpp

Request (Conduct) a Parallel Poll

Type

Parallel poll function

Syntax

rpp<CR>

Purpose

You can use rpp if you want to conduct a parallel poll to obtain information from several GPIB devices at the same time.

Remarks

rpp causes the GPIB-232/485CT-A to conduct a parallel poll of previously configured devices by sending the **IDY*** (Identify) message (**ATN*** and **EOI*** both asserted) and reading the response from the GPIB data lines. The GPIB-232/485CT-A pulses the **IDY*** message for greater than or equal to 2 microseconds and expects valid responses within that time. It remains Active Controller after pulsing the **IDY*** message.

The GPIB-232/485CT-A returns the Parallel Poll Response (PPR) following the poll in the form of a numeric string representing the decimal value of the response.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

See Also

ist, ppc, ppu, and Appendix H, *Parallel Polling*.

Example

```
PRINT #1,"ppc 13,1,0 15,3,0"+CHR$(13)+"rpp"
                            'Configure 2 devices for parallel polls
                            'and poll them.
```

response: 5<CR><LF> (both devices responded positively)

rsc

Request System Control

Type

Initialization function

Syntax

rsc [bool]<CR>

Purpose

You can use rsc if you want to change which device in your GPIB system is System Controller.

Remarks

If the argument bool is 1, the GPIB-232/485CT-A configures itself to be the GPIB System Controller. If the argument bool is 0, the GPIB-232/485CT-A unconfigures itself to be the System Controller.

If you call rsc without an argument, the GPIB-232/485CT-A returns its System Controller status, which is 0 if the GPIB-232/485CT-A is not currently System Controller or 1 if the GPIB-232/485CT-A is System Controller.

As System Controller the GPIB-232/485CT-A can send the Interface Clear (**IFC***) and Remote Enable (**REN***) messages to GPIB devices. If some other Controller asserts **IFC***, the GPIB-232/485CT-A can only respond if it is not configured as System Controller.

In most applications, the GPIB-232/485CT-A is System Controller. In some applications, the GPIB-232/485CT-A is never System Controller. In either case, rsc is used only if the computer is not going to be System Controller while the program executes. The IEEE 488 standard does not specifically allow schemes in which System Control can be passed from one device to another; however, rsc could be used in such a scheme. The GPIB-232/485CT-A configures itself to be System Controller at power-on. The assignment made by rsc remains in effect until you call rsc again, call onl, or turn off the GPIB-232/485CT-A.

See Also

sic and sre.

Example 1

```
PRINT #1,"rsc 1"          'Enable GPIB-232/485CT-A to be System
                          'Controller.
```

Example 2

```
PRINT #1,"rsc 0"          'Disable system control.
```

Example 3

```
PRINT #1,"rsc"            'What is the current System Controller
                          'status?
```

response: 0<CR><LF> (GPIB-232/485CT-A is not System Controller)

rsp

Request (Conduct) a Serial Poll

Type

Serial poll function

Syntax

rsp alist<CR>

Purpose

You can use rsp if you want to conduct a serial poll to obtain device-specific status information from one or more GPIB devices.

Remarks

The argument alist is a list of addrs that are separated by commas or spaces. addrs are device addresses that specify the GPIB addresses you want to poll.

rsp serial polls the specified devices to obtain their status bytes. If bit 6 (hex 40 or RQS bit) of a device's response is set, its status response is positive—that is, that device is requesting service. Before rsp completes, all devices are unaddressed.

The interpretation of each device response, other than the RQS bit, is device specific. For example, the polled device might set a particular bit in the response byte to indicate that it has data to transfer, and another bit to indicate a need for reprogramming. Consult the device documentation for interpretation of the response byte.

Each device serial poll response byte is returned as a numeric string giving the decimal value of the byte, followed by <CR> and <LF>. If a device does not respond in the timeout period, the GPIB-232/485CT-A returns the string –1 and records the EABO error. The time limit is set to 1/10 second unless you called tmo to change it. Each response corresponds directly to an address you specify. So the number of response lines, including –1, exactly matches the number of addresses you specify.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

If you call rsp without an argument, the GPIB-232/485CT-A records the EARG error.

See Also

tmo.

Example

```
PRINT #1,"rsp 1+28,5,9"    'Poll 3 devices.

response:      42<CR><LF> (device 9 did not respond
               30<CR><LF> within the timeout period)
               -1<CR><LF>
```

rsv

Request Service/Set or Change Serial Poll Status Byte

Type

Serial poll function

Syntax

rsv [spbyte]<CR>

Purpose

You can use rsv if the GPIB-232/485CT-A is not the GPIB Controller and you want to request service from the Controller using the Service Request (**SRQ***) signal. The GPIB-232/485CT-A provides a user-defined status byte when the Controller serial polls it.

Remarks

The argument spbyte is a numeric string specifying the decimal value of the new GPIB-232/485CT-A serial poll response byte.

The serial poll response byte is the status byte the GPIB-232/485CT-A provides when serial polled by another device that is CIC. If bit 6 (hex 40 or RQS bit) is also set, the GPIB-232/485CT-A requests service by asserting the **SRQ*** line.

If you call rsv without an argument, the GPIB-232/485CT-A returns a numeric string containing the decimal value of its serial poll status byte.

The assignment made by rsv remains in effect until you call rsv again, call onl, or you turn off the GPIB-232/485CT-A.

Example 1

```
PRINT #1,"rsv \x46"        'Request service with serial poll
                           'response = 6.
```

Example 2

```
PRINT #1,"rsv"             'What is the current serial poll status
                           'byte?
response: 70<CR><LF>       (The current status byte = decimal 70 or
                            hex 46)
```

sic

Send Interface Clear

Type

Low-level bus management function

Syntax

sic [time]<CR>

Purpose

You can use sic if the initialization, I/O, or high-level bus management functions do not meet the needs of your device, and you want to have more precise control over the GPIB. sic makes the GPIB-232/485CT-A CIC and initializes the GPIB. sic is not a function you need to use frequently, because in most cases the first I/O or high-level bus management function you call does the same things automatically.

Remarks

The argument time is a numeric string specifying any length of time between .0001 and 3600 seconds, which corresponds to time limits between 100 microseconds and 1 hour. time must not contain a comma.

If you are in a debugging environment, you might want to vary the amount of time **IFC*** is asserted. For example, you might set time to 10 seconds to allow you to check on a bus analyzer that **IFC*** is actually being asserted. Otherwise, you do not need to include the time argument.

If you call sic without an argument, **IFC*** is sent for 500 microseconds. The action of asserting **IFC*** for at least 100 microseconds initializes the GPIB and makes the System Controller become CIC. sic is sometimes used at the beginning of a program to make the GPIB-232/485CT-A CIC and when a bus fault condition is suspected.

The **IFC*** signal resets only the GPIB interface functions of bus devices and not the internal device functions. Device functions are reset with the clr programming message. To determine the effect of these messages, consult the device documentation.

The GPIB-232/485CT-A records the ESAC error if you have disabled its System Controller capability with the rsc function. It records the EARG error if you specify a time outside the range .0001 to 3600.

See Also

clr and Appendix F, *GPIB Basics*.

Example 1

```
PRINT #1,"sic"              'Send interface clear for 500
                            'microseconds.
```

Example 2

```
PRINT #1,"SIC .01"          'Send interface clear for 10
                            'milliseconds.
```

spign

Ignore Serial Port Errors

Type

Serial port function

Syntax

spign [bool]<CR>

Purpose

You can use spign at the beginning of your program if you want to change the effect that serial port errors have on how the GPIB-232/485CT-A processes programming messages and data. This function tells the GPIB-232/485CT-A to ignore or not to ignore the occurrence of serial port errors. By default, serial port errors are ignored.

Remarks

If the argument bool is 0, the GPIB-232/485CT-A does not ignore serial port errors. In this case, the GPIB-232/485CT-A does not execute programming messages that contain serial port errors. Also, if a serial port error occurs with any byte contained in a cmd or wrt data string, the GPIB-232/485CT-A discards that data byte and all remaining bytes in the string.

The serial port errors include parity, overrun, framing, and overflow errors. A list of serial port errors is given in Appendix C, *Status and Error Message Information*.

If the argument bool is 1, the GPIB-232/485CT-A executes all programming messages and sends all data, even if serial port errors occur as the messages and data bytes are received. No serial error code is reported by the stat function.

If you call spign without an argument, the GPIB-232/485CT-A returns the current setting.

The assignment made by spign remains in effect until you call spign again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

cmd and wrt.

Example 1

```
PRINT #1,"spign 0"          'Do not execute programming messages or
                            'process data that contain serial port
                            'errors.
```

Example 2

```
PRINT #1,"spign 1"          'Execute all programming messages and
                            'send all data, even if serial port
                            'errors occur.
```

sre

Set (or Clear) Remote Enable

Type

Low-level bus management function

Syntax

sre [bool]<CR>

Purpose

Use sre to turn the Remote Enable signal on and off. In some cases, the first I/O or high-level bus management function you call sets remote enable automatically.

Remarks

If the argument bool is 1, the GPIB-232/485CT-A asserts the Remote Enable (**REN***) signal. If the argument bool is 0, the GPIB-232/485CT-A unasserts **REN***.

Many GPIB devices have a remote program mode and a local program mode. It is usually necessary to place devices in remote mode before programming them from the GPIB. A device enters the remote mode when the **REN*** line is asserted and the device receives its listen address.

Use cmd to send a device its listen address after using sre. Use loc to return the device to local program mode.

If you call sre with an argument and the GPIB-232/485CT-A is not System Controller, the ESAC error is recorded.

If you call sre without an argument, the GPIB-232/485CT-A returns its current remote status: 1 = remote, 0 = local.

See Also

cmd, loc, and rsc.

Example 1

```
PRINT #1,"SRE 1"          'Set REN*.
```

Example 2

```
PRINT #1,"sre 0"          'Unassert REN*.
```

stat

Return GPIB-232/485CT-A Status

Type

General use function

Syntax

stat [[c] n]<CR>
or
stat [c] s<CR>
or
stat [c] n s<CR>

Purpose

You can use stat to obtain the status of the GPIB-232/485CT-A to see if certain conditions are currently present. stat is used most often to see if the previous operation resulted in an error.

Remarks

You should use stat frequently in the early stages of your program development when the responses of your GPIB devices are likely to be unpredictable.

The GPIB-232/485CT-A responds with status information in a form depending on the mode or combination of modes you choose. n indicates that the status information is returned as numeric strings. s indicates that the status information is returned in symbolic format—that is, as mnemonic strings. Normally, you use s only when you are debugging your code and you want to print the mnemonic for each piece of status information. c specifies that the status is returned after each programming message, eliminating the need to call stat after each programming message.

If you call stat without an argument, continuous status reporting is disabled.

The status information returned by the GPIB-232/485CT-A contains four pieces of information: the GPIB-232/485CT-A status, a GPIB error code, a serial error code, and a count. A <CR><LF> follows each piece of the response.

Status represents a combination of GPIB-232/485CT-A conditions. Inside the GPIB-232/485CT-A, status is stored as a 16-bit integer. Each bit in the integer represents a single condition. A bit value of 1 indicates that the corresponding condition is in effect. A bit value of 0 indicates that the condition is not in effect. Because more than one GPIB-232/485CT-A condition can exist at one time, more than one bit can be set in status. The highest order bit of status, also called the sign bit, is set when the GPIB-232/485CT-A

detects either a GPIB error or a serial port error. When status is negative, an error condition exists.

Table 5-2 lists the values and descriptions of the GPIB status conditions that might be returned by the `stat` function.

Table 5-2. S Mode GPIB Status Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description	Bit
–32768	ERR	Error detected	15
16384	TIMO	Timeout	14
8192	END	**EOI*** or EOS detected	13
4096	SRQI	**SRQ*** detected while CIC	12
2048	—	Reserved	11
1024	—	Reserved	10
512	—	Reserved	9
256	CMPL	Operation completed	8
128	LOK	Lockout state	7
64	REM	Remote state	6
32	CIC	Controller-In-Charge	5
16	ATN	Attention asserted	4
8	TACS	Talker active	3
4	LACS	Listener active	2
2	DTAS	Device trigger active state	1
1	DCAS	Device clear active state	0

The GPIB error code represents a single GPIB error condition present.

The serial error code represents a single serial port error condition present.

`count` is the number of bytes transferred over the GPIB by the last `rd`, `wrt`, or `cmd` function.

Table 5-3 lists values and descriptions of GPIB error conditions that might be returned by the `stat` function. Table 5-4 lists the serial port error conditions that might be returned by the `stat` function.

Table 5-3. S Mode GPIB Error Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description
0	NGER	No GPIB error condition to report
1	ECIC	Command requires GPIB-232/485CT-A to be CIC
2	ENOL	Write detected no Listeners
3	EADR	GPIB-232/485CT-A not addressed correctly
4	EARG	Invalid argument or arguments
5	ESAC	Command requires GPIB-232/485CT-A to be System Controller
6	EABO	I/O operation aborted
7–10	—	Reserved
11	ECAP	No capability for operation
12–13	—	Reserved
14	EBUS	Command bytes could not be sent
15–16	—	Reserved
17	ECMD	Unrecognized command

Table 5-4. S Mode Serial Port Error Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description
0	NSER	No serial port error condition to report
1	EPAR	Serial port parity error
2	EORN	Serial port overrun error
3	EOFL	Serial port receive buffer overflow
4	EFRM	Serial port framing error

A detailed description of the conditions under which each bit in status is set or cleared can be found in Appendix C, *Status and Error Message Information*.

In general, the GPIB-232/485CT-A updates the first three status variables at the end of each programming message. It updates the fourth status variable, `count`, after a `cmd`, `rd`, or `wrt` function. The errors reported correspond to the previous programming message. For example, if you call `wrt` and then `stat s`, any errors returned to you correspond to errors in the `wrt` programming message, not `stat`. However, if status is returned in continuous mode, the status information corresponds to the current programming message. For example, suppose you called `stat c s` to set up continuous status reporting. After reading the status information returned from the `stat` call, you call `wrt`. The GPIB-232/485CT-A then returns the status information that corresponds to the `wrt` message.

When you want to begin continuous status reporting, send the `stat c s`, `stat c n`, or `stat c n s` programming message. Status information is immediately returned indicating the current status conditions. When you call `stat` with both `s` and `n` the numeric status is always returned first.

Notice that when you send several programming messages to the GPIB-232/485CT-A, it buffers them and processes each one without any delay in between. However, if you enable continuous status reporting and check the status of each programming message before sending the next, the GPIB-232/485CT-A waits for each subsequent programming message to arrive at the serial port before processing it. This slows down the overall performance of your program. If speed is a primary concern, disable continuous status reporting.

The continuous status setting remains in effect until you call `stat` again, call `onl`, or you turn off the GPIB-232/485CT-A.

Example 1

```
10   PRINT #1,"stat n"     'Get GPIB-232/485CT-A status.
20                         'GPIB-232/485CT-A responds with:
30                         '340<CR><LF>0<CR><LF>0<CR>
40                         '<LF>0<CR><LF>. Now read
50                         'status into variables.
60   INPUT#1,STATUS%,GPIBERR%,SPERR%,COUNT%
70                         'Go to error routine at 500 if error.
90   IF STATUS% < 0 THEN GOTO 500
100                        'Go to SRQ service routine
110                        'if SRQ is asserted
120  IF (STATUS% AND &H1000) THEN GOTO 400
410  ' Place code here to service SRQ.
500                        'Print GPIB-error and serial-error
                           'values to determine what errors
530                        'occurred.
```

```
540  PRINT "GPIB-error = ";GPIBERR%
550  PRINT "Serial-error = ";SPERR%
560  STOP
```

Example 2

```
10   PRINT #1, "stat s"
20                         'If it has just read 3 bytes from the
30                         'GPIB, the GPIB-232/485CT-A responds
40                         'with:CMPL,CIC,LACS<CR><LF>
50                         'NGER<CR><LF>NSER<CR><LF>3<CR><LF>
```

Example 3

The following list illustrates what appears on the screen when you are programming the GPIB-232/485CT-A from a terminal. GPIB-232/485CT-A responses are in **bold** text. The statements in parentheses are comments.

```
stat c s n                 (Enable continuous status reporting.)
344
0
0
3
CMPL,REM,ATN,TACS          ( Status returned. )
NGER
NSER
3
wrt 10
ABCDE                      ( Write the string ABCDE. )
296                        ( to device 10. )
0                          ( Status returned. )
0
5
CMPL,CIC,TACS
NGER
NSER
5
```

tmo

Change or Disable Time Limit

Type

Initialization function

Syntax

tmo [timeio][,timesp]<CR>

Purpose

You can use tmo at the beginning of your program to change the time limits in effect on the GPIB-232/485CT-A. The time limits prevent the GPIB-232/485CT-A from waiting indefinitely for critical events to occur.

Remarks

The arguments timeio and timesp are numeric strings. timeio specifies the amount of time in seconds the GPIB-232/485CT-A waits for an I/O operation (rd, wrt, cmd) or the wait function to complete. timesp specifies the amount of time in seconds each device is given to respond to a serial poll. The power-on timeouts are 10 seconds for timeio and 0.1 second for timesp.

timeio and timesp can be any decimal number between .00001 and 3600, which corresponds to time limits between 10 microseconds and 1 hour. For example, 10,.1 specifies a time of 10 seconds for I/O operations and 1/10 of a second for serial poll responses. timeio and timesp can also be 0, which disables either timeout accordingly. Neither timeio nor timesp can contain commas.

The timeio time limit is in effect for the cmd, rd, and wrt functions. If the GPIB-232/485CT-A cannot complete any of these functions within the period of time set by timeio, it aborts the function and records the EABO error. Bytes that were transferred before the timeout are not affected. The timeio time limit is also the maximum amount of time the wait function waits when you call it with the TIMO bit set in the wait mask.

The timesp time limit is in effect only for the rsp function. If a polled device fails to respond within the amount of time indicated by timesp, the GPIB-232/485CT-A reports an error.

If you want to change only the timeout value for serial polls, a comma must precede the serial poll timeout value.

If you call tmo without an argument, the GPIB-232/485CT-A returns a numeric string representing the current timeout settings. It records the EARG error if you specify a time value outside the range of .00001 to 3600 seconds.

The assignment made by tmo remains in effect until you call tmo again, call onl, or turn off the GPIB-232/485CT-A.

See Also

rsp.

Example 1

```
PRINT #1,"tmo 30"          'Set timeout for I/O operations to 30
                           'seconds.
```

Example 2

```
PRINT #1,"tmo"             'Print current timeout settings.
```

response: 30,.1<CR><LF>

Example 3

```
PRINT #1,"tmo ,1"          'Set serial poll timeout for one second;
                           'leave I/O timeout unchanged.
```

trg

Trigger Selected Device(s)*

Type

High-level bus management function

Syntax

trg alist<CR>

Purpose

You can use trg to trigger the specified GPIB devices. The documentation for each GPIB device explains when you should trigger it and what effect the trigger has.

Remarks

The argument alist is a list of addrs separated by commas or spaces. addrs are device addresses that specify the GPIB addresses you want to trigger.

If you call trg without an argument, the EARG error is posted.

If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC. It also asserts Remote Enable.

If you passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is posted.

Example

```
PRINT #1,"trg 2+10,4,5+7"  'Trigger 3 devices.
```

wait

Wait for Selected Event

Type

General use function

Syntax

wait mask<CR>

Purpose

You can use wait to monitor selected GPIB events and to delay further GPIB-232/485CT-A activity until any of them occur.

Remarks

The argument mask is a numeric string that specifies the events to wait for. The numeric string represents a bit mask containing a subset of the same bit assignments as the status word described in the stat function. Each bit is set to wait or cleared not to wait, for the corresponding event to occur. The numeric string can be expressed as decimal, octal, or hexadecimal.

After receiving the wait programming message, the GPIB-232/485CT-A monitors GPIB activity. When any event corresponding to the bits set in mask occurs, the GPIB-232/485CT-A returns status information indicating its current status. If continuous status reporting has been enabled, status is reported in the requested format. If continuous status has not been enabled, status is returned in numeric format.

You can use wait, for example, if you want to wait until a device requests service before you perform a serial poll. In this case, you send the wait programming message with mask=4096, and wait for status information to be returned. You then check that status to see if the SRQI bit is set in the returned status indicators.

To prevent the GPIB-232/485CT-A from waiting indefinitely for **SRQ*** to be asserted, set the SRQI and TIMO bits by setting the mask equal to 20480 (16384 + 4096). This causes the wait to terminate either on SRQI or TIMO, whichever occurs first.

