ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *National Instruments GPIB-422CT* at our website: *Click HERE*

Each device's 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-422CT returns 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, therefore, there are exactly as many lines of responses, including -1, as the number of addresses you specify.

If you call `rsp` and the GPIB-422CT is not CIC, it attempts to become CIC. If it cannot become CIC, it records the ECIC error. Refer to Appendix B for more information.

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-422CT 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 will be posted.

If you call `rsp` without an argument, the GPIB-422CT records the EARG error.

See Also: `tmo` for timeout information.

Example:

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

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

   DIM SPR%(2)               'Read 3 responses from
   FOR I=0 to 2              'serial port buffer.
   LINE INPUT #1,SPR%(I)  'Store each serial poll
   IF SPR%(I) = -1 THEN GOSUB 1000
   NEXT I                    'Response in the array.
                             '1000 is an error routine.
   REM Code will now interpret poll responses.
```

rsv - Request Service/Set or Change Serial Poll Status Byte

Type: Serial Poll function

Syntax: **rsv** [spbyte]<CR>

Purpose: You use rsv if the GPIB-422CT is not the GPIB Controller and you wish to request service from the Controller using the Service Request (SRQ) signal. The GPIB-422CT will provide a user defined status byte when the Controller serially polls it.

Remarks: The argument spbyte is a numeric string specifying the decimal value of the new GPIB-422CT serial poll response byte.

The serial poll response byte is the status byte the GPIB-422CT provides when serially polled by another device that is CIC. If bit 6 (hex 40 RQS bit) is also set, the GPIB-422CT additionally requests service by asserting the SRQ line.

If you call rsv without an argument, the GPIB-422CT returns a numeric string containing the decimal value of its serial poll status byte.

Examples:

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

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: Specialized Controller function

Syntax: `sic [time]<CR>`

Purpose: You use `sic` if the initialization, I/O, or bus management functions do not meet the needs of your device, and you wish to have more precise control over the GPIB. `sic` makes the GPIB-422CT CIC and initializes the GPIB. `sic` is not a function you will use frequently because in most cases the first I/O or bus management function you call will do this automatically.

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

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 interface board become CIC. When needed, `sic` is generally used at the beginning of a program to make the GPIB-422CT CIC and is used 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.

If you are in a debugging environment, you may want to vary the amount of time IFC is asserted. For example, you may 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.

The GPIB-422CT 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 D.

Examples:

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

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 use `spign` at the beginning of your program if you wish to change the effect that serial port errors have on how the GPIB-422CT processes programming messages and data. This function tells the GPIB-422CT to ignore or not to ignore the occurrence of serial port errors. By default, the GPIB-422CT ignores serial port errors.

Remarks: If the argument `bool` is 0, the GPIB-422CT will not ignore serial port errors. When `bool` is 0, the GPIB-422CT does not execute programming messages that contain serial port errors. A list of serial port errors are given in Appendix B. Also, if a serial port error occurs with any byte contained in a `cmd` or `wrt` data string, the GPIB-422CT discards that data byte and all remaining bytes in the string.

The serial port errors include parity, overrun, framing, and overflow errors.

If the argument `bool` is 1, the GPIB-422CT executes all programming messages and sends all data, even if serial port errors occur as the messages and data bytes are received.

If you call `spign` without an argument, the GPIB-422CT returns to you the current setting.

See Also: `cmd` and `wrt`.

Examples:

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

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: Specialized Controller function

Syntax: **sre** [bool]<CR>

Purpose: You use sre if the I/O and bus management functions do not meet the needs of your device. sre gives you more precise control over the GPIB. Use sre to turn the Remote Enable signal on and off. sre is not a function you will use frequently because in most cases, the first I/O or bus management function you call will set remote enable automatically.

Remarks: If the argument bool is 1, the GPIB-422CT asserts the Remote Enable (REN) signal. If the argument bool is 0, the GPIB-422CT 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-422CT is not System Controller, the GPIB-422CT records the ESAC error.

If you call sre without an argument, the GPIB-422CT returns its current remote status: 1=remote, 0=local.

See Also: rsc, cmd, and loc.

Examples:

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

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

stat - Return GPIB-422CT Status

Type: General Use function

Syntax:
```
stat [[c] n]<CR>
  or
stat [c] s<CR>
  or
stat [c] n s<CR>
```

Purpose: You use `stat` to obtain the status of the GPIB-422CT to see if certain conditions are currently present. You use `stat` 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 your device's responses are likely to be unpredictable. The GPIB-422CT responds with status information in a form depending on the mode or combination of modes you chose. `n` indicates that the status information will be returned as numeric strings. `s` indicates that the status information will be returned in symbolic format, that is, as mnemonic strings. `c` specifies that the status will be returned after each programming message, eliminating the need to call `stat` after each programming message.

Normally, you use `s` only when you are debugging your code and you want to print the mnemonic for each piece of status information.

The status information returned by the GPIB-422CT contains four pieces of information: the GPIB-422CT status, a GPIB-error code, a serial-error code, and a count. The GPIB-422CT returns a <CR><LF> following each piece of the response.

Status represents a combination of GPIB-422CT conditions. Internally in the GPIB-422CT, 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 zero indicates that the condition is not in effect. Since more than one GPIB-422CT 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-422CT detects either a GPIB error or a serial port error. Consequently, when status is negative, an error condition exists, and when status is positive, no error condition exists.

GPIB-error represents a single GPIB error condition present.

Serial-error represents a single serial error condition present.

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

Table 5-2. GPIB Status Conditions

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

(continues)

Table 5-2. GPIB Status Conditions (continued)

Numeric Value (n)	Symbolic Value (s)	Description	Bit
8	TACS	Talker active	3
4	LACS	Listener active	2
2	DTAS	Device trigger state	1
1	DCAS	Device clear state	0

Table 5-3. GPIB Error Conditions

Numeric Value (n)	Symbolic Value (s)	Description
0	NGER	No GPIB error condition to report
1	ECIC	Command requires GPIB-422CT to be CIC
2	ENOL	Write detected no Listeners
3	EADR	GPIB-422CT not addressed correctly
4	EARG	Invalid argument or arguments
5	ESAC	Command requires GPIB-422CT to be SC
6	EABO	I/O operation aborted
7-10	—	Reserved
11	ECAP	No capability for operation

(continues)

Table 5-3. GPIB Error Conditions (continued)

Numeric Value (n)	Symbolic Value (s)	Description
12-16	—	Reserved
17	ECMD	Unrecognized command

Table 5-4. Serial Port Error Conditions

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 B.

In general, the GPIB-422CT 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 call `stat c s` to set up continuous status reporting. After reading the status information returned for the `stat` call, you call `wrt`. The GPIB-422CT then

returns the status information that corresponds to the `wrt` message.

Refer to the following examples for ways in which to make use of the status information.

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

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

Notice that when you send several programming messages to the GPIB-422CT, 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-422CT 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.

Examples:

1.
```
10   PRINT #1,"stat n"     'Get GPIB-422CT status.
20   REM GPIB-422CT responds with:
30   REM 340<CR><LF>0<CR><LF>0<CR><LF>0<CR><LF>
40   REM Now read status into variables.
50   INPUT#1,STATUS%,GPIBERR%,SPERR%,COUNT%
60   REM Go to error routine at 500 if error.
70   IF STATUS% < 0 THEN GOTO 500
80   REM Go to SRQ service routine if SRQ is
90   REM asserted.
100  IF (STATUS% AND &H1000) THEN GOTO 400
     ..
400  REM
410  REM Place code here to service SRQ.
420  REM
     ..
500  REM Print GPIB-error and serial-error values
```

```
510  REM to determine what errors occurred.
520  PRINT "GPIB-error = ";GPIBERR%
530  PRINT "Serial-error = ";SPERR%
540  STOP
```

2.
```
10   PRINT #1, "stat s"
20   REM If it has just read 3 bytes from the
30   REM GPIB, the GPIB-422CT responds with:
40   REM     CMPL,REM,ATN,LACS<CR><LF>NGER<CR><LF>
50   REM     NSER<CR><LF>3<CR><LF>
```

3. The following list illustrates what appears on the screen when you are programming the GPIB-422CT from a terminal. Programming messages you enter are in regular type. GPIB-422CT responses are in **boldface**. 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 use `tmo` at the beginning of your program to change the time limits in effect on the GPIB-422CT. The time limits prevent the GPIB-422CT from hanging indefinitely when waiting for critical events to occur.

Remarks: The arguments `timeio` and `timesp` are numeric strings. `timeio` specifies the amount of time in seconds the GPIB-422CT 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 in which to respond to a serial poll. The power-on timeouts are 10 seconds for `timeio` and 1/10 of a 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. `10,.1` specifies a time of 10 seconds for I/O operations and 1/10 of a second for serial poll response. `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-422CT 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-422CT returns an error flag.

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-422CT returns a numeric string representing the current timeout settings. It records the EARG error if you specify a time value outside the range .00001 to 3600.

The assignment made by this function remains in effect until you call `tmo` again, call `onl`, or turn off the GPIB-422CT.

See Also: `rsp`.

Examples:

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

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

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

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

trg - Trigger Selected Device(s) *

Type: Bus Management function

Syntax: `trg alist<CR>`

Purpose: You use `trg` to trigger the specified devices. The instructions for each GPIB device explain when you should trigger them 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 wish to trigger.

A device address consists of a primary address and an optional secondary address. The secondary address is separated from the primary address by a plus sign (+).

Only the lower five bits of each address are significant. These bits can be in the range from 0 through 30 for both the primary and the secondary addresses. Therefore, the binary value 01100010 (decimal 98) is interpreted as decimal 2.

The following examples all specify a primary address of 0 and a secondary address of 2. The listen address is 32, the talk address is 64, and the secondary address is 2 or 98, which are equivalent.

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

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-422CT 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 will be posted.

Example:

```
1. 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 use `wait` to monitor selected GPIB events and to delay any further GPIB-422CT 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 or cleared to wait or not to wait, respectively, 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-422CT monitors GPIB activity. When any event corresponding to the bits set in `mask` occurs, the GPIB-422CT returns status information indicating its current status. If continuous status reporting has been enabled, status will be reported in the format requested. If continuous status has not been enabled, status will be returned in numeric format.

You could use `wait`, for example, if you wish 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`, then 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-422CT from waiting indefinitely for SRQ to be asserted, set the SRQI and TIMO bits by setting the mask equal to `4096 + 16384`. This will cause the wait to terminate either on SRQI or TIMO, whichever occurs first.

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 with `tmo`, timeouts for this function are disabled. You should disable timeouts only when you are certain the selected event will occur; otherwise, the GPIB-422CT waits indefinitely for the event to occur.

If you call `wait` without an argument, the GPIB-422CT records the EARG error.

See Also: `stat` and `tmo`.

Examples:

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.
```

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-422CT 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]<CR>`
`data<CR>`

Purpose: You use `wrt` to send data over the GPIB.

Remarks: The argument `count` is a numeric string preceded by a number sign (`#count`). The string specifies a number between 1 and 65535 and must not contain a comma. `#count` specifies the number of data bytes to send. The number of data bytes must not include the carriage return that indicates the end of the programming message.

The argument `data` is a string of 8-bit characters that are transferred without any translation to the GPIB.

The argument `alist` is a list of `addrs` separated by commas or spaces. `addrs` are addresses that specify the GPIB addresses of the Listener (or Listeners, if more than one address is given).

A device address consists of a primary address and an optional secondary address. The secondary address is separated from the primary address by a plus sign (+).

Only the lower five bits of each address are significant. These bits can be in the range from 0 through 30 for both the primary and the secondary addresses. Therefore, the binary value 01100010 (decimal 98) is interpreted as decimal 2.

The following examples all specify a primary address of 0 and a secondary address of 2. The listen address is 32, the talk address is 64, and the secondary address is 2 or 98, which are equivalent.

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

When `#count` is not specified, the GPIB-422CT recognizes the end of the data string when it encounters a carriage return

or a linefeed. `#count` is required when your data string contains embedded carriage return or linefeed characters.

If you specify an address list, the GPIB-422CT 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-422CT sends Interface Clear (IFC) to make itself CIC.

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 will be posted.

If you do not give an `alist` and the GPIB-422CT is not CIC, it assumes it will be addressed by the Controller. If you do not give an `alist` and the GPIB-422CT is CIC, it addresses itself as Talker and assumes the Listeners are already addressed.

The first part of this programming message, up to <CR>, is buffered, meaning the GPIB-422CT will not act upon it until it receives <CR>. The string that follows the first line is piped to the GPIB, non-buffered. This allows you to send a string larger than the GPIB-422CT's internal buffer with one programming message.

The GPIB-422CT aborts the GPIB write if it receives Device Clear or Selected Device Clear and its Listen Address. The GPIB-422CT aborts the GPIB write and records the EABO error if, at any time during the GPIB write, the time limit set for I/O functions expires. This limit is 10 seconds unless you use `tmo` to change it. The GPIB-422CT also aborts the `wrt` and records the ENOL error if there are no addressed Listeners when it begins to send data.

See Also: `tmo` for timeout information, Appendix B for more error information, and `spign` for serial port error handling information.

Examples:

```
1. PRINT #1,"wrt #50 9+97" 'Write 50 bytes to
   FOR I = 1 TO 50         'device at primary
   PRINT #1,CHR$(A(I));    'address 9 and
   NEXT I                  'secondary address 97.
   PRINT #1,CHR$(13);      'Send carriage return.

2. PRINT #1,"wrt 2"     'Write the data bytes ABCDE
   PRINT #1,"ABCDE"     'at device at address 2.
```

xon - Change Serial Port XON/XOFF Protocol

Type: Serial Port function

Syntax: `xon [booltx][,boolrx]<CR>`

Purpose: You use `xon` at the beginning of your program to configure the GPIB-422CT 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 a 1, the GPIB-422CT 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 will immediately stop 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 may 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 a 1, and the GPIB-422CT is receiving data over the serial port, it sends XOFF over the serial port (if its serial receive buffer is almost full). This tells the sender to stop sending data. When the GPIB-422CT serial port receive buffer has room to safely receive more bytes, the GPIB-422CT 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 it, you could be in danger of overflowing either the GPIB-422CT or your computer's internal buffer.

When would you want to enable the protocol in one case and not the other? Some computers use XON/XOFF protocol when transmitting data but not when receiving data. In this

case you might configure the GPIB-422CT using the second example.

Notice that the comma must precede the argument in this case.

The power-on default is that XON/XOFF for both cases is disabled.

If you call `xon` without an argument, the GPIB-422CT returns to you the current settings (1=protocol enabled, 0=protocol disabled).

Examples:

```
1. PRINT #1,"XON 1,1"      'Enable GPIB-422CT
                           'XON/XOFF protocol for
                           'TX and RX.

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

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

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

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

Chapter 6
Installation and Configuration in G Mode

If you plan to operate in G mode, use this chapter to install and configure the GPIB-422CT. Then read Chapters 7 and 8 to learn about the programming messages.

Installation

There are five basic steps to installing the GPIB-422CT.

1. Inspect the GPIB-422CT for damage that may have been caused in shipment.
2. Verify the voltage requirement.
3. Configure the operating parameters.
4. Connect the cables.
5. Power on the unit.

These steps are described in more detail in the following sections.

Step 1: Inspection

Before you install the GPIB-422CT, inspect the shipping container and its contents for damage. If damage appears to have been caused in shipment, file a claim with the carrier. Retain the packing material for possible inspection and/or reshipment.

If the equipment appears to be damaged, do not attempt to operate it. Contact National Instruments for instructions.

Step 2: Verify the Voltage Requirement

The GPIB-422CT is shipped from the factory with either a 115V or 230V wall-mount supply. Verify that the voltage on the supply matches the voltage that is supplied in your area.

Caution: Operating the unit at any voltage other than the one specified could damage the unit.

Step 3: Configure the Operating Parameters

The GPIB-422CT is shipped from the factory configured to operate in S mode. To configure the GPIB-422CT to operate in G mode, follow these steps:

1. Disconnect power to the unit and disconnect any cables that may be connected to the unit.
2. Unscrew the two screws on the opposite sides of the rear panel.
3. Grab the rear panel bezel and pull it straight away from the rest of the unit. The card should slide out the back of the enclosure.
4. To change the electrical specifications of the RS-422 port so that it is compatible with the Apple Macintosh serial port, locate the three jumpers at location W4 through W6. Move all three jumpers so that they rest across the pins labeled 1 and 2 (see Figures 6-1 and 6-2).