Table 5-5 lists the wait mask values.

Table 5-5. Wait Mask Values

Decimal Value	Mnemonic	Description	Hex Value	Bit
—	—	Reserved	—	15
16384	TIMO	Timeout	4000	14
8192	END	**EOI*** or EOS detected	2000	13
4096	SRQI	**SRQ*** detected while CIC	1000	12
—	—	Reserved	—	11
—	—	Reserved	—	10
—	—	Reserved	—	9
—	—	Reserved	—	8
128	LOK	Lockout state	80	7
64	REM	Remote state	40	6
32	CIC	Controller-In-Charge	20	5
16	ATN	Attention asserted	10	4
8	TACS	Talker active	8	3
4	LACS	Listener active	4	2
2	DTAS	Device trigger state	2	1
1	DCAS	Device clear state	1	0

If `mask=0` the function completes immediately and returns the current status.

If the TIMO bit is 0 or the `timeio` time limit is set to 0 by `tmo`, timeouts for this function are disabled. You should disable timeouts only when you are sure that the selected event occurs. Otherwise, the GPIB-232/485CT-A waits indefinitely for the event to occur.

If you call `wait` without an argument, the GPIB-232/485CT-A records the EARG error.

See Also

`stat` and `tmo`.

Example 1

```
PRINT #1,"wait \x5000"
                            'Wait for TIMO or SRQI.
INPUT#1,STATUS%,GPIBERR%,SPERR%,COUNT%
                            'Get status info.
IF (STATUS% AND &H4000) <> 0 THEN GOTO 1000
                            'If TIMO bit is set we timed out before
                            'getting SRQI. Go to an error routine
                            'at line 1000.
IF(STATUS% AND &H1000) <> 0 THEN GOTO 200
                            'If SRQI bit set, go to routine to
                            'conduct a serial poll.
```

Example 2

```
PRINT #1,"wait 4"           'Wait indefinitely to become LACS.
INPUT#1,STATUS%,GPIBERR%,SPERR%,COUNT%
                            'Get status info.
PRINT #1,"rd #10"           'Now that GPIB-232/485CT-A is addressed
                            'to listen, read 10 bytes from the GPIB.
RESP$=INPUT$(10,#1)         'Input 10 bytes from serial port buffer.
INPUT #1,CNT%               'Input number of valid bytes in CNT$.
```

wrt

Write Data*

Type

I/O function

Syntax

wrt [#count][alist]<LF>
data<CR>

Purpose

You can use wrt to send data over the GPIB.

Remarks

The argument data is a string of 8-bit characters that are transferred, without any translation, to the GPIB. The argument count is a numeric string preceded by a number sign (#). count specifies the number of GPIB data bytes to write, and can range from 1 to 4294967295. count must not include the <LF> or <CR> characters, which are used to terminate the programming message or string of GPIB data bytes.

If you call wrt without count, the GPIB-232/485CT-A recognizes the end of the data string when it sees a <CR> or <LF>. Consequently, count is required only if the data string contains a <CR> or <LF> character.

The argument alist is a list of addrs separated by commas or spaces. addrs specify the GPIB addresses of the Listeners.

Do not terminate the wrt programming message with the <CR> character alone. Use either <LF> or <CR><LF> to terminate the programming message. If you use <CR> alone, and the first character of the data string is <LF>, the GPIB-232/485CT-A discards the <LF> as the second character of a <CR><LF> termination.

The GPIB-232/485CT-A writes data to the GPIB until one of the following events occurs:

- The GPIB-232/485CT-A successfully transfers all data
- The I/O time limit is exceeded. The EABO error is recorded.
- The GPIB-232/485CT-A receives a Device Clear. The EABO error is recorded.
- There is no listening device on the bus receiving the data. The ENOL error is recorded.
- The alist argument is specified and the requested GPIB addressing bytes cannot be sent. The EBUS error is recorded.
- A serial port error occurs and is not ignored (see spign).

After wrt terminates, the GPIB-232/485CT-A records the number of data bytes it actually sent. If one of the errors described above occurs the count might be less than expected.

If an error occurs and the GPIB-232/485CT-A is unable to transmit the entire data string, the GPIB-232/485CT-A reads in and discards the remaining bytes of the data string.

If you call wrt with the alist argument, the GPIB-232/485CT-A must be CIC to perform the addressing. If this is the first function you call that requires GPIB Controller capability, and you have not disabled System Controller capability with rsc, the GPIB-232/485CT-A sends Interface Clear (**IFC***) to make itself CIC.

If you call wrt with the alist argument and you previously passed control to some other GPIB device, control must be passed back to you or you must send **IFC*** to make yourself CIC before making this call. Otherwise, the ECIC error is recorded.

When performing the addressing for a specified alist, the GPIB-232/485CT-A sends out its own talk address as well as the listen addresses of the specified devices. It then places itself in Standby Controller state with **ATN*** unasserted and remains there after the write operation is complete.

If the alist argument is not specified, the GPIB-232/485CT-A assumes that it has already been addressed by the Controller. If the GPIB-232/485CT-A is the Controller and did not address itself to talk before calling wrt, the EADR error is recorded and no data bytes are transferred.

See Also

eos, eot, spign, and tmo.

Example 1

```
PRINT #1,"wrt #10 9+97"+CHR$(10)+"0123456789"
                          'Write 10 bytes to device at primary
                          'address 9 and secondary address 97.
```

Example 2

```
PRINT #1,"wrt 2"+CHR$(10)+"ABCDE"
                          'Write the data bytes ABCDE to the
                          'device at address 2.
```

xon

Change Serial Port XON/XOFF Protocol

Type

Serial port function

Syntax

xon [booltx][,boolrx]<CR>

Purpose

You can use xon at the beginning of your program to configure the GPIB-232/485CT-A to communicate over the serial port using the same XON/XOFF protocol as your serial device.

Remarks

The argument booltx specifies whether to enable the XON/XOFF protocol when sending data out on the serial port. If the argument booltx is 1, the GPIB-232/485CT-A monitors its serial receive buffer for XON/XOFF characters as it sends data over the serial port. If it receives the XOFF character (decimal 19 or <Ctrl-S>), it immediately stops sending data. When it receives the XON character (decimal 17 or <Ctrl-Q>), it begins sending data again. If you want to send a data string that might contain a <Ctrl-S> or <Ctrl-Q>, you must disable booltx.

The argument boolrx specifies whether to enable the XON/XOFF protocol when receiving data over the serial port. If the argument boolrx is 1, and the GPIB-232/485CT-A is receiving data over the serial port, it sends XOFF over the serial port when its serial receive buffer is almost full. This tells the sender to stop sending data. When the GPIB-232/485CT-A serial port receive buffer is ready to receive more bytes, the GPIB-232/485CT-A sends XON over the serial port. This tells the sender to begin sending data again.

You should use XON/XOFF when your computer or terminal does not recognize the hardware handshake protocol, and you are transferring very large amounts of data (greater than the serial port buffer size). Without handshaking, there is danger of overflowing the internal buffer of either the GPIB-232/485CT-A or your computer.

The power-on default is that XON/XOFF for both sending and receiving is disabled.

You might not want to enable XON/OFF for both sending and receiving. Some computers use XON/XOFF protocol when transmitting data but not when receiving data. In this case you might configure the GPIB-232/485CT-A using example two in this section.

If you want to change the setting for receiving data, but leave the setting for sending data unchanged, a comma must precede the boolrx argument.

If you call `xon` without an argument, the GPIB-232/485CT-A returns the current handshake settings: 1 if the protocol is enabled or 0 if protocol is disabled.

The assignment made by `xon` remains in effect until you call `xon` again, call `onl`, or turn off the GPIB-232/485CT-A.

Example 1

```
PRINT #1,"XON 1,1"          'Enable GPIB-232/485CT-A XON/XOFF
`                           'protocol for TX and RX.
```

Example 2

```
PRINT #1,"XON 0,1"          'Disable protocol on TX; enable protocol
                            'on RX.
```

Example 3

```
PRINT #1,"XON"              'Return current settings.
```

response: 0,1<CR><LF> (transmit protocol disabled, receive protocol enabled)

Example 4

```
PRINT #1, "XON ,0"          'Disable protocol on RX, keep current
                            'setting on TX.
```

6

Programming in G Mode

This chapter describes how to program the GPIB-232/485CT-A in G mode. It describes status and error handling information, programming considerations, programming messages, function arguments, addressing in G mode, GPIB read and write termination methods, serial port transmissions, and the G mode functions.

This chapter also explains how to communicate with your serial device through the GPIB-232/485CT-A.

Status Information and Error Handling Characteristics

The function descriptions in Chapter 7, *G Mode Functions*, explain that the GPIB-232/485CT-A *records* specific status and error information. This means that it stores that information in its memory so that it is available when you request it.

The function descriptions also explain that the GPIB-232/485CT-A returns certain information to you. This means that the GPIB-232/485CT-A sends information to you over the GPIB.

The GPIB-232/485CT-A continuously monitors the serial port for transmission errors. If it encounters an error in the serial data, the GPIB-232/485CT-A records the error. You can program the GPIB-232/485CT-A to ignore serial port errors using the `spign` function.

Programming Considerations

- The program examples within the function descriptions are written in Microsoft QuickBASIC Version 4.5, using National Instruments NI-488.2 function calls. Although the examples in this manual are written in BASIC, you can program the GPIB-232/485CT-A using any programming language that has access to a GPIB port.

- The following NI-488.2 function call addresses the GPIB-232/485CT-A to listen and sends it the programming message `spset`, followed by a carriage return.

```
WRT$="SPSET"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

  If you are not using the National Instruments NI-488.2 software, be sure your program properly addresses the GPIB-232/485CT-A and the serial device when writing to and reading from them.
- In the function syntax descriptions, arguments shown in square brackets (`[]`) are optional. Do not enter the brackets as part of your argument.
- You can use function name abbreviations, which include only as many characters as necessary to distinguish them from other functions. The abbreviated forms are indicated by **bold** text in the syntax description of each function.

Programming Messages

You can program the GPIB-232/485CT-A by sending it programming messages, which are ASCII strings, by way of its GPIB port.

Programming Message Format

Each programming message is terminated with a carriage return (<CR>), a linefeed (<LF>), or a carriage return followed by a linefeed (<CR><LF>). This is denoted by <CR> in the syntax portions of the function descriptions and by `CHR$(13)` in the BASIC examples.

You can enter programming messages in any combination of uppercase and lowercase letters.

Programming Message Example 1

The following lines of code are an example of a programming message in BASIC:

```
WRT$ ="eos x,10"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

The programming message `WRT$` contains `eos`, which is the function name, `x` and `10` are the arguments, and `CHR$(13)` is the terminating carriage return. This programming message tells the GPIB-232/485CT-A to assert the **EOI*** line when it sends the end-of-string character linefeed. The

second line of the example is a NI-488.2 function call that allows a personal computer to control the GPIB from Microsoft BASIC. This function outputs the string in `WRT$` to the GPIB-232/485CT-A.

Programming Message Example 2

To send more than one programming message with one GPIB write operation, embed a <CR> or an <LF> in the data string you send. For example, to send the two programming messages "change serial port parameters" and "change serial port XON/XOFF protocol", you could use either of the two following sequences.

- You can send two separate strings to the GPIB in two separate GPIB writes as in the following example:

```
WRT$="SPSET 1200,n,8"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
WRT2$="XON 1,1"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

- You could also put both messages in one string and send it to the GPIB-232/485CT-A in one GPIB write as in the following example:

```
WRT$="SPSET 1200,n,8"+CHR$(13)+"XON 1,1"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

How Messages are Processed

The GPIB-232/485CT-A processes each programming message on a line-by-line basis. When the GPIB-232/485CT-A receives a message, it buffers the message, interprets the function name and arguments, then executes the message.

Function Arguments

When you type in a function, separate the first argument from the function name with at least one space. Separate each additional argument with at least one space or comma.

In the syntax portions of the function descriptions in Chapter 7, *G Mode Functions*, the square brackets (`[]`) that enclose some arguments indicate that those arguments are optional. Do not enter the brackets as part of your arguments.

Abbreviations for Arguments

The term `bool` is an abbreviation used for an argument in the function descriptions. The values for `bool` are 1 = true, on, or enable and 0 = false, off, or disable.

Addressing the GPIB-232/485CT-A and Serial Device

The GPIB-232/485CT-A uses dual addressing to determine what type of GPIB data it is processing. With dual addressing, the GPIB-232/485CT-A recognizes two different GPIB addresses. The first address is the GPIB-232/485CT-A address. When the GPIB-232/485CT-A receives its own address, the data it receives is referred to as *programming messages*; the data it sends is referred to as *status information*. The second address is the serial device address. When the GPIB-232/485CT-A receives the serial device address, the data it sends and receives is referred to as *serial data*.

Address of the GPIB-232/485CT-A

The address of the GPIB-232/485CT-A is the primary address you set with the DIP switch on the rear panel, with secondary addressing disabled.

Address of the Serial Device

The address of the serial device is the GPIB-232/485CT-A primary address plus 1, with secondary addressing disabled. However, if you select a primary address of 30 with the configuration switch, the serial device is at address 0. The primary address is set with the DIP switch on the rear panel.

Addressing the GPIB-232/485CT-A and Serial Device as Listeners

When the GPIB-232/485CT-A receives its own listen address, it examines the data received over the GPIB, treats it as a programming message, and takes actions based on that data.

When the GPIB-232/485CT-A receives the serial device listen address, it forwards the data received over the GPIB to the serial port without examining or modifying the data.

For example, you might have a serial printer connected to the GPIB-232/485CT-A and you want to send a data file from your computer over the GPIB to the printer. Ordinarily, when the printer buffer is full, the printer sends the XOFF character; when the printer is ready to receive more characters, it sends the XON character. So, before you send your file to the printer, you must tell the GPIB-232/485CT-A to watch for XON/XOFF characters from the printer.

Follow these steps to send a data file from your computer to a serial printer:

1. Address the GPIB-232/485CT-A to listen by sending its listen address.
2. Send the programming message `xon ,1` to the GPIB-232/485CT-A. The GPIB-232/485CT-A interprets this programming message and acts upon it without sending any data on to the serial device.
3. Address the serial device to listen by sending the serial device listen address to the GPIB-232/485CT-A.
4. Send your data file over the GPIB. The GPIB-232/485CT-A sends the data to the printer without examining it for meaning.

   The **LISTEN** LED on the GPIB-232/485CT-A is lit when either the GPIB-232/485CT-A or the serial device is addressed to listen.

Addressing the GPIB-232/485CT-A and Serial Device as Talkers

When the GPIB-232/485CT-A receives its own talk address, it sends out status information.

When the GPIB-232/485CT-A receives the serial device talk address, it sends data out to the GPIB that it has received from the serial device.

For example, you might have your serial device programmed to perform some calculations and you want it to return the data to you. Follow these steps to retrieve the data:

1. Address the serial device to talk by sending the serial device talk address to the GPIB-232/485CT-A. If the GPIB-232/485CT-A receives the serial device talk address but has no data in its serial port receive buffer to send, it waits for data from the serial device to fill the request.
2. Perform a GPIB read of 100 bytes. The GPIB-232/485CT-A retrieves 100 bytes from its serial port receive buffer and sends them to you.
3. Find out if the serial device has sent more bytes to the GPIB-232/485CT-A by asking the GPIB-232/485CT-A to send you status information.
   a. Send the GPIB-232/485CT-A its listen address.
   b. Send the programming message `stat n` to the GPIB-232/485CT-A.
   c. Send the GPIB-232/485CT-A its talk address.
   d. Perform a GPIB read of 20 bytes. The GPIB-232/485CT-A sends you its status information, terminated by the GPIB END message. If the GPIB-232/485CT-A receives its talk address but has nothing

to send, it responds to GPIB reads with a carriage return and a linefeed, accompanied by END.

The status information returned to you contains the number of bytes remaining in the serial port receive buffer. This information helps you decide how much data to continue to collect from the serial device.

The **TALK** LED on the GPIB-232/485CT-A is lit when either the GPIB-232/485CT-A or the serial device is addressed to talk.

GPIB Read and Write Termination Methods (END and EOS)

The IEEE 488 specification defines two ways that GPIB Talkers and Listeners can identify the last byte of data messages: END and EOS. The two methods permit a Talker to send data messages of any length without the Listener(s) knowing the number of transmission bytes in advance. END and EOS can be used individually or in combination, but the Listener must be configured to detect the end of a transmission.

END message	The Talker asserts the **EOI*** (End or Identify) signal while the last data byte is being transmitted. The Listener stops reading when it detects a data byte accompanied by **EOI***, regardless of the value of the byte.
EOS character	The Talker transmits an EOS (end-of-string) character at the end of its data string. The Listener stops receiving data when it detects the EOS character. Either a 7-bit ASCII character or a full 8-bit binary byte can be used.

When the GPIB-232/485CT-A receives its own talk or listen address, no EOS modes are in effect. When talking, the GPIB-232/485CT-A asserts **EOI*** with the last byte of its response. When the GPIB-232/485CT-A receives the serial device talk address, the EOS modes in effect are those that you select using the `eos` function.

Serial Port Transmission

The GPIB-232/485CT-A checks the data received from the serial device for errors while it buffers data. If a serial port error occurs, the GPIB-232/485CT-A records the appropriate error code. There are two ways to determine if a serial port error has occurred:

- Use the `stat` function to request the GPIB-232/485CT-A status information. After the serial error code has been reported, it is automatically cleared and no further action to the GPIB-232/485CT-A is necessary.
- Serial poll the GPIB-232/485CT-A and check the serial poll response byte to see if its SERR bit is set. Refer to *Serial Poll Responses* later in this chapter.

You can program the GPIB-232/485CT-A to ignore serial port errors using the `spign` function.

Operation of the GPIB-232/485CT-A as a GPIB Device

In G mode, the GPIB-232/485CT-A operates like any other GPIB device and, as such, is configured to respond in certain ways to GPIB commands.

Serial Poll Responses

When serial polled, the GPIB-232/485CT-A returns status information to the GPIB Controller through the serial poll response byte. The GPIB-232/485CT-A maintains this status byte throughout operation, regardless of the `srqen` configuration.

Table 6-1 lists the meaning of each bit in the serial poll response byte.

Table 6-1. Serial Poll Response Byte

Bit	Mnemonic	Meaning
0	—	Not Used
1	BF	Serial port receive buffer is full and serial device not addressed to talk
2	GERR	GPIB error; see `stat`
3	SERR	Serial error; see `stat`

Table 6-1. Serial Poll Response Byte (Continued)

Bit	Mnemonic	Meaning
4	BNE	Serial port receive buffer not empty and serial device not addressed to talk
5	EOS	EOS character received and serial device not addressed to talk
6	RQS	Request service (**SRQ*** asserted)
7	—	Not Used

Service Request Conditions

You can program the GPIB-232/485CT-A to assert Service Request (**SRQ***) in a variety of cases. After you power on, the GPIB-232/485CT-A defaults to never asserting service request. Using the `srqen` function, you can program the GPIB-232/485CT-A to assert **SRQ*** under any of the following circumstances:

- When a GPIB error occurs, that is, EARG, ECMD, or ECAP as reported by `stat`.
- When a serial port error occurs—that is, EPAR, EORN, EOFL, or EFRM as reported by `stat`.
- When the serial device is not addressed as a Talker, and
  - the serial port receive buffer is full, or
  - any byte is received from the attached serial device, or
  - the EOS byte is received from the attached serial device.

Parallel Polls

The GPIB-232/485CT-A sets the `ist` (individual status) bit whenever it asserts **SRQ***, and clears `ist` whenever it unasserts **SRQ***. The GPIB-232/485CT-A implements IEEE 488 Parallel Poll (PP) interface function subset PP1. This means that it cannot configure itself to respond to parallel polls. It must be configured remotely by an external Controller.

See Appendix H, *Parallel Polling*, for more information.

Take Control (TCT)

This command has no effect on the GPIB-232/485CT-A. It would not make sense for control to pass to the GPIB-232/485CT-A, since all programming messages and GPIB commands must be sent to it from another GPIB device.

Group Execute Trigger (GET)

This command has no effect on the GPIB-232/485CT-A.

Go To Local (GTL)

This command has no effect on the GPIB-232/485CT-A.

Device Clear

When the GPIB-232/485CT-A receives the universal Device Clear (DCL) command or when it receives its listen address and the Selected Device Clear (SDC) command, it clears both its status buffer and its serial port receive buffer. It also resets the GPIB serial poll response byte to zero and unasserts **SRQ***.