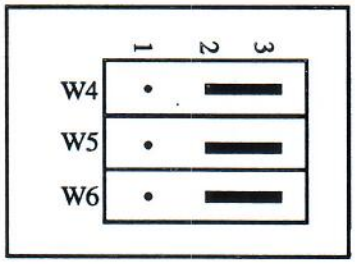


Figure 6-1. Standard RS-422 Configuration (Factory Setting)

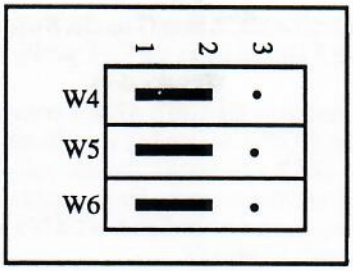


Figure 6-2. Macintosh Serial Port Configuration

5. Locate the configuration DIP switches (U20 and U22) on the printed wire board.

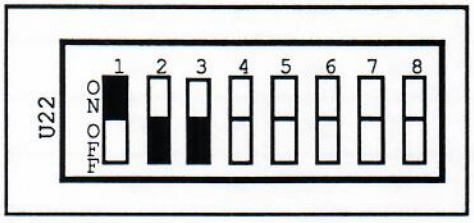
6. Set the switches for the desired mode of operation. Refer to the following section, *Set Configuration Switches.*

   Caution: Most of the circuitry in the GPIB-422CT uses advanced CMOS technology and can be damaged by static electricity. Avoid touching any of the components and take any necessary CMOS handling precautions.

7. Close the unit and re-insert the screws removed in Step 2.

Set Configuration Switches

The DIP switch at location U22 on the printed wire board is used to configure the GPIB address of the GPIB-422CT while in G mode. The DIP switch, shown in Figure 6-3, has eight configuration switches.



ON indicates that the switch is pressed on the "ON" side.

OFF indicates that the switch is pressed on the "OFF" side.

Figure 6-3. G Mode Switch Settings

Notice that to put the GPIB-422CT into G mode, Switch 1 of U22 should be ON and Switches 2 and 3 should be in the OFF position.

Connecting the serial device to the GPIB-422CT converts the serial device into a GPIB device. Each GPIB device has a GPIB address. The GPIB-422CT uses two GPIB addresses—the one you select by setting the configuration switches, and the second address equal to the address you select plus one. The first becomes the GPIB-422CT's address. The second becomes the serial device's address.

The GPIB addresses of the GPIB-422CT and the serial device must be different from the addresses of the other GPIB devices in your system. Determine the GPIB addresses of the other GPIB devices; then, using the list on the following page, select two consecutive addresses that are not already in use.

Configure the GPIB-422CT primary address by setting the switches as shown in Table 6-1. The corresponding talk and listen addresses are also shown.

Table 6-1. Primary Address Configurations

Switches					Primary Address	Listen Address Dec/ASCII		Talk Address Dec/ASCII	
4	5	6	7	8					
OFF	OFF	OFF	OFF	OFF	0	32	SP	64	@
OFF	OFF	OFF	OFF	ON	1	33	!	65	A
OFF	OFF	OFF	ON	OFF	2	34	"	66	B
OFF	OFF	OFF	ON	ON	3	35	#	67	C
OFF	OFF	ON	OFF	OFF	4	36	$	68	D
OFF	OFF	ON	OFF	ON	5	37	%	69	E
OFF	OFF	ON	ON	OFF	6	38	&	70	F

(continues)

Table 6-1. Primary Address Configurations (continued)

Switches 4	5	6	7	8	Primary Address	Listen Address Dec/ASCII		Talk Address Dec/ASCII	
OFF	OFF	ON	ON	ON	7	39	'	71	G
OFF	ON	OFF	OFF	OFF	8	40	(	72	H
OFF	ON	OFF	OFF	ON	9	41	)	73	I
OFF	ON	OFF	ON	OFF	10	42	*	74	J
OFF	ON	OFF	ON	ON	11	43	+	75	K
OFF	ON	ON	OFF	OFF	12	44	,	76	L
OFF	ON	ON	OFF	ON	13	45	-	77	M
OFF	ON	ON	ON	OFF	14	46	.	78	N
OFF	ON	ON	ON	ON	15	47	/	79	O
ON	OFF	OFF	OFF	OFF	16	48	0	80	P
ON	OFF	OFF	OFF	ON	17	49	1	81	Q
ON	OFF	OFF	ON	OFF	18	50	2	82	R
ON	OFF	OFF	ON	ON	19	51	3	83	S
ON	OFF	ON	OFF	OFF	20	52	4	84	T
ON	OFF	ON	OFF	ON	21	53	5	85	U
ON	OFF	ON	ON	OFF	22	54	6	86	V
ON	OFF	ON	ON	ON	23	55	7	87	W

(continues)

Table 6-1. Primary Address Configurations (continued)

Switches 4	5	6	7	8	Primary Address	Listen Address Dec/ASCII		Talk Address Dec/ASCII	
ON	ON	OFF	OFF	OFF	24	56	8	88	X
ON	ON	OFF	OFF	ON	25	57	9	89	Y
ON	ON	OFF	ON	OFF	26	58	:	90	Z
ON	ON	OFF	ON	ON	27	59	;	91	[
ON	ON	ON	OFF	OFF	28	60	<	92	\
ON	ON	ON	OFF	ON	29	61	=	93	]
ON	ON	ON	ON	OFF	30	62	>	94	^

Example Settings for Configuration Switch

Figure 6-4 shows an example setting of the configuration switch. Switches 1, 2, and 3 are ON, OFF, and OFF, respectively, which indicates that the GPIB-422CT is operating in G mode. Switches 4 through 8 are OFF, OFF, ON, OFF, and ON, respectively. This configures the GPIB-422CT at GPIB address 5 and the serial device at address 6.

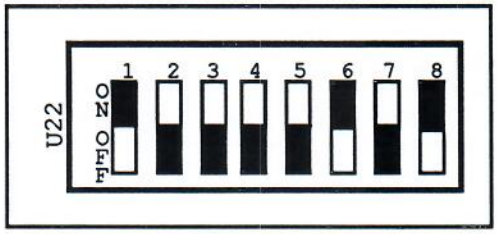


Figure 6-4. Example Settings for Switch U22

Note: Switch U20 is not used by S or G mode, but can be defined and used by user applications in IBCL mode.

Step 4: Connect the Cables

Connect the cables as follows:

1. Connect the serial cable to the GPIB-422CT and securely fasten the holding screws. Connect the other end of the cable to your serial device. Be sure to use only shielded serial cable, and obey all RS-422 cabling restrictions.

2. Connect the GPIB cable to the GPIB-422CT and tighten the thumb screws on the connector. Connect the other end to your GPIB system. Be sure to obey all IEEE-488 cabling restrictions, and use only double-shielded GPIB cable.

3. Connect the power jack of the wall-mount power supply to the power receptacle on the back panel of the GPIB-422CT, then plug the supply into an AC outlet of the correct voltage.

Step 5: Power on the Unit

Power on your GPIB-422CT by using the front panel rocker switch. The POWER LED should come on immediately. The READY indicator should come on after the GPIB-422CT has passed its power-on self-test, indicating the unit is ready for operation.

If the READY indicator 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 light still fails to come on, contact National Instruments for further instructions.

Chapter 7
Programming in G Mode

This chapter shows how to program the GPIB-422CT in G mode using programming messages. It describes programming messages, their format, and how they are processed, along with the functions and function arguments that make up the programming messages.

This chapter also explains how to communicate with your serial device through the GPIB-422CT.

Programming Messages

You program the GPIB-422CT by sending it programming messages (which are ASCII strings) by way of its GPIB port.

Programming Message Format

The programming message consists of a function name, one or more arguments (optional), followed by a carriage return (<CR>), or a linefeed (<LF>), or a carriage return followed by a linefeed <CR><LF>. Carriage return and linefeed can be expressed in BASIC as `CHR$(13)` and `CHR$(10)`, respectively.