Function Names

The function names have been selected to indicate the purpose of each function, thereby making your programs easy to understand. However, if you want to reduce some overhead in your program and do not mind giving up these advantages, you can use only as much of the function name as is necessary to distinguish it from other functions. This abbreviated form of the function name is shown in **bold** text in the function tables and in the syntax portions of the function descriptions.

G Mode Default Settings and Related Functions

Tables 6-2 and 6-3 list power-on characteristics of the GPIB-232/485CT-A and the functions you can use to change those characteristics.

Table 6-2. G Mode Serial Port Characteristics

Characteristic	Power-On Value	Related Function
Echo bytes to serial port	no	**ec**ho
Enable serial port communication	yes	**on**l
Baud rate	9600	**sps**et
Parity	none	**sps**et
Data bits	8	**sps**et
Stop bits	1	**sps**et
Send XON/XOFF	no	**x**on
Recognize XON/XOFF	no	**x**on
Report serial errors	no	**spi**gn

Table 6-3. G Mode GPIB Characteristics

Characteristic	Power-On Value	Related Function
End-of-string modes	none	**eos**
Allow GPIB-232/485CT-A to assert SRQ	no	**srq**en

List of G Mode Functions by Group

The GPIB-232/485CT-A functions are divided into three groups: GPIB functions, serial port functions, and general use functions. For more information about the G mode functions, refer to the alphabetical list of functions at the end of this chapter, or to Chapter 7, *G Mode Functions*.

The following are programming messages that are sent to the GPIB-232/485CT-A from a GPIB Talker.

GPIB Functions

GPIB functions manage the GPIB port of the GPIB-232/485CT-A.

- **eos**
- **srq**en

Serial Port Functions

Serial port functions initialize and manage the serial port of the GPIB-232/485CT-A.

- **ec**ho
- **sps**et
- **x**on
- **spi**gn

General Use Functions

General use functions are used for general operations that are not provided by the GPIB functions or serial port functions.

- **id**
- **o**nl
- **st**at

Alphabetical List of G Mode Functions

Table 6-4 lists all of the G mode functions in alphabetical order.

Table 6-4. Alphabetical List of G Mode Functions

Function	Purpose
echo on/off	Echo characters received from serial port
eos modes, eoschar	Change or disable GPIB end-of-string termination mode
id	Identify system
onl on/off	Place the GPIB-232/485CT-A online/offline
spign on/off	Ignore serial port errors
spset modes	Change serial port parameters

Table 6-4. Alphabetical List of G Mode Functions (Continued)

Function	Purpose
srqen mask	Set conditions for asserting SRQ*
stat options	Return GPIB-232/485CT-A status
xon modes	Change serial port XON/XOFF protocol

7

G Mode Functions

This chapter contains descriptions of the G mode functions that you can use to program the GPIB-232/485CT-A. These functions are in alphabetical order for easy reference.

For general information about using G mode functions, refer to Chapter 6, *Programming in G Mode*.

echo

Echo Characters Received from Serial Port

Type

Serial port function

Syntax

echo [bool]<CR>

Purpose

You can use echo when a terminal is connected to the GPIB-232/485CT-A and you want everything you type to display on the screen.

Remarks

If the argument bool is 1, characters received from the serial port are echoed back to the serial port. If the argument bool is 0, characters are not echoed. By default, echoing is disabled.

If you call echo without an argument, the GPIB-232/485CT-A returns the current setting.

In a debugging environment, the success of your communication with the serial device might be unclear. In this case, you could connect a terminal to the GPIB-232/485CT-A instead of connecting the serial device. Then the data that the GPIB-232/485CT-A would send to the serial device is displayed on the terminal screen. Also, you can type characters on the terminal to send to the GPIB-232/485CT-A, just as your serial device would.

The assignment made by echo remains in effect until you call echo again, call onl, or you turn off the GPIB-232/485CT-A.

Example 1

```
WRT$="echo 1"+CHR$(13)          'Enable character
CALL IBWRT(GPIB232%,WRT$)       'echoing.
```

Example 2

```
WRT$="ECHO 0"+CHR$(13)          'Disable character
CALL IBWRT(GPIB232%,WRT$)       'echoing.
```

Example 3

```
WRT$="echo"+CHR$(13)            'What is the
CALL IBWRT(GPIB232%,WRT$)       'current echo
CALL IBRD(GPIB232%,RESP$)       'status?
                                'RESP$ contains
                                '0<CR><LF>
                                '(character echo is
                                'disabled).
```

eos

Change/Disable GPIB EOS Termination Mode

Type

GPIB function

Syntax

eos [X[B]eoschar]<CR>
or
eos D<CR>

Purpose

You can use eos to enable the GPIB-232/485CT-A to add the GPIB END message to the data string sent by the serial device when the string contains the specified end-of-string character.

Remarks

eos applies only when the GPIB-232/485CT-A has received the serial device talk address and is sending serial data to the GPIB. It does not apply when the GPIB-232/485CT-A reads serial data or programming messages from the GPIB. The arguments X, B, and D specify GPIB data transfer termination methods.

The argument eoschar is a numeric string representing a single ASCII character that is to be the EOS byte. The arguments X and B are used to enable the corresponding EOS mode. The argument D disables all EOS modes.

The three termination methods are shown in Table 7-1.

Table 7-1. Data Transfer Termination Methods in G Mode

Description	Letter
XEOS—Set **EOI*** with EOS when sending data from serial device	X
BIN—Compare all 8 bits of EOS byte rather than low 7 bits	B
DISABLE—Disable all EOS modes	D

If Methods X, or X and B are chosen, the GPIB-232/485CT-A automatically sends the END message along with eoschar when performing GPIB writes of serial data. That is, when the GPIB-232/485CT-A receives eoschar over the serial port and sends it on to the GPIB, it also asserts **EOI*** along with that byte. When X alone is chosen, END is sent with the EOS byte when the low seven bits of that byte match the low seven bits of eoschar. When X and B are chosen, a full 8-bit comparison is used.

If B is the only mode chosen, the EARG error is posted.

If D is chosen, all EOS modes are disabled.

By default, all EOS modes are disabled.

If you call eos without an argument, the GPIB-232/485CT-A returns the current eos settings.

The assignment made by this function remains in effect until you call eos again, you call onl, or you turn off the GPIB-232/485CT-A.

See Also

srqen and the *GPIB Read and Write Termination Methods (END and EOS)* section in Chapter 6, *Programming in G Mode*.

Example 1

```
WRT$="EOS X,13"+CHR$(13)        'Send EOI with <CR>.
CALL IBWRT(GPIB232%,WRT$)       'Compare 7 bits.
```

Example 2

```
WRT$="eos"+CHR$(13)             'What are the
CALL IBWRT(GPIB232%,WRT$)       'current EOS
                                'settings?
CALL IBRD(GPIB232%,RESP$)       'RESP$ contains
                                'X,13<CR><LF>.
```

id

Identify System

Type

General use function

Syntax

id<CR>

Purpose

You can use `id` if you want to know the revision level of your software or how much RAM is installed in your GPIB-232/485CT-A.

Remarks

The identification is returned in three strings. The first string identifies the company product model and software revision level. The second string is a copyright notice. The third string identifies the number of bytes of RAM in the GPIB-232/485CT-A.

The following example shows the current identification string at the time of this printing. The general format will be as shown; however, version-specific information such as revision levels and copyright dates change as needed.

Example

```
WRT$="id"+CHR$(13)          'Get system identification.
CALL IBWRT(GPIB232%,WRT$)

CALL IBRD (GPIB232%,RESP$)

RESR$ contains:    GPIB-232/485CT-A, Rev. B.3<CR><LF>
                   (c)1995 National Instruments<CR><LF>
                   256K bytes RAM<CR><LF>
```

onl

Place the GPIB-232/485CT-A Online/Offline

Type

General use function

Syntax

`onl [bool]`<CR>

Purpose

You can use `onl` to disable communication between the GPIB-232/485CT-A and the serial port, or to reset the GPIB-232/485CT-A characteristics to their default values.

Remarks

If the argument `bool` is 1, the GPIB-232/485CT-A places itself online. If the argument `bool` is 0, it places itself offline. By default, the GPIB-232/485CT-A powers up online.

If you call `onl` without an argument, the GPIB-232/485CT-A returns its current state, which is 0 if the GPIB-232/485CT-A is offline and 1 if the it is online.

Placing the GPIB-232/485CT-A offline is like disconnecting its serial cable from the serial device. When placed offline, no data is sent out the GPIB-232/485CT-A serial port and data coming in to the GPIB-232/485CT-A serial port is not saved until `onl` is called with `bool` = 1.

Placing the GPIB-232/485CT-A online enables data to be sent and received over its serial port. Placing the GPIB-232/485CT-A online also restores all its settings to their power-on default values.

See Also

The *G Mode Default Settings and Related Functions* section in Chapter 6, *Programming in G Mode*.

Example 1

```
WRT$="onl 1"+CHR$(13)           'Put the GPIB-232/485CT-A
CALL IBWRT(GPIB232%,WRT$)       'online, and restore its
                                'power-on settings.
```

Example 2

```
WRT$="ONL 0"+CHR$(13)           'Put the GPIB-232/485CT-A
CALL IBWRT(GPIB232%,WRT$)       'offline.
```

spign

Ignore Serial Port Errors

Type

Serial port function

Syntax

spign [bool]<CR>

Purpose

You can use spign at the beginning of your program if you want to change the effect that serial port errors have on the storage of a character received with a serial error. This function tells the GPIB-232/485CT-A to ignore or not to ignore the occurrence of serial port errors. By default, the GPIB-232/485CT-A ignores serial port errors.

Remarks

If the argument bool is 0, the GPIB-232/485CT-A does not ignore serial port errors. In this case, the GPIB-232/485CT-A does not store characters that contain serial errors. Also, the error is indicated in the serial error code that is returned by the stat function.

The serial port errors include parity, overrun, framing, and overflow errors. Appendix C, *Status and Error Message Information*, includes a list of serial port errors.

If the argument bool is 1, the GPIB-232/485CT-A ignores serial port errors. Bytes that arrive with serial port errors are stored in the buffer and are later sent out of the GPIB port when the serial device is talk addressed. No serial error code is reported by the stat function.

If you call spign without an argument, the GPIB-232/485CT-A returns the current setting.

The assignment made by spign remains in effect until you call spign again, call onl, or you turn off the GPIB-232/485CT-A.

Example 1

```
WRT $,"spign 0"+CHR$(13)        'Do not ignore
CALL IBWRT(GPIB232%,WRT$)       'serial port errors.
```

Example 2

```
WRT $,"spign 1"+CHR$(13)        'Ignore serial port
CALL IBWRT(GPIB232%,WRT$)       'errors.
```

spset

Change Serial Port Parameters

Type

Serial port function

Syntax

spset [baud] [parity] [databits] [stopbits]<CR>

Purpose

You can use spset at the beginning of your program to set the GPIB-232/485CT-A serial port characteristics (baud rate, parity, data bits, and stop bits) to match those required by your serial device.

Remarks

The argument baud is a numeric string specifying the baud rate (300, 600, 1200, 2400, 4800, 9600, 19200, 38400). The argument parity is a character specifying the parity (e for even, o for odd, n for none). The argument databits is a character specifying the number of data bits (7 or 8). The argument stopbits is a character specifying the number of stop bits (1 or 2). If you call spset without an argument, the GPIB-232/485CT-A returns its current serial port configuration.

Until you call spset, the following characteristics are in effect: 9600 is the baud rate, parity is disabled, 8 is the number of data bits, and 1 is the number of stop bits.

If you must reconfigure the GPIB-232/485CT-A serial port, wait until communication with the serial port is not taking place.

The assignment made by spset remains in effect until you call spset again, call onl, or you turn off the GPIB-232/485CT-A.

Example 1

```
'Set up the serial port of GPIB-232/485CT-A to
'keep its current baud rate, current parity,
'and to use 7 data bits and 2 stop bits.
WRT$="spset 7 2"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

Example 2

```
'What are the current GPIB-232/485CT-A serial port settings?
WRT$="SPSET"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
'RESP$ will contain 9600,N,7,2<CR><LF>
'(9600 baud, no parity, 7 data bits,
'2 stop bits).
CALL IBRD(GPIB232%,RESP$)
```

Example 3

```
'Set the GPIB-232/485CT-A serial port to 1200
'baud, no parity, 8 data bits, and 1 stop bit.
WRT$="spset 1200 n 8 1"+CHR$(13)
CALL IBWRT(GPIB232%,WRT$)
```

srqen

Enable/Disable Setting of SRQ*

Type

GPIB function

Syntax

srqen [mask]<CR>

Purpose

You can use srqen when you want to allow the GPIB-232/485CT-A to assert **SRQ*** under the conditions described in Chapter 6, *Programming in G Mode*, the section entitled *Service Request Conditions*.

Remarks

When the argument mask is 0, the GPIB-232/485CT-A never asserts **SRQ***. When the argument mask is > 0, the GPIB-232/485CT-A asserts **SRQ*** under the conditions represented by each bit in the mask. Table 7-2 describes the mask bits.

Table 7-2. SRQ Mask Bits in G Mode

Bit	Hex Value	Decimal Value	Mnemonic	Description
0	1	1	—	Not used
1	2	2	BF	Serial port receive buffer full and serial device not addressed to talk
2	4	4	GERR	GPIB error; see stat
3	8	8	SERR	Serial error; see stat
4	10	16	BNE	Serial port receive buffer not empty and serial device not addressed to talk
5	20	32	EOS	EOS character received and serial device not addressed to talk
6	40	64	—	Not used
7	80	128	—	Not used

To determine the mask value you want, add up the hex or decimal values of each of the conditions on which you want **SRQ*** to be asserted. For example, if you want **SRQ*** asserted on GPIB errors and serial port errors you should call srqen with a mask of 12 (4 for GERR and 8 for SERR).

The power on default of srqen is disabled—that is, **SRQ*** is never asserted.

If you call srqen without an argument, the GPIB-232/485CT-A returns a decimal string that indicates the decimal value of the current setting.

The assignment made by srqen remains in effect until you call srqen again, call onl, or you turn off the GPIB-232/485CT-A.

See Also

eos, spign, and stat.

Example 1

```
WRT$="srqen 0"+CHR$(13)          'Never assert SRQ*.
CALL IBWRT(GPIB232%,WRT$)
```

Example 2

```
WRT$="srqen 4"+CHR$(13)          'Assert SRQ* when
CALL IBWRT(GPIB232%,WRT$)        'a GPIB error occurs.
```

stat

Return GPIB-232/485CT-A Status

Type

General use function

Syntax

stat [[c] n]<CR>
or
stat [c] s<CR>
or
stat [c] n s<CR>

Purpose

You can use stat to obtain the status of the GPIB-232/485CT-A to see if certain conditions are currently present. stat is used most often to see if the previous operation resulted in an error.

Remarks

The GPIB-232/485CT-A returns status information to you in a form depending on the mode or combination of modes you choose. n indicates that the status information is returned as numeric strings. s indicates that the status information is returned in symbolic format—that is, as mnemonic strings. c specifies that the status is returned after each programming message, eliminating the need to call stat after each programming message.

You would probably use s, or symbolic format, only when you are debugging your code and you want to print the mnemonic for each piece of status information.

If you call stat without an argument, continuous status reporting is disabled.

The status information returned by the GPIB-232/485CT-A contains four pieces of information: the GPIB-232/485CT-A status, a GPIB-error code, a serial-error code, and a count. The GPIB-232/485CT-A returns a <CR><LF> following each piece of the response and asserts **EOI*** with the final <LF> that comes after count.

Status represents a combination of GPIB-232/485CT-A conditions. Inside the GPIB-232/485CT-A, status is stored as a 16-bit integer. Each bit in the integer represents a single condition. A bit value of 1 indicates the corresponding condition is in effect. A bit value of 0 indicates the condition is not in effect. Because more than one GPIB-232/485CT-A condition can exist at one time, more than one bit can be set in status. The highest order bit of status, also called the sign bit, is set when the GPIB-232/485CT-A detects either a GPIB error or a serial port error. When status is negative, an error condition exists.

Table 7-3 lists the values and descriptions of GPIB status conditions that might be returned by the `stat` function.

Table 7-3. G Mode GPIB-232/485CT-A Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description	Bit
–32768	ERR	Error detected	15
16384	—	Reserved	14
8192	—	Reserved	13
4096	—	Reserved	12
2048	—	Reserved	11
1024	—	Reserved	10
512	—	Reserved	9
256	CMPL	Operation completed	8
128	—	Reserved	7
64	—	Reserved	6
32	—	Reserved	5
16	—	Reserved	4
8	—	Reserved	3
4	—	Reserved	2
2	—	Reserved	1
1	—	Reserved	0

The GPIB error code represents a single GPIB error condition present. The serial error code represents a single serial error condition present. `count` is the number of bytes currently contained in the GPIB-232/485CT-A serial port receive buffer.

Table 7-4 lists the values and descriptions of GPIB error conditions that might be returned by the `stat` function. Table 7-5 lists the serial port errors that might be returned by the `stat` function.

Table 7-4. G Mode GPIB Error Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description
0	NGER	No GPIB error condition to report
1	—	Reserved
2	—	Reserved
3	—	Reserved
4	EARG	Invalid argument or arguments
5	—	Reserved
6	—	Reserved
7–10	—	Reserved
11	ECAP	No capability for operation
12–16	—	Reserved
17	ECMD	Unrecognized command

Table 7-5. Serial Port Error Conditions Returned by stat

Numeric Value (n)	Symbolic Value (s)	Description
0	NSER	No serial port error condition to report
1	EPAR	Serial port parity error
2	EORN	Serial port overrun error
3	EOFL	Serial port receive buffer overflow
4	EFRM	Serial port framing error

Appendix C, *Status and Error Message Information*, includes a detailed description of the conditions under which each bit in status is set or cleared.

The GPIB-232/485CT-A updates the status and count variables at the end of each programming message. The serial and GPIB error variables are updated whenever a new error occurs and are cleared automatically when status is reported.

When you want to begin continuous status reporting, send the `stat c s`, `stat c n`, or `stat c n s` programming message. When you call stat with both `n` and `s` modes specified, the numeric status is always returned first.

The continuous status setting remains in effect until you call `stat` again, call `onl`, or turn off the GPIB-232/485CT-A.

Example 1

```
 10  'Tell GPIB-232/485CT-A to send numeric
 20  'status.
 30  WRT$="stat n"+CHR$(13)
 40  CALL IBWRT(GPIB232%,WRT$)
 50  'Now read the status from the
 60  'GPIB-232/485CT-A.
 70  STATUS$=SPACE$(10) : GPIBERR$=SPACE$(10)
 80  SPERR$=SPACE$(10) : COUNT$=SPACE$(10)
 90  CALL IBRD(GPIB232%,STATUS$)
100  'Read up to 10 bytes of each piece of status.
110  'The GPIB-232/485CT-A returns 4 pieces of
120  'status. We are set up to terminate
130  'IBRD on linefeed, which is what
140  'terminates each piece of status.
150  STATUS% = VAL(STATUS$)
160  CALL IBRD(GPIB232%,GPIBERR$)'Read GPIB-error.
170  CALL IBRD(GPIB232%,SPERR$) 'Read serial-error.
180  CALL IBRD(GPIB232%,COUNT$) 'Read count.
190  'Call error routine at 500 if error occurred.
200  IF STATUS% < 0 THEN GOTO 500
...
500  'Print GPIB-error, and serial-error
510  'values to determine what errors occurred.
520  PRINT "GPIB-error = ";GPIBERR$
530  PRINT "serial-error = ";SPERR$
540  STOP
```

Example 2

```
 10  'Turn on continuous status reporting,
 20  'in numeric format.
 30  WRT$="stat c n"+CHR$(13)
 40  CALL IBWRT(GPIB232%,WRT$)
 50  'If we have 3 bytes in the serial port
 60  'buffer, a typical response would be:
 70  '262<CR><LF>0<CR><LF>0<CR><LF>3<CR><LF>
 80  'Read the GPIB-232/485CT-A status; read 30
```

```
 90  'bytes or until EOI is received.
100  RD$=SPACE$(30)
110  CALL IBRD(GPIB232%,RD$)
120  'Print the status information.
130  PRINT "GPIB-232/485CT-A status is: ";RD$
```

Example 3

```
 10  'Turn on continuous status reporting,
 20  'in symbolic format.
 30  WRT$="stat c s"+CHR$(13)
 40  CALL IBWRT(GPIB232%,WRT$)
 50  'Read the GPIB-232/485CT-A status; read 50
 60  'bytes or until EOI is received.
 70  RD$=SPACE$(50)
 80  CALL IBRD(GPIB232%,RD$)
 90  'Print the status information.
100  PRINT "GPIB-232/485CT-A status is: ";RD$