Example of a Programming Message

In the following lines of BASIC code:

```
WRT$ ="eos x,10"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

`WRT$` contains the programming message in which `eos` is the function name, `x` and `10` are the arguments, and `CHR$(13)` is the terminating carriage return. This programming message tells the GPIB-422CT to assert the EOI line when it sends the end-of-string character linefeed. The second line of the example is a GPIB-PC function call provided as part of a National Instruments product that allows a personal computer to control the

GPIB from Microsoft BASIC. This function outputs the string in `WRT$` to the device GPIB-422CT.

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

How Messages are Processed

The GPIB-422CT processes each programming message on a line-by-line basis. The GPIB-422CT buffers the entire message, interprets the function name and arguments, then executes the message.

Function Names

The function names have been selected to indicate each function's purpose, thereby making your programs easy to understand. However, if you wish 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 **boldface** in the function tables and in the syntax portions of the function descriptions.

Function Argument Delimiters

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 8, the square brackets (`[]`) that enclose some arguments indicate that those arguments are optional. Do not enter the brackets as part of your arguments.

Abbreviation for Argument

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

Status Information

The function descriptions in Chapter 8 explain that the GPIB-422CT *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-422CT *returns* to you certain information. This means that the GPIB-422CT sends information to you over the GPIB.

Communicating with the GPIB-422CT and the Serial Device

The GPIB-422CT knows the type of GPIB data it is processing by using dual addressing. With dual addressing, the GPIB-422CT recognizes two different GPIB addresses. The first is the GPIB-422CT's address. The second is the serial device's address.

Address of the GPIB-422CT

The address of the GPIB-422CT is the primary address you set on the configuration switch, with secondary addressing disabled.

Address of the Serial Device

The address of the serial device is the GPIB-422CT primary address plus 1, with secondary addressing disabled. However, if you select a primary address of 30 with the configuration switch, the serial device will be at address 0.

The two primary addresses recognized by the GPIB-422CT are referred to as the GPIB-422CT address and the serial device address.

Addressing Terminology

When the GPIB-422CT receives the serial device address, the data it sends and receives is referred to as serial data.

When the GPIB-422CT 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 GPIB-422CT and Serial Device as Listener

When the GPIB-422CT receives its own listen address, it examines the data received over the GPIB, treats it as a programming message or messages, and takes actions based on that data.

When the GPIB-422CT receives the serial device listen address, it forwards the data received over the GPIB to the serial port without examining the data for meaning.

For example, let's say you have a serial printer connected to the GPIB-422CT and you wish 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-422CT to watch for XON/XOFF characters from the printer.

Do this by first addressing the GPIB-422CT to listen by sending its listen address. Then, send it the programming message `xon ,1`. The GPIB-422CT interprets this programming message and acts upon it without sending any data on to the serial device.

Next, address the serial device to listen by sending to the GPIB-422CT the serial device listen address. Then, send the data over the GPIB. The GPIB-422CT sends that data on to the printer without examining that data for meaning. The LISTEN LED on the GPIB-422CT front panel is lit when either the GPIB-422CT is addressed to listen or the serial device is addressed to listen.

The GPIB-422CT and Serial Device as Talker

When the GPIB-422CT receives its own talk address, it sends out status information.

When the GPIB-422CT receives the serial device talk address, it sends all data received from the serial device to the GPIB.

For example, let's say your serial device is programmed to perform some calculations and you want it to return data to you.

First, address the serial device to talk by sending to the GPIB-422CT the serial device talk address. Next, perform a GPIB read of 100 bytes. The GPIB-422CT retrieves 100 bytes from its serial port receive buffer and sends them to you.