Printed information is:

GPIB-232/485CT-A status is :
CMPL
NGER
NSER
3
```

xon

Change Serial Port XON/XOFF Protocol

Type

Serial port function

Syntax

xon [booltx][,boolrx]<CR>

Purpose

You can use xon at the beginning of your program to configure the GPIB-232/485CT-A to communicate over the serial port using the same XON/XOFF protocol as your serial device.

Remarks

The argument booltx specifies whether to enable the XON/XOFF protocol when sending data out on the serial port. When booltx is enabled, the GPIB-232/485CT-A monitors its serial receive buffer for XON/XOFF characters as it sends data over the serial port. If it receives the XOFF character (decimal 19 or <Ctrl-S>), it immediately stops sending data. When it receives the XON character (decimal 17 or <Ctrl-Q>), it begins sending data again. If you want to send a data string that might contain a <Ctrl-S> or <Ctrl-Q>, you must disable booltx.

The argument boolrx specifies whether to enable the XON/XOFF protocol when receiving data over the serial port. When boolrx is enabled, as the GPIB-232/485CT-A receives data over the serial port, it sends XOFF over the serial port when its serial port receive buffer is almost full. This tells the sender to stop sending data. When the GPIB-232/485CT-A serial port receive buffer is ready to receive more bytes, the GPIB-232/485CT-A sends XON over the serial port. This tells the sender to begin sending data again.

You should use XON/XOFF if your serial device does not recognize the hardware handshake protocol, and you are transferring large amounts of data at high speeds. Without handshaking, there is danger of overflowing the serial port receive buffers of the GPIB-232/485CT-A and the serial device.

The power-on default of XON/XOFF for both sending and receiving is disabled.

You might not want to enable XON/XOFF for both sending and receiving. Some serial devices use XON/XOFF protocol when transmitting data but not when receiving. In this case, you might configure the GPIB-232/485CT-A using example 2 in this section.

If you want to change the setting for receiving data, but leave the setting for sending data unchanged, a comma must precede the boolrx argument.

If you call `xon` without an argument, the GPIB-232/485CT-A returns the current handshake setting: 1 if the protocol is enabled or 0 if protocol is disabled.

The assignment made by `xon` remains in effect until you call `xon` again, call `onl`, or turn off the GPIB-232/485CT-A.

Example 1

```
WRT$="XON 1,1"+CHR$(13)          'Enable GPIB-232/485CT-A
CALL IBWRT(GPIB232%,WRT$)  'XON/XOFF protocol for TX and RX.
```

Example 2

```
WRT$="XON 0,1"+CHR$(13)          'Disable protocol on
CALL IBWRT(GPIB232%,WRT$)  'TX; enable protocol on RX.
```

Example 3

```
WRT$="XON"+CHR$(13)              'Return current settings.
CALL IBWRT(GPIB232%,WRT$)
CALL IBRD(GPIB232%,RESP$)  'RESP$ contains 0,1<CR><LF>
                                 '(transmit protocol disabled,
                                 'receive protocol enabled).
```

A

Specifications

This appendix contains tables which specify the electrical, environmental, and physical characteristics of the GPIB-232/485CT-A, as well as the IEEE 488 capability codes.

Electrical Characteristics

AC Version

Power supply unit (50–60 Hz) ..... 100–120 VAC ± 10% or
220–240 VAC ± 10%

Current

100–120 VAC ..... 55 mA
220–240 VAC ..... 45 mA

Fuse rating and type

100–120 VAC ..... 300 mA, UL/CSA approved
220–240 VAC ..... 500 mA, IEC approved

DC Version

Power supply unit (50–60 Hz input, 9 VDC at 1 A output)

Wall-mount type ..... 100–120 VAC ± 10%
Desktop type ..... 220–240 VAC ± 10%

DC input ..... +5 to +13, regulated

Current ..... 700 mA minimum

Environmental Characteristics

AC Version

Operating temperature .....10 to 40 °C

Storage temperature.....0 to 70 °C

Relative humidity .....10% to 95%, noncondensing

EMI.....FCC Class A Verified

DC Version

Operating temperature .....10 to 40 °C

Storage temperature.....0 to 70 °C

Relative humidity .....10% to 95%, noncondensing

EMI.....FCC Class B Certified

Physical Characteristics

AC Version

Overall case size (dimensions) .....118.1 by 76.2 by 44.2 mm
(4.65 by 3.0 by 1.74 in.)

Case material .....All metal enclosure

Weight .....340 g (12 oz)

DC Version

Overall case size (dimensions) .....118.1 by 76.2 by 28.2 mm
(4.65 by 3.0 by 1.11 in.)

Case material .....All metal enclosure

Weight .....198 g (7 oz)

The IEEE 488 standard specifies allowable subsets of interface functions. The codes supported by the GPIB-232/485CT-A are detailed in Table A-1. For more information, refer to the ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*.

Table A-1. IEEE 488 Capability Codes for the GPIB-232/485CT-A

S Mode Code	G Mode Code	Description
SH1	SH1	Source Handshake
AH1	AH1	Acceptor Handshake
T6, TE6	T6, TE0	Talker, Extended Talker
L4, LE4	L4, LE0	Listener, Extended Listener
SR1	SR1	Service Request
RL1	RL0	Remote/Local
PP1, PP2	PP1	Parallel Poll
DC1	DC1	Device Clear
DT1	DT0	Device Trigger
C1, C2, C3, C4, C5	C0	Controller
E1, E2	E1, E2	Three-state bus drivers with automatic switch to open collector during parallel poll

B

Multiline Interface Messages

This appendix lists the multiline interface messages and describes the mnemonics and messages that correspond to the interface functions.

The multiline interface messages are commands defined by the IEEE 488 standard. The messages are sent and received with ATN asserted. The interface functions include initializing the bus, addressing and unaddressing devices, and setting device modes for local or remote programming. For more information about these messages, refer to the ANSI/IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*.

Table B-1. Multiline Interface Messages

Hex	Dec	ASCII	Message
00	0	NUL	—
01	1	SOH	GTL
02	2	STX	—
03	3	ETX	—
04	4	EOT	SDC
05	5	ENQ	PPC
06	6	ACK	—
07	7	BEL	—
08	8	BS	GET
09	9	HT	TCT
0A	10	LF	—
0B	11	VT	—
0C	12	FF	—
0D	13	CR	—
0E	14	SO	—
0F	15	SI	—
10	16	DLE	—
11	17	DC1	LLO
12	18	DC2	—
13	19	DC3	—
14	20	DC4	DCL
15	21	NAK	PPU
16	22	SYN	—
17	23	ETB	—
18	24	CAN	SPE
19	25	EM	SPD
1A	26	SUB	—
1B	27	ESC	—
1C	28	FS	—
1D	29	GS	—
1E	30	RS	—
1F	31	US	CFE

Hex	Dec	ASCII	Message
20	32	SP	MLA0
21	33	!	MLA1
22	34	"	MLA2
23	35	#	MLA3
24	36	$	MLA4
25	37	%	MLA5
26	38	&	MLA6
27	39	'	MLA7
28	40	(	MLA8
29	41	)	MLA9
2A	42	*	MLA10
2B	43	+	MLA11
2C	44	,	MLA12
2D	45	-	MLA13
2E	46	.	MLA14
2F	47	/	MLA15
30	48	0	MLA16
31	49	1	MLA17
32	50	2	MLA18
33	51	3	MLA19
34	52	4	MLA20
35	53	5	MLA21
36	54	6	MLA22
37	55	7	MLA23
38	56	8	MLA24
39	57	9	MLA25
3A	58	:	MLA26
3B	59	;	MLA27
3C	60	<	MLA28
3D	61	=	MLA29
3E	62	>	MLA30
3F	63	?	UNL

Table B-1. Multiline Interface Messages (Continued)

Hex	Dec	ASCII	Message
40	64	@	MTA0
41	65	A	MTA1
42	66	B	MTA2
43	67	C	MTA3
44	68	D	MTA4
45	69	E	MTA5
46	70	F	MTA6
47	71	G	MTA7
48	72	H	MTA8
49	73	I	MTA9
4A	74	J	MTA10
4B	75	K	MTA11
4C	76	L	MTA12
4D	77	M	MTA13
4E	78	N	MTA14
4F	79	O	MTA15
50	80	P	MTA16
51	81	Q	MTA17
52	82	R	MTA18
53	83	S	MTA19
54	84	T	MTA20
55	85	U	MTA21
56	86	V	MTA22
57	87	W	MTA23
58	88	X	MTA24
59	89	Y	MTA25
5A	90	Z	MTA26
5B	91	[	MTA27
5C	92	\	MTA28
5D	93	]	MTA29
5E	94	^	MTA30
5F	95	_	UNT

Hex	Dec	ASCII	Message
60	96	`	MSA0, PPE
61	97	a	MSA1, PPE, CFG1
62	98	b	MSA2, PPE, CFG2
63	99	c	MSA3, PPE, CFG3
64	100	d	MSA4, PPE, CFG4
65	101	e	MSA5, PPE, CFG5
66	102	f	MSA6, PPE, CFG6
67	103	g	MSA7, PPE, CFG7
68	104	h	MSA8, PPE, CFG8
69	105	i	MSA9, PPE, CFG9
6A	106	j	MSA10, PPE, CFG10
6B	107	k	MSA11, PPE, CFG11
6C	108	l	MSA12, PPE, CFG12
6D	109	m	MSA13, PPE, CFG13
6E	110	n	MSA14, PPE, CFG14
6F	111	o	MSA15, PPE, CFG15
70	112	p	MSA16, PPD
71	113	q	MSA17, PPD
72	114	r	MSA18, PPD
73	115	s	MSA19, PPD
74	116	t	MSA20, PPD
75	117	u	MSA21, PPD
76	118	v	MSA22, PPD
77	119	w	MSA23, PPD
78	120	x	MSA24, PPD
79	121	y	MSA25, PPD
7A	122	z	MSA26, PPD
7B	123	{	MSA27, PPD
7C	124	|	MSA28, PPD
7D	125	}	MSA29, PPD
7E	126	~	MSA30, PPD
7F	127	DEL	—

Multiline Interface Message Definitions			
CFE †	Configuration Enable	PPD	Parallel Poll Disable
CFG †	Configure	PPE	Parallel Poll Enable
DCL	Device Clear	PPU	Parallel Poll Unconfigure
GET	Group Execute Trigger	SDC	Selected Device Clear
GTL	Go To Local	SPD	Serial Poll Disable
LLO	Local Lockout	SPE	Serial Poll Enable
MLA	My Listen Address	TCT	Take Control
MSA	My Secondary Address	UNL	Unlisten
MTA	My Talk Address	UNT	Untalk
PPC	Parallel Poll Configure		

† This multiline interface message is a proposed extension to the IEEE 488 specification to support the HS488 protocol.

C

Status and Error Message Information

This appendix describes the status and error information that the GPIB-232/485CT-A records as it executes each programming message. This information is returned in response to the `stat` command, or automatically after each command if continuous status reporting is enabled.

Status is recorded in four parts: the 16-bit status word (`status`), the GPIB error indicator (`GPIB-error`), the serial error indicator (`serial-error`), and a count of the last I/O performed (`count`). For more information, see the `stat` function description for S mode or G mode.

Status Bits

The following paragraphs describe the conditions represented by the bits in status. The number preceding each description is the numeric value of that bit in the status word.

ERR S or G mode –32768

The ERR bit is set in `status` following any call that results in an error. The particular error can be determined by examining the `GPIB-error` and `serial-error` values. The ERR bit is cleared following any call that does not result in an error.

By examining this bit, you can check for an error condition after each call. An error made early in your application program might not become apparent until a later instruction. At that time, the error can be more difficult to locate.

TIMO S mode 16384

The TIMO bit specifies whether a timeout has occurred. The TIMO bit is set in the status word following a call to `wait` if the TIMO bit of the wait mask parameter is also set and if the wait has exceeded the time limit value that is set by the `tmo` call. The TIMO bit is also set following a call to any of the I/O

functions (for example, `rd`, `wrt`, and `cmd`), if a timeout occurs during a call. The TIMO bit is cleared in the status word in all other circumstances.

END S mode 8192

The END bit specifies whether the END or EOS message has been received. The END bit is set in the status word following a `rd` function if the END or EOS message was detected during the read. While the GPIB-232/485CT-A is performing a shadow handshake as a result of the `gts` function, any other function call can return a status word with the END bit set if the END or EOS message occurred before or during that call. The END bit is cleared in the status word at the start of any subsequent programming message.

SRQI S mode 4096

The SRQI bit specifies whether a device is requesting service. This bit is set in the status word whenever the **SRQ*** line is asserted. The bit is cleared whenever the GPIB **SRQ*** line is unasserted.

CMPL S or G mode 256

The CMPL bit specifies that the operation relating to this status information is complete. This bit is always set, and is useful in identifying the status word from other responses.

LOK S mode 128

The LOK bit specifies whether the GPIB-232/485CT-A is in a lockout state. The LOK bit is set whenever the GPIB-232/485CT-A detects the Local Lockout (LLO) message has been sent either by the GPIB-232/485CT-A or by another Controller. The LOK bit is cleared when the Remote Enable (**REN***) GPIB line becomes unasserted. A call to `onl` also clears LOK.

REM S mode 64

The REM bit specifies whether the GPIB-232/485CT-A is in remote state. The REM bit is set whenever the Remote Enable (**REN***) GPIB line is asserted and the GPIB-232/485CT-A detects its listen address has been sent either by the GPIB-232/485CT-A or by another Controller. The REM bit is cleared whenever **REN*** becomes unasserted, or when the GPIB-232/485CT-A as a Listener detects the Go to Local

(GTL) command, or when the `loc` function is called while the LOK bit is cleared in status. A call to `onl` also clears REM.

CIC S mode 32

The CIC bit specifies whether the GPIB-232/485CT-A is the Controller-In-Charge. The CIC bit is set whenever `sic` is called while the GPIB-232/485CT-A is System Controller, or when another Controller passes control to the GPIB-232/485CT-A. The CIC bit is cleared whenever the GPIB-232/485CT-A detects Interface Clear (**IFC***) from some other device that is System Controller, or when the GPIB-232/485CT-A passes control to another device. A call to `onl` also clears CIC.

ATN S mode 16

The ATN bit specifies the state of the GPIB Attention (**ATN***) line. The ATN bit is set whenever the GPIB **ATN*** line is asserted and cleared when the **ATN*** line is unasserted.

TACS S mode 8

The TACS bit specifies whether the GPIB-232/485CT-A has been addressed as a Talker. The TACS bit is set whenever the GPIB-232/485CT-A detects that its talk address (and secondary address, if enabled) has been sent either by the GPIB-232/485CT-A itself or by another Controller. The TACS bit is cleared whenever the GPIB-232/485CT-A detects the Untalk (UNT) command, a talk address other than its own, its own listen address, or Interface Clear (**IFC***). A call to `onl` also clears TACS.

LACS S mode 4

The LACS bit specifies whether the GPIB-232/485CT-A has been addressed as a Listener. The LACS bit is set whenever the GPIB-232/485CT-A detects that its listen address (and secondary address, if enabled) has been sent either by the GPIB-232/485CT-A itself or by another Controller. The LACS bit is also set whenever the GPIB-232/485CT-A shadow handshakes as a result of the `gts` function. The LACS bit is cleared whenever the GPIB-232/485CT-A detects that Unlisten (UNL), its own talk address, Interface Clear (**IFC***), or `gts` is called without shadow handshake. A call to `onl` also clears LACS.

DTAS S mode 2

The DTAS bit specifies whether the GPIB-232/485CT-A has detected a device trigger command. The DTAS bit is set whenever the GPIB-232/485CT-A as a Listener, detects that another Controller has sent the Group Execute Trigger (GET) command. The DTAS bit is cleared in status at the start of any subsequent programming message.

DCAS S mode 1

The DCAS bit specifies whether the GPIB-232/485CT-A has detected a device clear command. The DCAS bit is set whenever the GPIB-232/485CT-A detects the Device Clear (DCL) command has been sent by another Controller, or whenever the GPIB-232/485CT-A as a Listener detects the Selected Device Clear (SDC) command has been sent by another Controller. The DCAS bit is cleared in status at the start of any subsequent programming message.

GPIB Error Codes

When the ERR bit is set in status, a GPIB error or a serial port error has occurred. The error code is indicated by `GPIB-error` or `serial-error`.

The following paragraphs describe the `GPIB-errors` in detail. The number preceding each description is the numeric value of the error code.

NGER S or G mode 0

The GPIB-232/485CT-A reports this error when GPIB-232/485CT-A detected no GPIB errors as a result of the last operation.

ECIC S mode 1

The GPIB-232/485CT-A records this value when you call a function that requires that the GPIB-232/485CT-A be CIC and it is not CIC.

In cases when the GPIB-232/485CT-A should always be the Controller-In-Charge, call `sic` to send Interface Clear before attempting any of the calls, and avoid sending the command byte TCT (hex 09, Take Control). In multiple CIC situations, always be certain that the CIC bit appears in status before attempting the calls. If it does not appear, you can call `wait`

(256) to delay further processing until control is passed to the GPIB-232/485CT-A.

ENOL S mode 2

The ENOL error occurs most frequently when the GPIB-232/485CT-A attempts to write data to the GPIB and no Listeners are addressed.

To correct this error, be sure that the proper listen address is in the `alist` argument string, use `cmd` to properly address the Listeners, or be sure some other Controller has addressed the Listeners before you call `wrt`.

This error might occur less frequently in situations in which the GPIB-232/485CT-A is not the CIC and the Controller asserts **ATN*** before the `wrt` call in progress has ended. In this case, either reduce the write byte count to that which is expected by the Controller, or resolve the situation on the Controller's end.

This error also occurs during `cmd` if there is no device on the GPIB bus receiving the command bytes. In this case, check the GPIB cabling and verify that the attached GPIB devices are powered on.

EADR S mode 3

The GPIB-232/485CT-A records this error when it is not addressed to listen or to talk before `rd` and `wrt` calls when it is not CIC. Be sure that the Controller addresses the GPIB-232/485CT-A to talk or listen before attempting the `wrt` or `rd`.

The GPIB-232/485CT-A also records this error during the function `gts` when the shadow handshake feature is requested and the GPIB **ATN*** line is already unasserted. In this case, the shadow handshake is not possible and the error is recorded to notify you of that fact. `gts` should almost never be called except immediately after a `cmd` call. (`cmd` causes **ATN*** to be asserted.)

EARG S or G mode 4

The GPIB-232/485CT-A records this error when you pass an invalid argument to a function call. The following are some examples:

- `tmo` called with a value not in the range .00001 to 3600
- `sic` called with a value not in the range .0001 to 3600
- `eos` called with meaningless termination method identifiers
- `caddr` called with the value 31
- `ppc` called with illegal parallel poll configurations

If your programming message contains more than one argument and you get this error, the GPIB-232/485CT-A discards all arguments and does not perform the function.

This error can also be caused by a transmission error which corrupts the argument portion of the programming message or that corrupts the <CR> or <LF> that terminates which programming message. Use `stat` and check `serial-error` to determine if a transmission error has occurred.

ESAC S mode 5

The GPIB-232/485CT-A records this error when `sic` or `sre` is called when the GPIB-232/485CT-A does not have System Controller capability. In this case, give the GPIB-232/485CT-A Controller capability by calling `rsc`. (At power-on, the GPIB-232/485CT-A assumes itself to be the System Controller.)

EABO S mode 6

The GPIB-232/485CT-A records this error when I/O has been canceled. The cause of this error is usually a timeout condition.

If I/O is actually progressing but times out anyway, lengthen the timeout period with `tmo`. More frequently, however, either the Listener is not continuing to handshake, the Talker has stopped talking, or the byte count in the call that timed out was more than the other device was expecting. Be sure that both parties to the transfer understand what byte count is expected. Or if possible, have the Talker use the END message to assist in early termination.

ECAP S or G mode 11

This error results when a particular capability has been disabled in the GPIB-232/485CT-A and a call is made that attempts to make use of that capability.

A common cause of this error is that a programming message contains an S mode function and the GPIB-232/485CT-A is configured for G mode, or a programming message contains a G mode function and the GPIB-232/485CT-A is configured for S mode.

EBUS S mode 14

This error indicates that there was a problem sending command bytes out of the GPIB port. The most common causes of this error are either that the bytes could not be sent out within the timeout period, or that there was not a device on the GPIB bus to receive the command bytes. This error can occur during `clr`, `loc`, `pct`, `ppc`, `ppu`, `rd`, `rsp`, `trg`, or `wrt`.

ECMD S or G mode 17

The GPIB-232/485CT-A records this error when your programming message received does not contain a recognizable function name. This error can happen if the function name is misspelled or if a transmission error occurred that resulted in the function name being corrupted. Check your function name spelling, and check `serial-error` to see if a serial port error has been posted.

Serial Port Error Codes

The following paragraphs describe the serial port errors in detail. The number preceding each description is the numeric value of the error code.

In S mode, when a serial port error occurs as the GPIB-232/485CT-A receives a programming message, the error is recorded and the message is discarded. If a serial port error occurs during a data transfer such as `cmd`, `rd`, or `wrt`, that transfer is aborted. You can use the `spign` function to tell the GPIB-232/485CT-A to ignore all serial port errors. When serial port errors are ignored, bytes that arrive with errors are not discarded, and no error is recorded in the serial port error code returned by `stat`.

NSER 0

The GPIB-232/485CT-A reports this value when there is no serial port error detected as a result of the last operation.

EPAR 1

The GPIB-232/485CT-A records this error when the parity of the received character is not what was expected; one or more bits of the received character were corrupted in a way that changed the character's parity.

EORN 2

The GPIB-232/485CT-A records this error when characters arrive at the serial port faster than the serial port can accept them. When this error occurs, one or more characters sent to the serial port have been lost. If this error occurs, check to see that the GPIB-232/485CT-A and your serial device are using the same serial port settings.

EOFL 3

The GPIB-232/485CT-A records this error when the GPIB-232/485CT-A's internal serial port buffer overflows. This error should only occur if XON/XOFF is disabled and no hardware handshake is in effect.

EFRM 4

The GPIB-232/485CT-A records this error when a character is received whose stop bits are not in the expected place. This error can happen when the number-of-bits-per-character setting of the GPIB-232/485CT-A does not match your serial device. It can also happen if the baud rates of the GPIB-232/485CT-A and your serial device do not match, or if one side of the serial link does not use parity and the other side does.

The GPIB-232/485CT-A also records this error when it receives a serial break from the attached serial device. The reception of a serial break is detected when the Receive Data (RXD) line is unasserted for longer than a full word transmission time (that is, the total time of start bit + data bits + parity + stop bits).

D

Interfacing to an RS-232 Device

This appendix describes the RS-232 serial port on the GPIB-232CT-A and explains how to interface a DCE or DTE serial device to the RS-232 serial port.

The GPIB-232CT-A transfers serial data using the electrical signals, mechanical connections, data format, and control protocols defined in the ANSI/EIA-232-D (RS-232) standard. The RS-232 port on the GPIB-232CT-A provides an asynchronous serial communication link to a serial peripheral device.

RS-232 Standard

RS-232, as specified in the ANSI/EIA-232-D Standard, *Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*, standardizes serial communication between computers and between computer terminals and modems. Most applications use the RS-232 standard for interfacing peripherals to personal computers. RS-232 uses transmission lines in which the state of each signal is represented by referencing the voltage level of a single line to ground. RS-232 was designed for serial communication up to distances of 50 ft and with data rates up to 20 kbytes/s. However, because of improvements in line drivers and cabling, you can often increase the actual performance of the bus past the limitations on speed and distance recommended in the specification. For more information on the RS-232 standard, contact the following organization:

Global Engineering Documents
7730 Carondelet Avenue, Suite 4007
St. Louis, MO 63105
(800) 854-7179

Description of the RS-232 Port

The RS-232 serial port on the GPIB-232CT-A uses a male 9-pin D-Subminiature connector with a DTE interface configuration. Table D-1 shows the signal lines supported on the GPIB-232CT-A.

Table D-1. RS-232 Serial Port Signal Configuration

Pin Number	Signal Description	RS-232 Code	Function
2	RXD (Receive Data)	BB	This signal carries serial data from the serial device to the GPIB-232CT-A.
3	TXD (Transmit Data)	BA	This signal carries serial data from the GPIB-232CT-A to the serial device.
4	DTR (Data Terminal Ready)	CD	This signal is asserted by the GPIB-232CT-A to signal that it has been powered on and is ready to operate.
5	GND (Ground)	AB	This signal establishes a reference point for all interface voltages.
7	RTS (Request to Send)	CA	This signal is driven by the GPIB-232CT-A. When asserted, it indicates that the GPIB-232CT-A is ready to accept serial data. When unasserted, it indicates that the GPIB-232CT-A is no longer ready to accept serial data because the buffer is full.
8	CTS (Clear to Send)	CB	This signal is sensed by the GPIB-232CT-A. When asserted, it indicates that the serial device is ready to accept serial data. When unasserted, it indicates that data transmission should be disabled.

DTE vs. DCE

Data Terminal Equipment (DTE) and Data Communications Equipment (DCE) were the terms used in the RS-232 specification for the types of equipment on either end of a serial connection. (A DCE is now called Data Circuit-Terminating Equipment in Revision D of the RS-232 specification.) In general, DTE and DCE refer to computer equipment and modems respectively. Because the RS-232 specification mainly involved connecting

a DTE directly to a DCE and vice versa, the pinouts were defined so that cabling was simple. That is, a cable connected a computer to a modem by wiring pin 1 to pin 1, pin 2 to pin 2, and so on. This method is commonly known as *straight-through* cabling.

Figure D-1 shows straight-through cabling in a DTE-to-DCE interface.

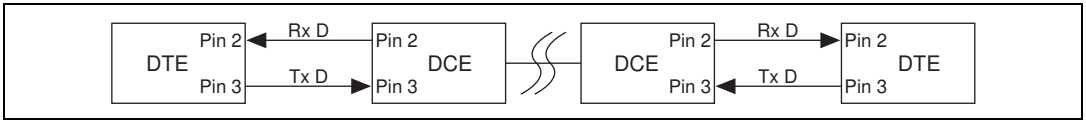


Figure D-1. Straight-Through Cabling in a DTE-to-DCE Interface

Straight-through cabling is still the standard method to connect a modem to your PC. However, because many applications use serial communication to connect two or more DTEs without modems, the cabling becomes more complicated. If two DTEs were wired together using a straight-through cable, one transmitter would be connected to the other transmitter, and one receiver would be connected to the other receiver. In this setup, no transmissions could occur. Thus, these applications must use a cabling scheme that connects the transmitter on one device to the receiver on the other device and vice versa. This method is known as *null-modem* cabling, because it replaces the two modems that traditional RS-232 applications would require between the two DTEs. You should use a null-modem cable to communicate from one AT serial port to another. Figure D-2 shows null-modem cabling in a DTE-to-DCE interface.

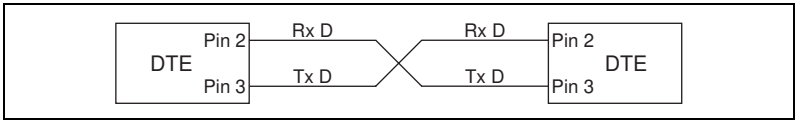


Figure D-2. Null-Modem Cabling in a DTE-to-DCE Interface

Interfacing Serial Devices to the RS-232 Serial Port

To interface other serial devices to the RS-232 serial port on the GPIB-232CT-A, first refer to the manual that came with your serial device to determine if the device is configured as a DTE or DCE. Also, determine from the manual how the control lines are used and whether they must be driven for the serial port to operate.

Note The GPIB-232CT-A serial port is configured to be a DTE.

Figure D-3 shows the location of the RS-232 connector.

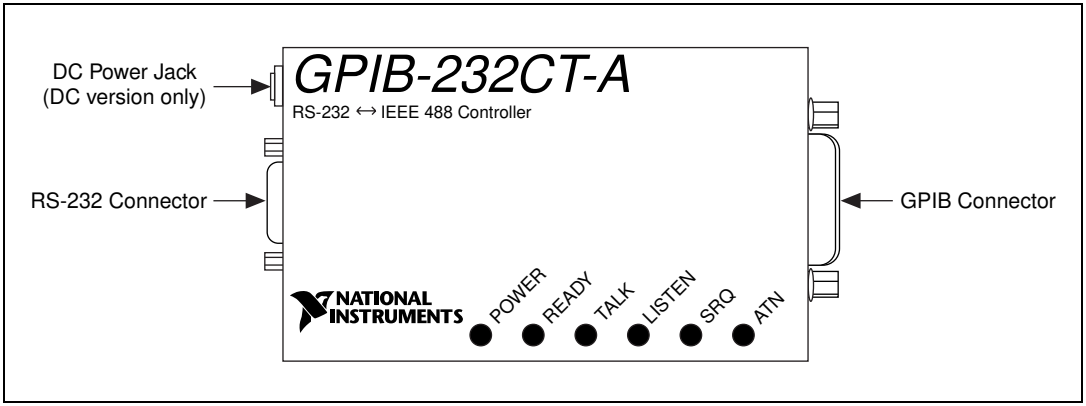


Figure D-3. Location of the RS-232 Connector

Interfacing the GPIB-232CT-A to a DCE with Handshaking

A correctly configured DTE-to-DCE interface is wired *straight across*: GPIB-232CT-A TXD to DCE TXD, GPIB-232CT-A RXD to DCE RXD, and so on as shown in Table D-2.

Table D-2. Cable Wiring Scheme for GPIB-232CT-A DTE to Serial Device DCE

GPIB-232CT-A Signal to DCE Serial Device Signal	Standard DTE to DCE 9-pin 9-pin	Standard DTE to DCE 9-pin 25-pin
RXD to RXD [1]	***2 to 2***	***2 to 3***
TXD to TXD [1]	***3 to 3***	***3 to 2***
DTR to DTR	4 to 4	4 to 20
GND to GND [1]	***5 to 5***	***5 to 7***
RTS to RTS	7 to 7	7 to 4
CTS to CTS	8 to 8	8 to 5

[1] The connections must be implemented.

Figure D-4 shows a properly configured 9-pin DTE to 9-pin DCE cable including the hardware handshake lines RTS, CTS, and DTR. With this configuration, the GPIB-232CT-A can function properly (handshake) on

buffer full conditions. Figure D-5 shows an equivalent 9-pin DTE to 25-pin DCE cable configuration.

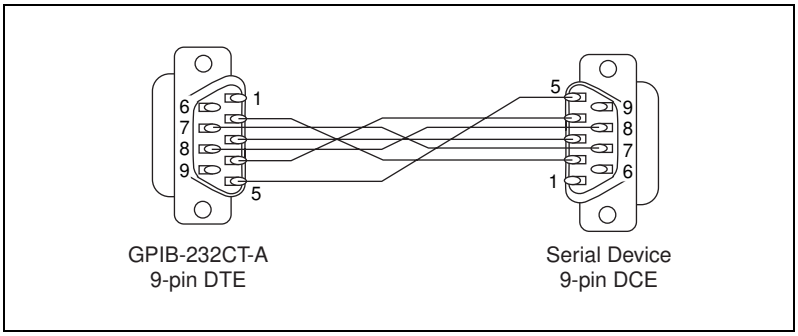


Figure D-4. Cable Configuration for 9-pin DTE to 9-pin DCE with Handshaking

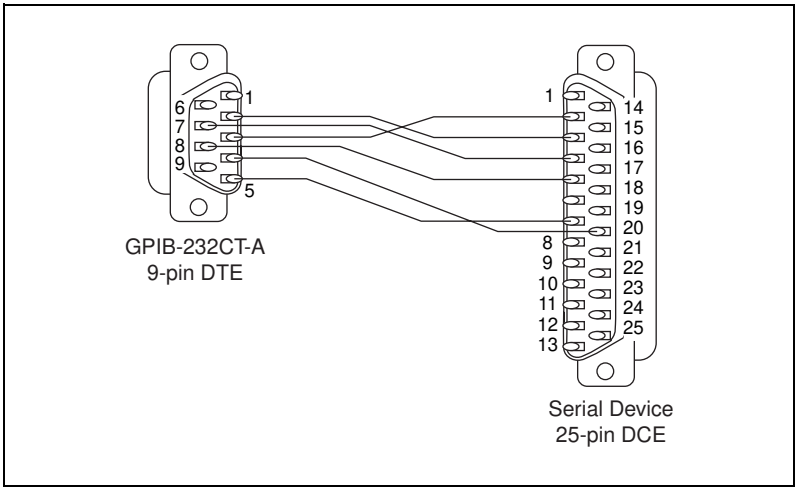


Figure D-5. Cable Configuration for 9-pin DTE to 25-pin DCE with Handshaking

Interfacing the GPIB-232CT-A to a DCE without Handshaking

If your serial device does not require or use the hardware handshaking protocol used by the GPIB-232CT-A, you have two options:

- Use a minimum configuration cable which does not support hardware handshaking and use XON/XOFF software handshaking (if necessary). To create a minimum configuration cable, connect the signals shown in bold italics in Table D-2.

- Wire a custom cable that properly interfaces the GPIB-232CT-A hardware handshaking protocol to the handshaking protocol of your serial device.

Minimum Configuration Cable

The minimum configuration cable assumes that the DCE does not require external hardware handshaking. The minimum configuration for a DTE 9-pin to DCE 9-pin cable is shown in Figure D-6. Figure D-7 shows an equivalent 9-pin to 25-pin cable.

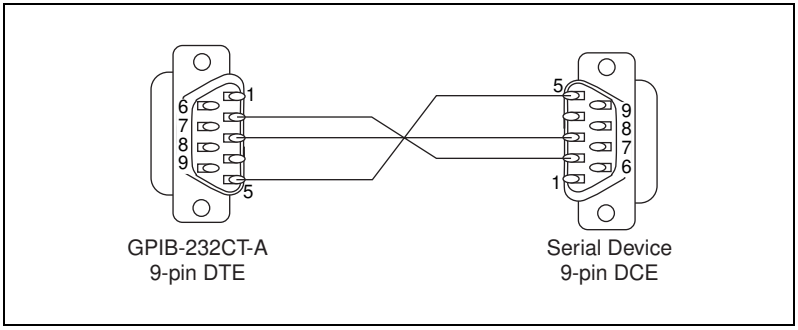


Figure D-6. Minimum Configuration for 9-pin DTE to 9-pin DCE

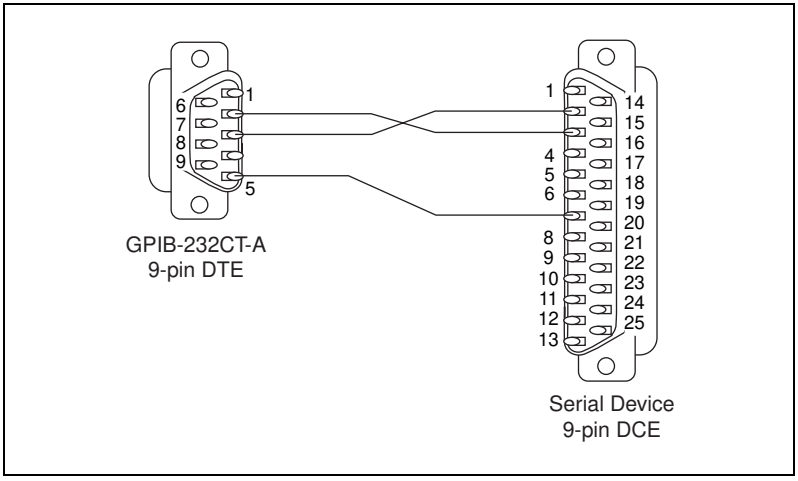


Figure D-7. Minimum Configuration for 9-pin DTE to 25-pin DCE

Custom Cables

If your application requires a custom cable, you can construct one if you have a thorough knowledge of the handshaking protocols involved. Review the RS-232 characteristics of your serial device and build the cable to properly connect the handshake lines of the two devices. Because the DTE-to-DCE connection is a straight across connection, it often involves only connecting RTS to RTS, CTS to CTS, DTR to DTR, and so on. If the documentation for your serial device does not provide a thorough explanation of its handshaking protocol, the ANSI/EIA-232-D standard is a good reference, provided your device conforms to the RS-232 protocol.

Caution Although handshaking might not be required, it is best to use some form of handshaking to prevent loss of data.

Interfacing the GPIB-232CT-A to a DTE with Handshaking

For serial devices set up as DTEs, you must wire a DTE-to-DTE interface cable, commonly known as a null modem cable. The cable allows the GPIB-232CT-A to act as though it is communicating with a DCE, but it swaps the appropriate pins to achieve a DTE configuration. This wiring configuration is shown in Table D-3.

Table D-3. Cable Wiring Scheme for GPIB-232CT-A DTE to Serial Device DTE

GPIB-232CT-A Signal to DTE Serial Device Signal	Standard DTE to DTE 9-pin 9-pin	Standard DTE to DTE 9-pin 25-pin
RXD to TXD [1]	***2 to 3***	***2 to 2***
TXD to RXD [1]	***3 to 2***	***3 to 3***
DTR to DSR	4 to 6	4 to 6
GND to GND [1]	***5 to 5***	***5 to 7***
RTS to CTS	7 to 8	7 to 5
CTS to RTS	8 to 7	8 to 4

[1] The connections must be implemented.

Figure D-8 shows a typical 9-pin to 9-pin null modem cable with the RTS, CTS, and DTR handshake lines implemented. Figure D-9 shows an equivalent 9-pin DTE to 25-pin DTE cable configuration.

The cable configuration in Figure D-8 allows you to connect to the 9-pin serial port of a personal computer. The cable in Figure D-9 shows how to connect to the 25-pin serial port of a personal computer.

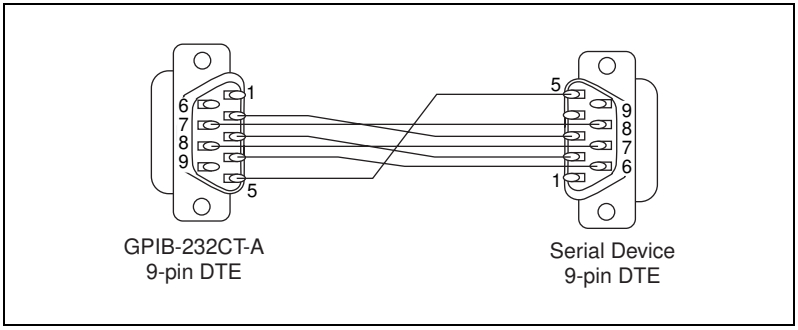


Figure D-8. Cable Configuration for 9-pin DTE to 9-pin DTE with Handshaking

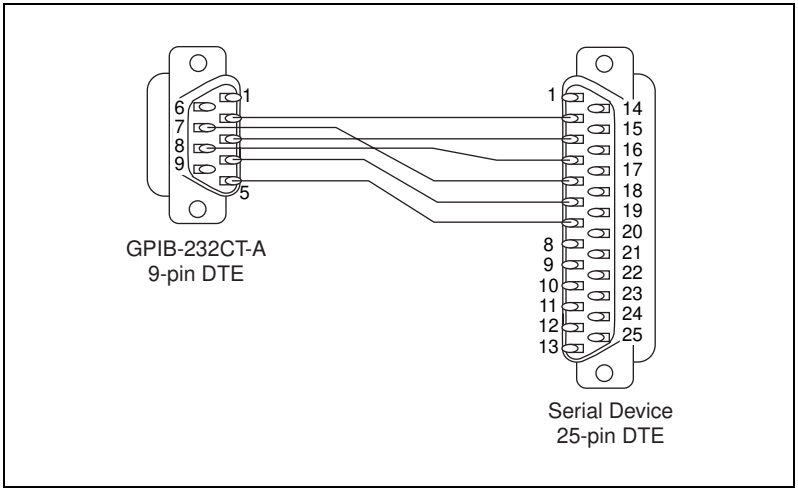


Figure D-9. Cable Configuration for 9-pin DTE to 25-pin DTE with Handshaking

Interfacing the GPIB-232CT-A to a DTE without Handshaking

If your serial device does not require or use the same hardware handshaking protocol used by the GPIB-232CT-A, you have two options:

- Use a minimum configuration null modem cable which does not support the hardware handshake lines and use XON/XOFF software handshaking (if necessary).

  To create a minimum configuration cable, connect the signals shown in bold italics in Table D-3.

- Wire a custom cable that properly interfaces the GPIB-232CT-A hardware handshaking protocol to the handshaking protocol of your serial device.

Minimum Configuration Cable

The minimum cable configuration assumes that the serial device does not require external hardware handshaking. The minimum configuration for a 9-pin to 9-pin null modem cable is shown in Figure D-10. Figure D-11 shows an equivalent 9-pin to 25-pin cable.

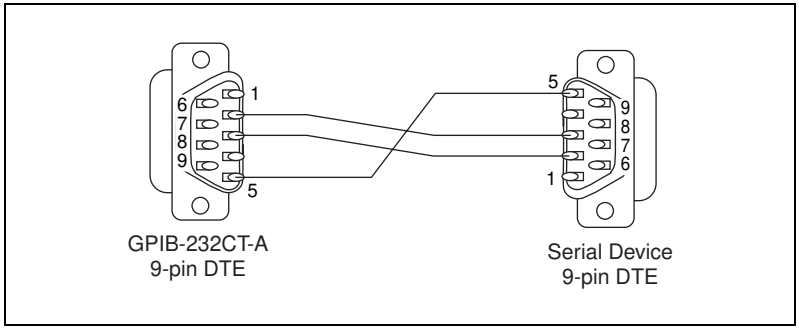


Figure D-10. Minimum Configuration for 9-pin DTE to 9-pin DTE

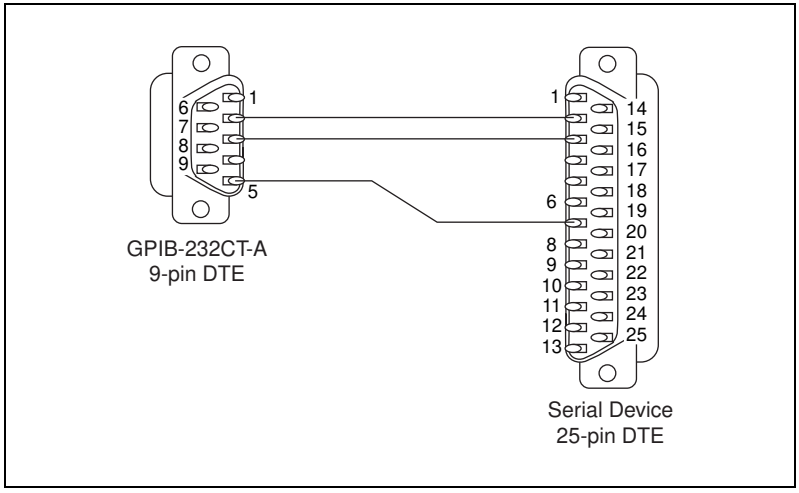


Figure D-11. Minimum Configuration for 9-pin DTE to 25-pin DTE

Custom Cables

If your application requires a custom cable, you can construct one if you have a thorough knowledge of the handshaking protocols involved. Review the RS-232 characteristics of your serial device and build the cable to properly connect the handshake lines of the two devices. If the documentation for your serial device does not provide a thorough explanation of its handshaking protocol, the ANSI/EIA-232-D standard is a good reference, provided your device conforms to the RS-232 protocol.

Caution Although handshaking might not be required, it is best to use some form of handshaking to prevent loss of data.

E

Interfacing to an RS-485 Device

This appendix describes the RS-485 serial port on the GPIB-485CT-A and explains how to interface an RS-485 device to the RS-485 serial port.

The GPIB-485CT-A transfers serial data using the electrical signals defined in the EIA-485 standard. The RS-485 port on the GPIB-485CT-A provides an asynchronous serial communication link to a serial peripheral device. For more information on the EIA-485 standard, contact the following organization:

Global Engineering Documents
7730 Carondelet Avenue, Suite 4007
St. Louis, MO 63105
(800) 854-7179

RS-422

RS-422, as specified in the EIA/RS-422-A Standard, *Electrical Characteristics of Balanced Voltage Digital Interface Circuits*, defines a serial interface much like RS-232. However, RS-422 uses balanced (or differential) transmission lines. Balanced transmission lines use two transmission lines for each signal. The state of each signal is represented, not by a voltage level on one line referenced to ground as in RS-232, but rather by the relative voltage of the two lines to each other. For example, the TX signal is carried on two wires, wire A and wire B. A logical 1 is represented by the voltage on line A being greater than the voltage on line B. A logical 0 is represented by the voltage on line A being less than the voltage on line B. Differential voltage transmission creates a signal that is much more immune to noise as well as voltage loss due to transmission line effects. Thus, you can use RS-422 for much longer distances (up to 4,000 ft) and much greater transmission speeds (up to 10 Mbytes/s) than RS-232.

RS-485

RS-485, as specified in the EIA-485 Standard, *Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*, expands on the RS-422 standard by increasing the number of devices you can use from 10 to 32 and by working with half-duplex bus architectures. Unlike the RS-422 standard, RS-485 addresses the issue of using multiple transmitters on the same line. RS-485 defines the electrical characteristics necessary to ensure adequate signal voltages under maximum load, short circuit protection, and the ability to withstand multiple drivers driving conflicting signals at the same time.

Description of the RS-485 Port

The RS-485 serial port on the GPIB-485CT-A uses a male 9-pin D-Subminiature connector. Figure E-1 shows the pin locations on the connector. Table E-1 shows the signal lines supported on the GPIB-485CT-A.

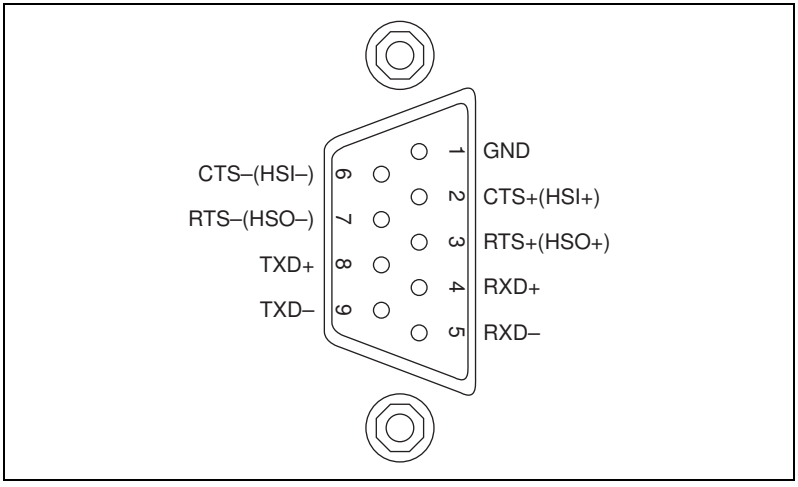


Figure E-1. Male DB-9 Connector Pin Locations

Table E-1. RS-485 Serial Port Signal Configuration

Pin Number	Signal Description	Function
1	GND (Ground)	This signal establishes a reference point for all interface voltages.
2	CTS+ (HSI+) (Handshake Input, Non-inverting)	This signal is sensed by the GPIB-485CT-A. CTS+ can be used together with CTS– to control serial data transmission from the GPIB-485CT-A.
3	RTS+ (HSO+) (Handshake Output, Non-inverting)	This signal is driven by the GPIB-485CT-A. RTS+ is used together with RTS– to control serial data transmission from the serial device. This signal is asserted when the internal buffer on the GPIB-485CT-A is full.
4	RXD+ (Receive Data, Non-inverting)	This signal, combined with RXD–, carries data from the serial device to the GPIB-485CT-A.
5	RXD– (Receive Data, Inverting)	This signal, combined with RXD+, carries data from the serial device to the GPIB-485CT-A.
6	CTS– (HSI–) (Handshake Input, Inverting)	This signal is sensed by the GPIB-485CT-A. CTS– can be used together with CTS+ to control serial data transmission from the GPIB-485CT-A.
7	RTS– (HSO–) (Handshake Output, Inverting)	This signal is driven by the GPIB-485CT-A. RTS– is used together with RTS+ to control serial data transmission from the serial device. This signal is asserted when the internal buffer on the GPIB-485CT-A is full.
8	TXD+ (Transmit Data, Non-inverting)	This signal, combined with TXD–, carries data from the GPIB-485CT-A to the serial device.
9	TXD– (Transmit Data, Inverting)	This signal, combined with TXD+, carries data from the GPIB-485CT-A to the serial device.

Interfacing Serial Devices to the RS-485 Serial Port

The GPIB-485CT-A is a full-duplex, point-to-point device. Separate wires must be connected for the TXD, RXD, CTS, and RTS lines. For example, to connect the GPIB-485CT-A to a National Instruments AT-485 serial interface, you would need to make the connections shown in Table E-2.

Table E-2. Cable Wiring Scheme for GPIB-485CT-A to AT-485 Serial Interface

GPIB-485CT-A Signal to AT-485 Signal			GPIB-485CT-A 9-pin	AT-485 9-pin
GND	to	GND	1	1
CTS+	to	RTS+	2	3
RTS+	to	CTS+	3	4
RXD+	to	TXD+ [1]	4	8
RXD–	to	TXD– [1]	5	9
CTS–	to	RTS–	6	7
RTS–	to	CTS–	7	6
TXD+	to	RXD+ [1]	8	4
TXD–	to	RXD– [1]	9	5

[1] The connections must be implemented.

Before attempting to build a custom cable, carefully review the RS-485 characteristics of your serial device. You should build the cable to implement handshaking if possible.

Caution Although handshaking might not be required, it is best to implement some form of handshaking to prevent loss of data.

Termination

Because each differential pair of wires is a transmission line, you must properly terminate the line to prevent reflections. A common method of terminating a two-wire multidrop RS-485 network is to install terminating resistors at each end of the multidrop network. The terminating resistor should match the characteristic impedance of the transmission line (typically 100–120 Ω). National Instruments offers an optional DB-9

RS-485 termination connector that contains embedded terminating resistors for easy termination.

Figure E-2 shows a full-duplex system with termination.

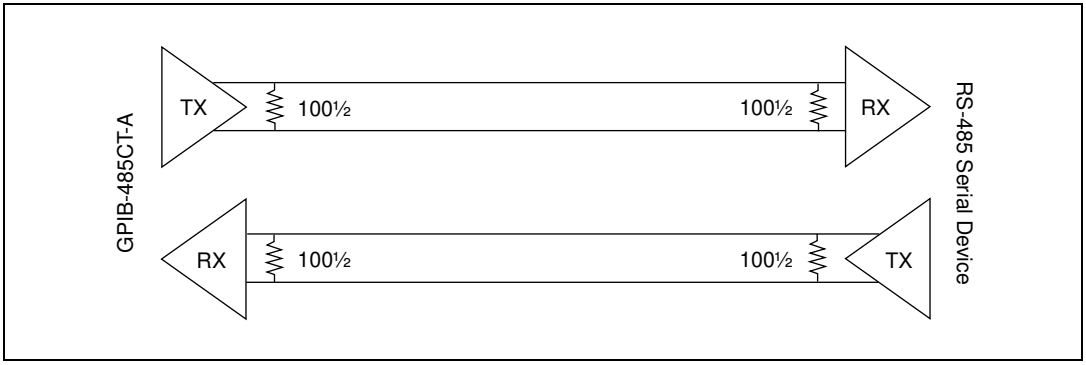


Figure E-2. Point-to-Point Network Using Terminating Resistors

F

GPIB Basics

This appendix contains information about how the GPIB operates, including information about GPIB messages, the roles of Talkers, Listeners, and Controllers, GPIB signals and lines, and connecting and configuring devices on the bus.

The ANSI/IEEE Standard 488.1-1987, also known as GPIB (General Purpose Interface Bus), describes a standard interface for communication between instruments and controllers from various vendors. It contains information about electrical, mechanical, and functional specifications. The GPIB is a digital, 8-bit parallel communications interface with data transfer rates of 1 Mbyte/s and above. The bus supports one System Controller, usually a computer, and up to 14 additional instruments. The ANSI/IEEE Standard 488.2-1992 extends IEEE 488.1 by defining a bus communication protocol, a common set of data codes and formats, and a generic set of common device commands.

Types of GPIB Messages

The GPIB carries device-dependent messages and interface messages.

- *Device-dependent messages*, often called *data* or *data messages*, contain device-specific information such as programming instructions, measurement results, machine status, and data files.
- *Interface messages* manage the bus. They are usually called *commands* or *command messages*. Interface messages perform such tasks as initializing the bus, addressing and unaddressing devices, and setting device modes for remote or local programming. Appendix B, *Multiline Interface Messages*, lists the GPIB command messages.

Talkers, Listeners, and Controllers

A Talker sends data messages to one or more Listeners. The Controller manages the flow of information on the GPIB by sending commands to all devices.

Devices can be Listeners, Talkers, and/or Controllers. A digital voltmeter, for example, is a Talker when sending measurements and a Listener when receiving programming messages.

The GPIB is like an ordinary computer bus, except that the computer has its circuit cards interconnected via a backplane bus, whereas the GPIB has standalone devices interconnected via a cable bus.

The Role of the GPIB Controller

The role of the GPIB Controller can be compared to the role of a switching center of a city telephone system. The switching center (Controller) monitors the communications network (GPIB). When the center (Controller) notices that a party (device) wants to make a call (send a data message), it connects the caller (Talker) to the receiver (Listener).

The Controller must address a Talker and a Listener before the Talker can send its message to the Listener. After the message is transmitted, the Controller can unaddress both devices.

Some bus configurations do not require a Controller. For example, one device might always be a Talker (a Talk-only device) and there can be one or more Listen-only devices.

A Controller is necessary when the active or addressed Talker or Listener must be changed. The Controller functions are usually handled by a computer.

The GPIB-232/485CT-A as Talker, Listener, and Controller

In S mode, the GPIB-232/485CT-A performs the following three roles:

- Controller—to manage the GPIB
- Talker—to send data to an attached GPIB device
- Listener—to receive data from an attached GPIB device

In G mode, the GPIB-232/485CT-A acts as a GPIB device. It performs only the following two roles:

- Talker—to send data to the GPIB host
- Listener—to receive data from the GPIB host

The Controller-In-Charge and System Controller

Although the GPIB can have multiple Controllers, only one Controller at a time is active, or Controller-In-Charge (CIC). Active control can be passed from the current CIC to an idle Controller. Only one device on the bus, the System Controller, can make itself the CIC. The GPIB interface is usually the System Controller in S mode, but it is never the System Controller in G mode.

GPIB Signals and Lines

The interface system consists of 16 signal lines and 8 ground return or shield drain lines.

The 16 signal lines are divided into the following three groups:

- Eight data lines
- Three handshake lines
- Five interface management lines

Figure F-1 shows the arrangement of these signals and lines on the GPIB cable connector (a * suffix indicates that the signal is active low).

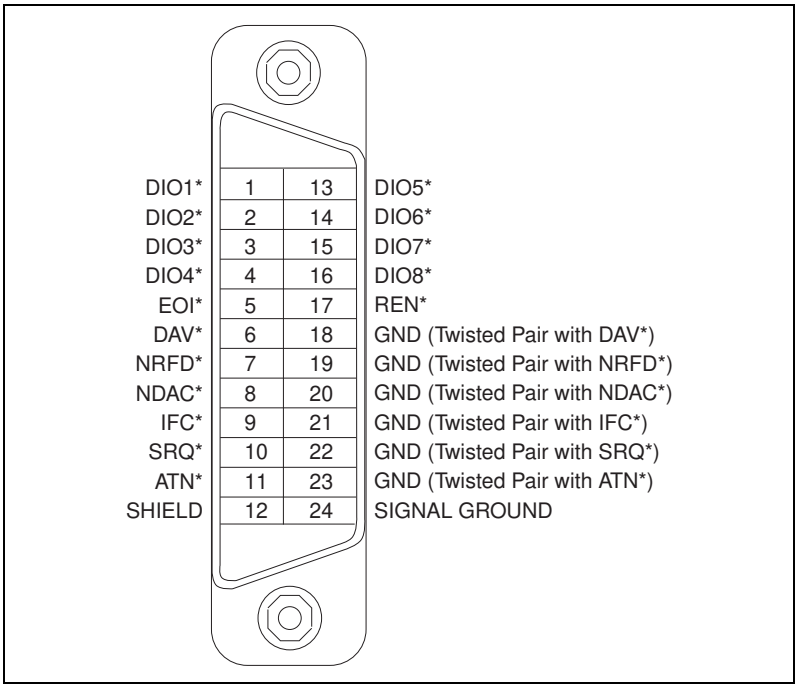


Figure F-1. GPIB Connector Signals and Lines

Data Lines

The eight data lines, **DIO1*** through **DIO8***, carry both data and command messages. All commands and most data use the 7-bit ASCII or ISO code set, in which case the eighth bit, **DIO8***, is unused or used for parity.

Handshake Lines

Three lines asynchronously control the transfer of message bytes among devices. The process is called a three-wire interlocked handshake, and it guarantees that message bytes on the data lines are sent and received without transmission error.

NRFD* (Not Ready for Data)

NRFD* indicates when a device is ready or not ready to receive a message byte. The line is driven by all devices when receiving commands and by Listeners when receiving data messages.

NDAC* (Not Data Accepted)

NDAC* indicates when a device has or has not accepted a message byte. The line is driven by all devices when receiving commands and by Listeners when receiving data messages.

DAV* (Data Valid)

DAV* tells when the signals on the data lines are stable (valid) and can be accepted safely by devices. The Controller drives **DAV*** when sending commands and the Talker drives it when sending data messages.

The way in which **NRFD*** and **NDAC*** are used by the receiving device is called the Acceptor Handshake. Likewise, the sending device uses **DAV*** in the Source Handshake.

Interface Management Lines

Five lines are used to manage the flow of information across the interface.

ATN* (Attention)

The Controller drives **ATN*** true when it uses the data lines to send commands and false when it allows a Talker to send data messages.

IFC* (Interface Clear)

The System Controller drives the **IFC*** line to initialize the bus and become Controller-In-Charge.

REN* (Remote Enable)

The System Controller drives the **REN*** line, which is used to place devices in remote or local program mode.

SRQ* (Service Request)

Any device can drive the **SRQ*** line to asynchronously request service from the Active Controller with the **SRQ*** line.

EOI* (End or Identify)

The **EOI*** line has two purposes. The Talker uses the **EOI*** line to mark the end of a message string. The Controller uses the **EOI*** line to tell devices to identify their response in a parallel poll.

Connecting Devices on the Bus

Devices are usually connected with a cable assembly consisting of a shielded 24 conductor cable with both a plug and receptacle connector at each end. This design allows devices to be linked in either a linear or a star configuration, as shown in Figures F-2 and F-3, or a combination of the two.

The standard connector is the Amphenol or Cinch Series 57 *Microribbon* or *Amp Champ* type. An adapter cable using a non-standard cable and/or connector is used for special interconnection applications.

The GPIB uses negative logic with standard TTL logic levels. When **DAV*** is true, for example, it is a TTL low level (≤ 0.8 V), and when **DAV*** is false, it is a TTL high level (≥ 2.0 V).

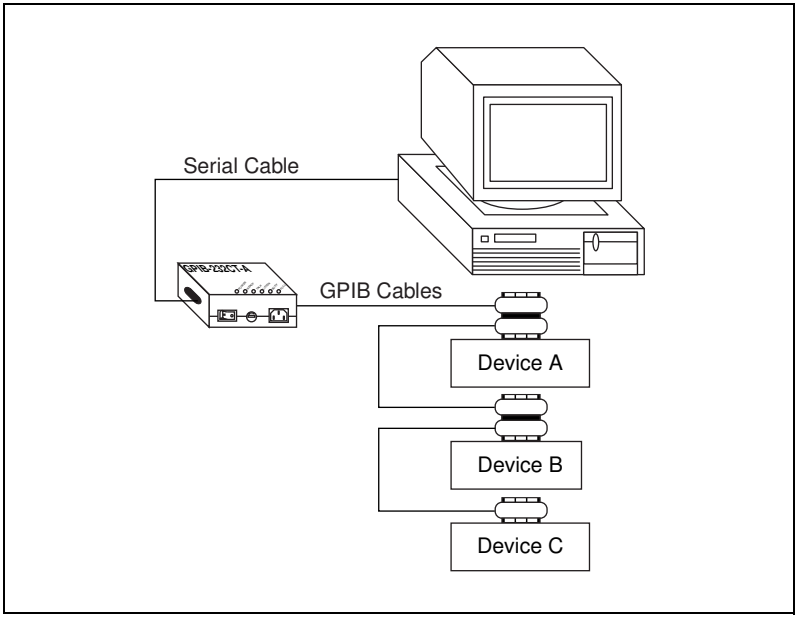


Figure F-2. Linear Configuration of GPIB Devices

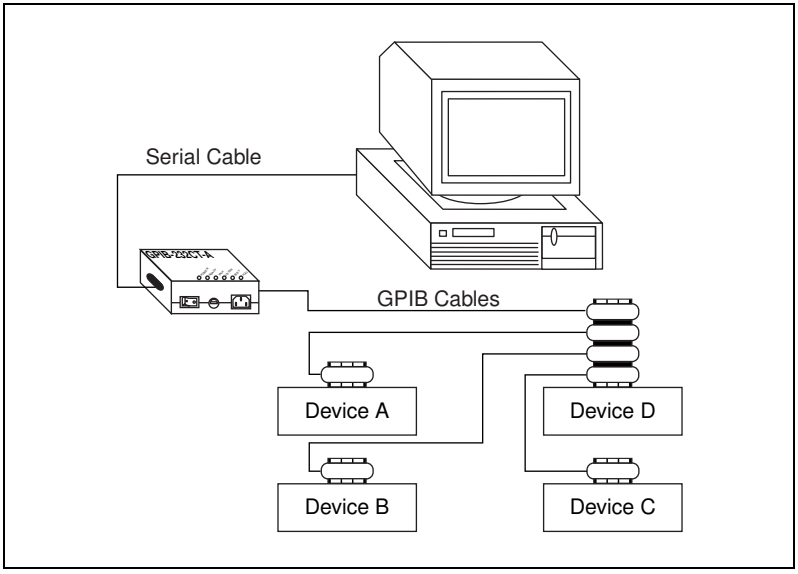


Figure F-3. Star Configuration of GPIB Devices

Configuring Devices on the Bus

If you want to achieve the high data transfer rate that the GPIB was designed for, you must limit the physical distance between devices and the number of devices on the bus. Bus extenders are available from National Instruments if you need to overcome these limitations.

Use the following guidelines:

- A maximum separation of 4 m between any two devices and an average separation of 2 m over the entire bus.
- A maximum total cable length of 20 m.
- No more than 15 devices connected to each bus, with at least two-thirds powered on.

G

Common Questions

This appendix contains answers to common questions about S mode and G mode.

S Mode Questions

Why is there an unmarked DIP switch on my GPIB-232/485CT-A?

The second DIP switch is reserved for future expansion and should remain OFF.

Why does the manual suggest that I use `INPUT$` sometimes, and `LINE INPUT#` at other times? Microsoft suggests using `INPUT$` to read from the serial port.

Use `LINE INPUT#` to read status information from the GPIB-232/485CT-A. GPIB-232/485CT-A software formats its status information so that your BASIC program can read and interpret each of its pieces easily. Each logical piece of status information is followed by a carriage return (<CR>) and linefeed (<LF>). `LINE INPUT#` allows you to read each piece of status information easily and assign it to a variable.

Use `INPUT$` to read a data string from your GPIB device. `INPUT$` requires that you know the exact number of characters you want to read from the serial port. When reading status information from the GPIB-232/485CT-A, the responses can vary in length from one call to the next. However, when reading a data string from your GPIB device, you request a certain number of bytes. You should use `INPUT$` to read the number of bytes you request in your `rd` function. The GPIB-232/485CT-A appends to the end of the data string a string containing the number of bytes that were actually read from the GPIB. When you have read in the data bytes using `INPUT$`, use `LINE INPUT#` to read the string containing the byte count. Refer to the example following the `rd` function description in Chapter 5, *S Mode Functions*.

When I use `LINE INPUT#`, my strings are usually preceded by a linefeed (<LF>) and followed by a carriage return (<CR>). Why don't my strings contain both a carriage return and linefeed at the end?

`LINE INPUT#` stops reading when a carriage return is seen and does not skip over the linefeed in the sequence. The linefeed is not read until the following `LINE INPUT#`. In most cases, you use the `val` function in BASIC to convert the string to a value, and the leading linefeed is ignored.

I sent the programming message `"rsp 10"` to the GPIB-232/485CT-A to serial poll device 10. Then, I used `LINE INPUT#` to read the response byte and got nothing but a carriage return and linefeed (<CR><LF>) as a response. Am I doing something wrong?

No. To conduct a serial poll, the GPIB-232/485CT-A must be CIC or it must be able to become Controller-In-Charge. If the GPIB-232/485CT-A cannot become CIC, no serial poll is conducted, and therefore you do not get a response string. To see if this is the problem, ask for status (see *stat* in Chapter 5, *S Mode Functions*) and check for the ECIC error. If ECIC did occur, then you passed control or System Controller authority to some other GPIB device, and are therefore not able to perform a serial poll until the GPIB-232/485CT-A has Controller authority.

G Mode Questions

Why is there an unmarked DIP switch on my GPIB-232/485CT-A?

The second DIP switch is reserved for future expansion and should remain OFF.

After I write the programming message `"stat c s"` to the GPIB-232/485CT-A, status is not returned. All I get is a carriage return followed by a linefeed (<CR><LF>). Why don't I get status information?

Every programming message must be followed by a carriage return and/or linefeed. In BASIC, you can build your string as follows:

```
"stat c s"+chr$(13)
```

After I write the programming message `"stat c s"+chr$(13)` to the GPIB-232/485CT-A in G mode, my system times out when I try to read status. Why?

Make sure you have addressed the GPIB-232/485CT-A to talk. To read responses to programming messages you send to the GPIB-232/485CT-A, you must address the GPIB-232/485CT-A to talk using the address you selected on the configuration switch and adding decimal 64 to it.

Whenever I send a programming message to the GPIB-232/485CT-A, it seems as if the GPIB-232/485CT-A never receives it. The GPIB-232/485CT-A is powered on and its READY LED is on.

Switch 1 of the DIP switch must be ON for G mode and switches 2 and 3 must be OFF. If you must change these switch settings, be sure to power the GPIB-232/485CT-A off and then power it on again. Check all of your cable connections.

How can I abort the currently executing function and reset the GPIB-232/485CT-A to its power-on state?

First, you can abort any ongoing data transfer or function execution by sending the GPIB Device Clear to the GPIB-232/485CT-A. This is done by sending either the universal Device Clear (DCL) command, or by sending its listen address followed by the Selected Device Clear (SDC) command. Then, you can send the `onl 1` programming message to reset all of the settings of the GPIB-232/485CT-A to their power-on state.

H

Parallel Polling

This appendix explains the use and operation of parallel polls.

By using parallel polls, a GPIB Controller can obtain information from several devices on the GPIB in one operation. The Controller polls configured devices and reads back a single response byte that contains one bit of information from each device. From this information, the Controller can determine which devices need service.

Operation

A parallel poll can be conducted by the GPIB Controller at any time. To execute the poll, the Controller sends the **IDY*** (identify) message on the bus. A device which is configured for parallel polls responds at this time by driving an assigned data line either TRUE or FALSE depending on the value of its individual status (`ist`) bit. When a parallel poll is conducted, the device determines the value of its `ist` bit and drives the line to the appropriate value. Whether this line is TRUE or FALSE depends on how the device is configured for the poll.

The circumstances under which a device sets its `ist` bit are specific to that device. For example, a device might always set its `ist` bit to 1 when it is busy and 0 when it is free, or vice versa. Consult your device documentation for this information.

There are two steps to conducting a parallel poll:

1. Configuration—sets up the devices to participate in the poll
2. The parallel poll—reads the data lines and reports the result

The following sections describe these two steps.

Configuration

There are two methods by which a device can be configured to respond to parallel polls. Only one of these two methods may be used by a device.

- The device configures itself. This is referred to in the IEEE 488 specification as Parallel Poll (PP) Interface Function Subset PP2, or local configuration.
- The device allows itself to be configured by an external Controller. This is referred to in the IEEE 488 specification as PP Interface Function Subset PP1, or remote configuration.

In S mode, it is possible to choose which configuration method to use by setting or clearing the PP2 option of the `conf` function. When the PP2 option is cleared (the default), the GPIB-232/485CT-A uses PP subset PP1. This causes the GPIB-232/485CT-A to accept only those configurations that come over the GPIB from an external Controller. When the PP2 option is set, the GPIB-232/485CT-A uses PP subset PP2. This causes the GPIB-232/485CT-A to allow local configurations, and to ignore configurations that come over the GPIB from an external Controller.

In G mode, PP subset PP1 is the only allowable method for parallel poll configurations. Thus, the GPIB Controller in your system must configure the GPIB-232/485CT-A in order for it to respond to parallel polls.

Parallel poll configurations are accomplished by using Parallel Poll Enable (PPE) messages and Parallel Poll Disable (PPD) messages. There are 16 possible PPE messages—hex 60 through hex 6F. There are also 16 possible PPD messages—hex 70 through hex 7F. Figure H-1 shows the meaning of the bits in the PPE and PPD messages.

			U	S	DIO lines 1 through 8		
0	1	1	X	X	X	X	X

Figure H-1. Parallel Poll Message (PPE or PPD) Bits

Table H-1 contains a list of the parallel poll message bits and a description of each bit. Because the U bit is set in all of the PPD messages, they all have the same effect. All of the PPD messages disable the device from responding to parallel polls.

Table H-1. Parallel Poll Message Bit Descriptions

Bit	Description
U	If 0 (hex 6X), parallel poll is enabled. If 1 (hex 7X), parallel poll is disabled.
S	If the ist (individual status) bit of the device matches the S bit, the device sets the appropriate data line. Hex 60 through hex 67, set S to 0; hex 68 through hex 6F, set S to 1.
DIO	The value n in bits 0 through 2 corresponds to one of the DIO lines 1 through 8, where n corresponds to DIO line n+1. Thus, a value of 2 (binary 010) corresponds to DIO line 3.

Issuing Remote Configurations in S Mode

In S Mode, the ppc and ppu functions are used to remotely configure devices for parallel polls. Remote configuration is indicated by passing the addresses of the GPIB devices to be configured in the addr arguments. For example, if you want to configure a device at address 5 to respond on DIO line 3 when its ist bit is 1, send the following programming message to the GPIB-232/485CT-A:

```
ppc 5,3,1<CR>
```

Figure H-2 shows the PPE message that is formed by the ppc programming message given above.

			U	S	DIO lines 1 through 8		
0	1	1	0	1	0	1	0

Figure H-2. Sample PPE Message Bits

The value of this byte is hex 6A where:

U = 0 Enable parallel poll responses

S = 1 When ist = 1 the device asserts DIO line 3 (which corresponds to 010 in bits 0 through 2) in response to a parallel poll.

The ppc function is used to enable GPIB devices to respond to parallel polls, and the ppu function is used to disable parallel poll responses.

For every addr, ppr, s configuration, the ppc function sends the talk address of the GPIB-232/485CT-A, unlisten, the listen address specified by addr, Parallel Poll Configure, then the Parallel Poll Enable message (PPE). The PPE message is formed by using ppr as the DIO line and s as the S bit.

For every addr argument given, the ppu function sends the talk address of the GPIB-232/485CT-A, unlisten, the listen address specified by addr, Parallel Poll Configure, then the Parallel Poll Disable (PPD) message. The PPD message disables only one GPIB device at a time from responding to parallel polls.

If the ppu function is called without an argument, it disables all GPIB devices from responding to parallel polls. It does this by sending out the Parallel Poll Unconfigure (PPU) message. This message has the same effect as using the ppu function to send PPD messages to every device on the GPIB.

For the ppc or ppu function to have an effect, the device that it is configuring must be using PP subset PP1. If the device is not using PP subset PP1, then it does not allow itself to be configured by an external Controller, and will probably ignore the configuration.

Issuing Local Configurations in S Mode

In S mode, the ppc and ppu functions are also used to locally configure the GPIB-232/485CT-A itself. Local configuration is indicated by sending one addr argument with the value 255. For example, if you want to configure the GPIB-232/485CT-A to respond on DIO line 5 when its ist bit is 0, send the following programming message to the GPIB-232/485CT-A:

```
ppc 255,5,0<CR>
```

When configuring the GPIB-232/485CT-A itself, the ppc or ppu function does not send messages out on the GPIB. Instead, the GPIB-232/485CT-A internally enables or disables parallel poll responses as indicated.

For the ppc or ppu function to have an effect, the GPIB-232/485CT-A must be using PP subset PP2. If the GPIB-232/485CT-A is not configured to use PP subset PP2, then it cannot allow itself to be locally configured, and returns an ECAP error. The GPIB-232/485CT-A can be configured to use PP subset PP2 with the conf function.

The Parallel Poll

In S mode, after parallel poll configuration is complete, the GPIB-232/485CT-A conducts a parallel poll by calling `rpp`. In the previous example, where the device was remotely configured using a PPE message of hex 6A, if the `ist` bit of the device is set, `rpp` returns a value of hex 04. Here, the third least significant bit is set, corresponding to DIO line 3. (If any other devices responded positively on other lines, those corresponding bits would be set as well.)

Note More than one device can be configured to respond on the same data line, in which case the bits in the response byte are set by the ORing of all the responses on that line.

In G Mode, the GPIB-232/485CT-A sets its `ist` bit whenever it asserts **SRQ***, and clears it whenever it unasserts **SRQ***. Refer to the `srqen` function description in Chapter 7, *G Mode Functions*, for the conditions under which the GPIB-232/485CT-A asserts **SRQ***. If the Controller remotely configures the GPIB-232/485CT-A using a PPE message of hex 6D (binary 0110 1101) and parallel polls it while it is asserting **SRQ***, it responds by asserting DIO line 6. If the Controller remotely configures the GPIB-232/485CT-A using a PPE message hex 65 (binary 0110 0101) and parallel polls it while it is asserting **SRQ***, it responds by not asserting DIO line 6.

S Mode Parallel Polling Examples

An example system has three line printers, one scanner, and one PC with a serial port. The PC uses a GPIB-232/485CT-A to communicate on the GPIB. The GPIB-232/485CT-A is the designated Controller, and all other devices recognize it as the Controller. All of the GPIB devices set their `ist` bit to 1 when they are busy and 0 when they are free. Furthermore, all of the GPIB devices use PP subset PP1 (remote configuration by the Controller).

Example 1

Figure H-3 shows an example parallel polling situation. The configuration and responses are described in the following paragraphs.

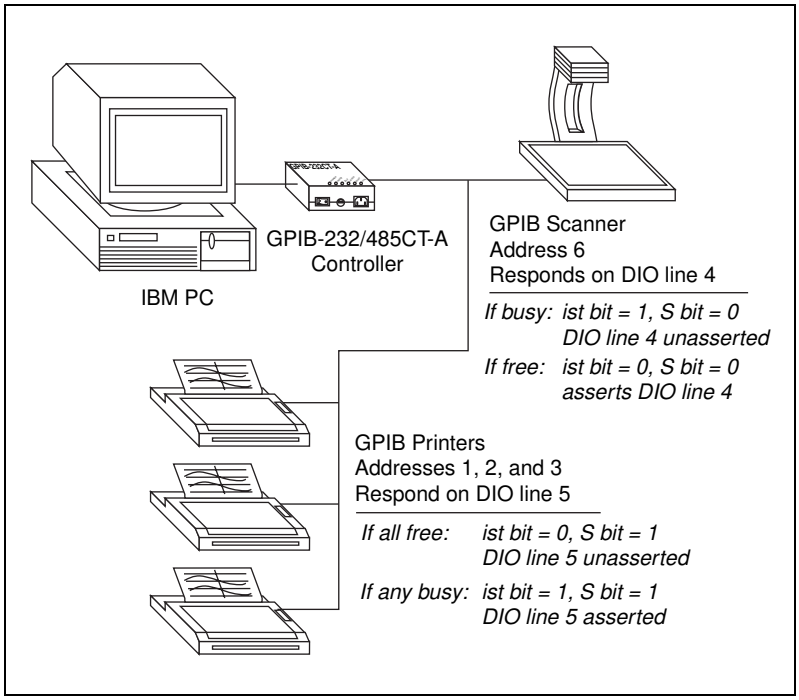


Figure H-3. Parallel Polling Setup Example 1

The PC configures the scanner (at address 6) to respond positively on DIO line 4 when free by sending the following programming message to the GPIB-232/485CT-A:

```
ppc 6,4,0<CR>
```

This programming message causes a PPE message of hex 63 to be sent to the scanner. When a parallel poll is conducted by calling `rpp`, one of two things happens. If the scanner is free, it asserts DIO line 4 (`rpp` returns hex 8); if it is busy, it unasserts DIO line 4 (`rpp` returns hex 0). When the scanner is free, its `ist` bit is 0, and because this equals the value of the `S` bit in the PPE message, the device asserts DIO line 4.

The PC configures all of the line printers (at addresses 1, 2, and 3) to respond positively on DIO line 5 when busy by sending the following programming message to the GPIB-232/485CT-A:

```
ppc 1,5,1 2,5,1 3,5,1<CR>
```

This programming message causes a PPE message of hex 6C to be sent to each of the line printers. When a parallel poll is conducted by calling `rpp`, the PC can immediately find out if any printer is busy or if all printers are free. If all line printers are free, `rpp` returns hex 0 (or hex 8 if the scanner is free). When a line printer is free, its `ist` bit is 0, and because this does not equal the value of the `S` bit in the PPE message, the device unasserts DIO line 5. If one or more line printers is busy, `rpp` returns hex 10 (or hex 18 if the scanner is free). When a line printer is busy, its `ist` bit is 1, and because this equals the value of the `S` bit in the PPE message, the device asserts DIO line 5.

Example 2

Instead of checking if any printer is *busy*, you might want to know if any printer is *free*. Figure H-4 shows a parallel polling setup to accomplish this. The configuration and responses are described in the following paragraphs.

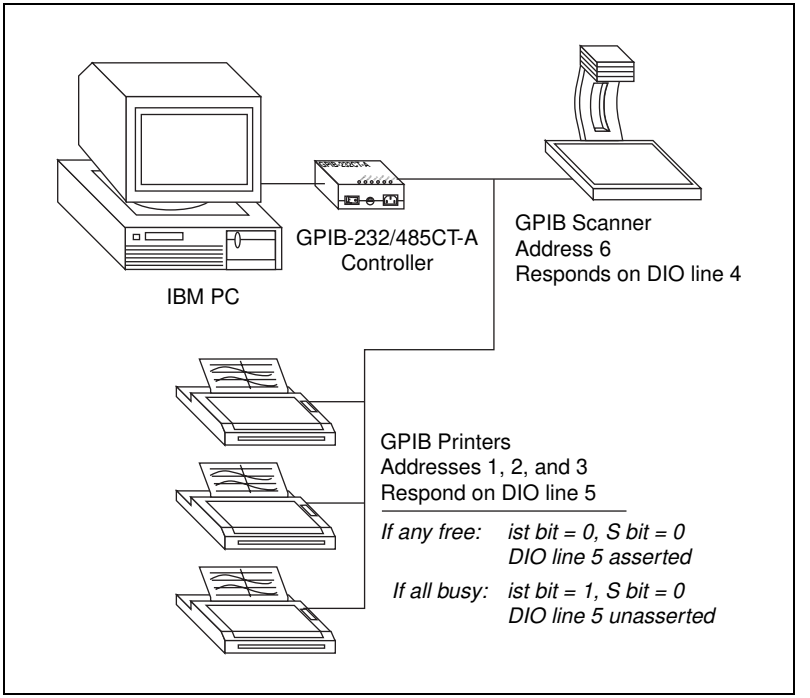


Figure H-4. Parallel Polling Setup Example 2

To find out if any printer is free, you must change the `S` bit/`ist` bit correspondence by reconfiguring the line printers with the following programming message:

```
ppc 1,5,0  2,5,0  3,5,0<CR>
```

This programming message causes a PPE message of hex 68 to be sent to each of the line printers. When a parallel poll is conducted by calling `rpp`, the PC can find out if any line printer is free or if all line printers are busy.

Programming Steps and Examples

This appendix contains program examples you can use as guidelines as you start writing programs for the GPIB-232/485CT-A.

The first part of the appendix applies to S mode. The remainder of the appendix applies to G mode.

General Programming Steps for S Mode

The following are general programming steps for S mode operation.

1. Send serial port initialization functions if you need to change default serial port settings.
2. Send the `stat` function to have status information returned to you after your programming message.
3. Send GPIB initialization functions if you need to change default GPIB settings.
4. Communicate with the device using the `rd` and `wrt` functions, and check status if you requested it.

After you initialize the GPIB-232/485CT-A, the `rd` and `wrt` functions may be the only functions you need.

S Mode Sample Programs

The following pages contain detailed explanations of these steps along with sample programs.

Using an HP 7475A Plotter with a Terminal

This program is a sequence of programming messages and data strings entered from a terminal to draw a circle using an HP 7475A Color Plotter.

Preparing to Program

1. Before you start programming, determine the serial port settings you need by looking at the settings of your terminal.
2. Set the configuration switches on the GPIB-232/485CT-A so that they match the terminal characteristics. Figure I-1 shows how to configure the GPIB-232/485CT-A if your terminal has the following characteristics:

Baud rate	19200
Parity	None
Data bits	8
Stop bits	1

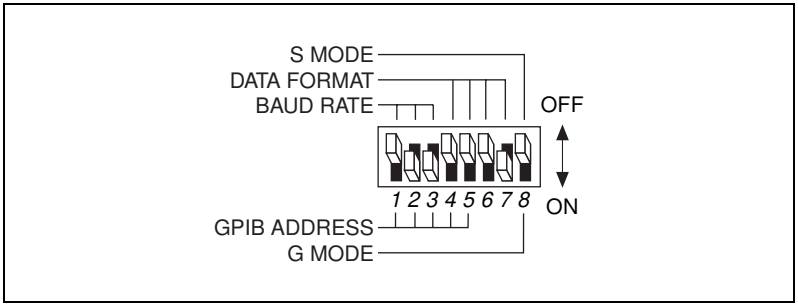


Figure I-1. Sample Switch Settings for a Terminal and HP Plotter

3. Connect the serial cable to the serial port of the terminal and to the GPIB-232/485CT-A.
4. Connect the GPIB cable to the GPIB port on your device and to the GPIB-232/485CT-A.
5. Power on the GPIB-232/485CT-A.

Programming Steps

Step 1. Send the Serial Port Functions

A program written from a terminal does not exactly follow the steps used for most programs. To see a display on the screen of the information you are entering, you must enable the `echo` function. To enable `echo`, enter the following command:

```
echo 1 <CR>
```

Now characters you type are echoed to your terminal.

Determine other serial port functions you need to change. No other changes are necessary for this example.

Step 2. Send the stat Function

When using the `stat` function, use `stat c s` so the GPIB-232/485CT-A reports status after each programming message. In `stat c s`, the `c` stands for continuous, and the `s` stands for symbolic. Because you are using a terminal, it is easier to interpret the status report if it is returned in symbolic form. Enter the following line:

```
stat c s <CR>
```

The status displays on the screen of the terminal immediately. The status should look something like this:

```
CMPL
NGER
NSER
0
```

This status report shows the status after the previous programming message. This status reports `CMPL` and no errors.

Step 3. Send the GPIB Initialization Functions

Determine what GPIB initialization functions are needed to change default settings. If you are not sure what GPIB settings you might want to change, try using the defaults. No changes are necessary for this GPIB plotter.

Step 4. Communicate with the Device Using rd and wrt Functions

Use `rd` and `wrt` to communicate with the device. In this example we want to send plotter commands to the GPIB plotter, so we only need the `wrt` programming message.

By looking at the status information returned after each programming message, you can see if any GPIB or serial port errors occurred, and you can see the number of bytes actually transferred out on the GPIB.

In the following example, the programming messages and data strings that you enter are shown in regular type. Responses sent by the GPIB-232/485CT-A are shown in **bold** type. The command (`tmo 30`)

on the second line lengthens the timeout from 10 seconds (default) to 30 seconds to allow you more time to type the data string for the `wrt` commands. The address of the plotter is 5.

```
echo 1<CR>
tmo 30<CR>
stat c s<CR>
CMPL
NGER
NSER
0
wrt 5<CR>
IN;SP1;IP2650,1325,7650,6325;<CR>
CMPL,CIC,TACS
NGER
NSER
29
wrt<CR>
SC-100,100,-100,100;PA0,0;CI40;<CR>
CMPL,CIC,TACS
NGER
NSER
31
```

Using an HP 7475A Plotter with an IBM PC

This example shows how to write a program on an IBM PC using Microsoft BASIC to draw a circle using an HP 7475A Plotter.

Preparing to Program

1. Before you start programming, determine the serial port settings you need.
2. Set the configuration switches on the GPIB-232/485CT-A. Figure I-2 shows how to configure the GPIB-232/485CT-A for the following characteristics:

Baud rate	9600
Parity	None
Data bits	7
Stop bits	1

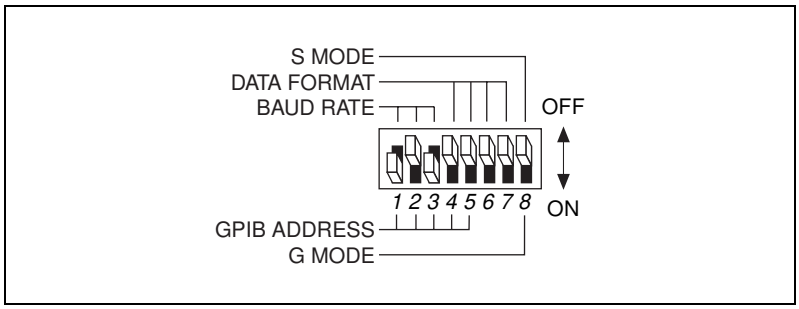


Figure I-2. Sample Switch Settings for an IBM PC and HP Plotter

3. Connect the serial cable to the serial port of your computer and to the GPIB-232/485CT-A.
4. Connect the GPIB cable to your device and the GPIB-232/485CT-A.
5. Power on the GPIB-232/485CT-A.

Programming Steps

Step 1. Send the Serial Port Functions

Send serial port initialization programming messages, if necessary. For this example, this step is not necessary.

Step 2. Send the stat Function

1. In BASIC, before reading or writing to the serial port, a device must be opened. Place the following BASIC statement at the beginning of your program to open and configure the serial port (COM1) and name it device `#1`.

   ```
   OPEN "COM1:9600,n,7,1" AS #1
   ```

2. Use the `PRINT #1` statement to redirect strings to the serial port.

   Send the `stat` function if you want status information returned after every programming message. To send `stat`, include the following code in your program:

   ```
   PRINT #1, "stat c n"
   ```

3. After you send this programming message, you can expect four lines of data at the serial port (each line is terminated by <CR><LF>). For now, call a subroutine to check the status. You can write this later. Add the following line to call the subroutine:

   ```
   GOSUB status
   ```

Step 3. Send the GPIB Initialization Functions

Send GPIB initialization programming messages, if necessary. For this example, this step is not necessary.

Step 4. Communicate with the Device rd and wrt Functions

Communicate with the device using `wrt` programming messages, and reading back status after each. This is the main part of the program. After each `wrt` string, call the subroutine `status`, which checks for errors. The plotter's GPIB address is 5.

```
OPEN "com1:9600,n,7,1" AS #1
PRINT #1,"stat c n"
GOSUB status
PRINT #1,"wrt 5"
PRINT #1,"in;sp1,pa1000,3000;ci500;"
GOSUB status
END

status:
STAT%=VAL(LINE INPUT #1, status$)
LINE INPUT #1,gpiberr$
LINE INPUT #1,sperr$
LINE INPUT #1,cnt$
PRINT status$ gpiberr$ sperr$ cnt$
if stat% < 0 GOSUB error

error:
REM Place your code to handle errors here.
STOP
```

General Programming Steps for G Mode

The following are general programming steps for G mode operation.

1. Send the `stat` function to have status information returned to you after your programming message.
2. Send GPIB initialization functions if you need to change default GPIB settings.
3. Send serial port initialization functions if you need to change default serial port settings.
4. Communicate with the serial device and obtain status from the GPIB-232/485CT-A, if desired.

After you initialize the GPIB-232/485CT-A, you might only need to perform reads and writes from the serial device.

To send data to the device you must address the device. To send programming messages to the GPIB-232/485CT-A, you must address the GPIB-232/485CT-A. The GPIB software you use on the Controller side can provide high-level calls that perform this addressing automatically.

G Mode Sample Programs

The following pages contain detailed explanations of the general programming steps along with sample programs. The examples use NI-488.2 functions to perform the GPIB operations. To communicate with the GPIB-232/485CT-A, send information to it using `ibwrt`. To receive information from the GPIB-232/485CT-A, use `ibrd`.

Using a Serial HP 7475A Plotter with a GPIB Controller

This example shows how to draw a circle using an HP 7475A Color Plotter with a serial interface. The Controller is an IBM PC with the National Instruments AT-GPIB interface and software.

Preparing to Program

1. Set the GPIB-232/485CT-A configuration switches to primary address 2, as shown in Figure I-3.

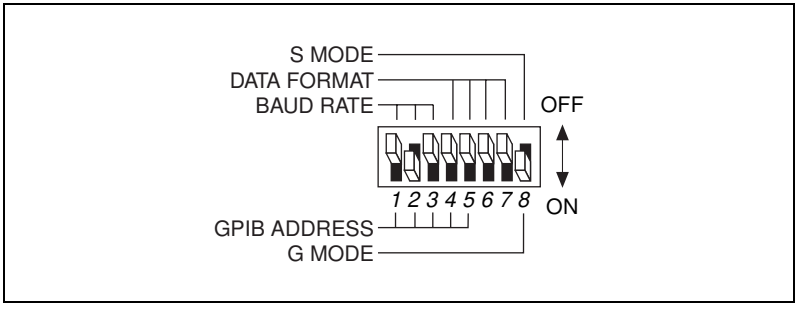


Figure I-3. Sample Switch Settings for an HP Serial Plotter

The serial port characteristics of the GPIB-232/485CT-A and the serial device must be exactly the same. In this case, the serial port settings of the plotter are as follows:

Baud rate	9600
Parity	None
Data bits	7
Stop bits	1

The GPIB-232/485CT-A default serial port characteristics are as follows:

Baud rate	9600
Parity	None
Data bits	8
Stop bits	1

The number of data bits do not match, so you must change the default setting of the GPIB-232/485CT-A before you read or write from the plotter. You accomplish this during the programming steps.

2. Connect the GPIB cable to the GPIB port on your computer and to the GPIB-232/485CT-A.
3. Connect the serial cable to the serial port on your device and to the GPIB-232/485CT-A.
4. Power on the GPIB-232/485CT-A.

Programming Steps

Step 1. Send the stat Function

Send `stat c n` if you want status information returned after each programming message. You can read the status after this and all subsequent commands sent to the GPIB-232/485CT-A until continuous status reporting is disabled.

Step 2. Send the GPIB Initialization Functions

You can change any default GPIB functions on the GPIB-232/485CT-A, but no changes are necessary to communicate with this plotter.

Step 3. Send the Serial Port Initialization Functions

Change any serial port initialization functions, if necessary. This plotter sends and receives 7-bit data. Since the default of the GPIB-232/485CT-A is 8-bit data, send the string `spset 7` to change to 7-bit data transfers. Read status after you send the string.

Step 4. Communicate with the Plotter

Before communicating with the plotter, send the initialization strings to the GPIB-232/485CT-A. Next, send strings to the plotter (the GPIB-232/485CT-A does not interpret these strings, but sends them straight to the plotter). Read status after you send a programming message to the GPIB-232/485CT-A.

```
 10  GCT$="GPIBCT"
 15  CALL IBFIND(GCT$,GPIBCT%)       'Open GPIB-232/485CT-A.
 20                                  'GPIBCT is a GPIB device
 25                                  'configured in ibconf to have
 30                                  'primary address.
 35  SDNAME$="PLOTTER"
 40  CALL IBFIND(SDNAME$,PLOTTER%)   'Open PLOTTER.
 45                                  'PLOTTER is a GPIB device
 50                                  'configured in ibconf to have
 55                                  'primary address.
 60  WRT$="stat c n" + CHR$(13)
 65  CALL IBWRT(GPIBCT%,WRT$)        'Enable continuous status
 70                                  'reporting.
 75  RD$=SPACE$(25)
 80  CALL IBRD(GPIBCT%,RD$)          'Read up to 25 bytes of
 85                                  'status information.
 90  IF ASC(RD$) = CHR$(45) THEN GOTO 210
 95                                  'If first character in RD$ is
100                                  'a minus sign, then go to
105                                  'error function.
110  WRT$="spset 7"                  'Change the serial port
115                                  'configuration to 7 data
120  CALL IBWRT(GPIBCT%,WRT$)        'bits (plotter configuration).
125  CALL IBRD(GPIBCT%,RD$)          'Read status info.
130  IF ASC(RD$) = CHR$(45) THEN GOTO 210
135                                  'If first character in RD$ is
140                                  'a minus sign, then go to
145                                  'error function.
150  WRT$="IN;SP1;IP2650,1325,7650,6325;"
155                                  'Initialize plotter, select
160                                  'pin 1, set 'scaling points.
165  WRT$=WRT$ + "SC-100,100,-100,100;"
```

```
170                                  'Scale the plotting area.
175  WRT$=WRT$ + "PA50,-40;CI30,30;"
180                                  'Plot absolute, circle=radius
185                                  '30 degrees, chordangle 30
190                                  'degrees.
200  CALL IBWRT(PLOTTER%,WRT$)
205  STOP
210  PRINT "An error occurred:"
215  PRINT "status, GPIB-error, serial-error,
220  count:";RD$
230  END
```

Controlling a Serial Printer with an IBM PC

This example demonstrates how to control a serial printer on the GPIB. The printer is the Apple Imagewriter. The Controller is an IBM PC with an AT-GPIB interface installed. The following program example prints the message **`Hello, world`**.

Preparing to Program

1. Set the GPIB-232/485CT-A configuration switch to primary address 18, as shown in Figure I-4.

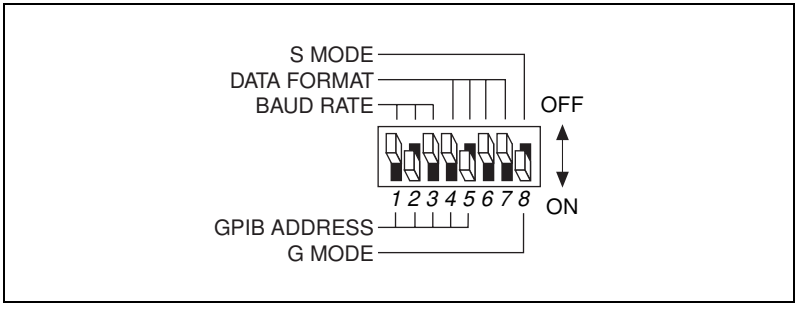


Figure I-4. Sample Switch Settings for an IBM PC and Serial Printer

2. Check the DIP switch settings on the Imagewriter. Switches 2–1 and 2–2 should be *closed* for 9600 baud rate.

Programming Steps

Step 1. Send the stat Function

Send `stat c n` if you want status information returned after each programming message. You can read the status after this and all subsequent commands sent to the GPIB-232/485CT-A until continuous status reporting is disabled.

Step 2. Send the GPIB Initialization Functions

You can change any default GPIB functions on the GPIB-232/485CT-A, but no changes are necessary to communicate with this plotter.

Step 3. Serial Port Initialization Functions

Next, change any serial port characteristics, if necessary. The serial port characteristics of the GPIB-232/485CT-A must match those of the serial device. Since the baud rate of the printer and the GPIB-232/485CT-A are both 9600, there is no need to initialize the software of the serial port.

Step 4. Communicate with the Printer

Before communicating with the printer, send the initialization strings to the GPIB-232/485CT-A. Next, send strings to the printer (the GPIB-232/485CT-A does not interpret these strings, but sends them straight to the printer). Read the status after you send a programming message to the GPIB-232/485CT-A.

```
10  GCT$="GPIBCT"
15  CALL IBFIND(GCT$,GPIBCT%)       'Open GPIB-232/485CT-A.
20                                  'GPIBCT is a GPIB device
25                                  'configured in ibconf to have
30                                  'primary address.
35  SDNAME$="PRINTER"
40  CALL IBFIND(SDNAME$,PRINTER%)   'Open PRINTER.
45                                  'PRINTER is a GPIB device
50                                  'configured in ibconf to have
55                                  'primary address.
60  WRT$="stat c n"+CHR$(13)
65  CALL IBWRT(GPIBCT%,WRT$)        'Enable continuous status
70                                  'reporting.
75  RD$=SPACE$(25)
80  CALL IBRD(GPIBCT%,RD$)          'Read up to 25 bytes of status
85                                  'information.
90  IF ASC(RD$) <> CHR$(45) THEN GOTO 310
95                                  'If first character in RD$ is
```

```
100                                 'a minus sign, then go to
105                                 'error function.
110  WRT$="Hello, world"+CHR$(13)
115  CALL IBWRT(PRINTER%,WRT$)      'Send string to printer.
120  STOP
125  PRINT "An error occurred:"
130  PRINT "status, GPIB-error, serial-error,
135  count:";RD%
140  END
```

J

Technical Support Resources

This appendix describes the comprehensive resources available to you in the Technical Support section of the National Instruments Web site and provides technical support telephone numbers for you to use if you have trouble connecting to our Web site or if you do not have internet access.

NI Web Support

To provide you with immediate answers and solutions 24 hours a day, 365 days a year, National Instruments maintains extensive online technical support resources. They are available to you at no cost, are updated daily, and can be found in the Technical Support section of our Web site at `www.ni.com/support`

Online Problem-Solving and Diagnostic Resources

- **KnowledgeBase**—A searchable database containing thousands of frequently asked questions (FAQs) and their corresponding answers or solutions, including special sections devoted to our newest products. The database is updated daily in response to new customer experiences and feedback.
- **Troubleshooting Wizards**—Step-by-step guides lead you through common problems and answer questions about our entire product line. Wizards include screen shots that illustrate the steps being described and provide detailed information ranging from simple getting started instructions to advanced topics.
- **Product Manuals**—A comprehensive, searchable library of the latest editions of National Instruments hardware and software product manuals.
- **Hardware Reference Database**—A searchable database containing brief hardware descriptions, mechanical drawings, and helpful images of jumper settings and connector pinouts.
- **Application Notes**—A library with more than 100 short papers addressing specific topics such as creating and calling DLLs, developing your own instrument driver software, and porting applications between platforms and operating systems.

Software-Related Resources

- **Instrument Driver Network**—A library with hundreds of instrument drivers for control of standalone instruments via GPIB, VXI, or serial interfaces. You also can submit a request for a particular instrument driver if it does not already appear in the library.
- **Example Programs Database**—A database with numerous, non-shipping example programs for National Instruments programming environments. You can use them to complement the example programs that are already included with National Instruments products.
- **Software Library**—A library with updates and patches to application software, links to the latest versions of driver software for National Instruments hardware products, and utility routines.

Worldwide Support

National Instruments has offices located around the globe. Many branch offices maintain a Web site to provide information on local services. You can access these Web sites from `www.ni.com/worldwide`

If you have trouble connecting to our Web site, please contact your local National Instruments office or the source from which you purchased your National Instruments product(s) to obtain support.

For telephone support in the United States, dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Brazil 011 284 5011, Canada (Calgary) 403 274 9391,
Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
China 0755 3904939, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Greece 30 1 42 96 427,
Hong Kong 2645 3186, India 91805275406, Israel 03 6120092,
Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456,
Mexico (D.F.) 5 280 7625, Mexico (Monterrey) 8 357 7695,
Netherlands 0348 433466, Norway 32 27 73 00, Poland 48 22 528 94 06,
Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085,
Sweden 08 587 895 00, Switzerland 056 200 51 51,
Taiwan 02 2377 1200, United Kingdom 01635 523545

Glossary

Prefix	Meaning	Value
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}

Symbols

°	degrees
%	percent
Ω	ohms

A

A	amperes
AC	alternating current
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ATN	Attention

C

C	Celsius
CIC	Controller-In-Charge
CSA	Canadian Standards Association
CTS	Clear to Send

D

DCAS	Device Clear Active State
DCE	Data Circuit-terminating Equipment
DCL	Device Clear
DIO	digital input/output
DIP	dual inline package
DTAS	Device Trigger Active State
DTE	Data Terminal Equipment
DTR	Data Terminal Ready

E

EIA	Electronic Industries Association
EOI	end or identify
EOS	end of string

F

FCC	Federal Communications Commission
ft	feet

G

g	grams
GND	ground
GPIB	General Purpose Interface Bus

H

hex	hexadecimal
Hz	hertz

I

IDY	Identify
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IFC	Interface Clear
in.	inches
I/O	input/output
ISO	International Standards Organization
IST	Individual Status

L

LACS	Listener Active State
LED	light-emitting diode
LSI	large scale integration

M

MB	megabytes of memory
m	meters

O

oz	ounces

P

PC	personal computer
PPE	Parallel Poll Enable

R

RAM	random-access memory
REM	Remote
REN	Remote Enable
ROM	read-only memory
RQS	Request Service
RTS	Request to Send
RXD	Receive Data

S

s	seconds
SRQ	Service Request
SRQI	Service Request Input

T

TACS	Talker Active State
TIMO	Timeout
TTL	Transistor-Transistor Logic
TXD	Transmit Data

U

UL	Underwriters Laboratories

V

V	volts
VAC	volts alternating current
VDC	volts direct current

Index

A

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

W

X