Now, find out if the serial device has sent more bytes to the GPIB-422CT by asking the GPIB-422CT to send you status information. Do this by sending the GPIB-422CT its listen address and the programming message `stat n`. Then, send the GPIB-422CT its talk address and perform a GPIB read of 20 bytes. The GPIB-422CT sends you its status information, terminated by the GPIB END message.

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

If the GPIB-422CT receives its talk address but has nothing to send, it will respond to GPIB reads with a carriage return and a linefeed, accompanied by END. If the GPIB-422CT receives the serial device talk address but has no data in its serial port receive buffer to send, it will wait for data from the serial device to fill the request.

The TALK LED on the GPIB-422CT's front panel is lit when either the GPIB-422CT is addressed to talk or the serial device is addressed to talk.

GPIB Read and Write Termination (END and EOS)

You program the GPIB-422CT and the serial device to Talk in order to send status information and serial data over the GPIB. You program the GPIB-422CT and the serial device to Listen in order to receive programming messages and serial data from the GPIB.

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 in advance the number of bytes in the transmission.

END message	The Talker asserts the EOI (End or Identify) signal simultaneously with transmission of the last data byte. The Listener stops reading when it detects a data message accompanied by EOI, regardless of the value of the byte.
EOS character	The Talker uses a special character at the end of its data string. The Listener stops receiving data when it detects that character. Either a 7-bit ASCII character or a full 8-bit binary byte can be used.

The two methods can be used individually or in combination. It is important that the Listener be configured to detect the end of a transmission.

When the GPIB-422CT receives its own talk or listen address, no EOS modes are in effect. When talking, the GPIB-422CT asserts EOI with the last byte of its response.

When the GPIB-422CT 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-422CT checks the data received from the serial device for errors, while buffering data. If a serial port error occurs, the GPIB-422CT records the appropriate error code in its status area. To determine if a serial port error has occurred, use `stat` to request the GPIB-422CT status information. After the serial error code has been reported, it will be reset automatically so that no further action to the GPIB-422CT is necessary.

You can also determine if a serial port error has occurred by performing a serial poll of the GPIB-422CT and checking the serial poll response byte to see if its ESDR bit is set. Refer to the section on serial poll responses at the end of this chapter.

G Mode - Default Settings

Tables 7-1 and 7-2 list power-on characteristics of the GPIB-422CT and the functions you can use to change those characteristics.

Table 7-1. GPIB Characteristics

Characteristic	Power-on Value	Function
End-of-String modes	none	`eos`
Allow GPIB-422CT to assert SRQ	no	`srqen`

Table 7-2. Serial Port Characteristics

Characteristic	Power-on Value	Function
Echo bytes to serial port	no	`echo`
Enable serial port communication	yes	`onl`
Baud rate	9600	`spset`
Parity	none	`spset`
Data bits	8	`spset`
Stop bits	1	`spset`

(continues)

Table 7-2. Serial Port Characteristics (continued)

Characteristic	Power-on Value	Function
Send XON/XOFF	no	`xon`
Recognize XON/XOFF	no	`xon`
Report serial errors	no	`spign`

List of G Mode Functions by Group

Tables 7-3 through 7-5 contain the programming messages that are sent to the GPIB-422CT from a GPIB Talker to configure the GPIB-422CT.

The GPIB-422CT functions are divided into three groups: GPIB Configuration functions, Serial Port Configuration functions, and General Use functions.

GPIB Configuration Functions

Table 7-3. GPIB Configuration Functions

Function	Purpose
`eos` modes, `eoschar`	Change or disable GPIB end-of-string termination mode
`srqen` mask	Set conditions for asserting SRQ

Serial Port Configuration Functions

Table 7-4. Serial Port Configuration Functions

Function	Purpose
`echo` on/off	Echo characters received from serial port
`spset` modes	Change serial port parameters
`xon` modes	Change serial port XON/XOFF protocol
`spign` on/off	Ignore serial port errors

General Use Functions

Table 7-5. General Use Functions

Function	Purpose
`id`	Identify system
`onl` on/off	Place the GPIB-422CT online/ offline
`stat` options	Return GPIB-422CT status

List of G Mode Functions in Alphabetical Order

Table 7-6 is an alphabetical list of G mode functions.

Table 7-6. GPIB-422CT 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-422CT online/offline
`spign` on/off	Ignore serial port errors
`spset` modes	Change serial port parameters
`srqen` mask	Set conditions for asserting SRQ
`stat` options	Return GPIB-422CT status
`xon` modes	Change serial port XON/XOFF protocol

Operation of the GPIB-422CT with a Serial Device

The serial device attached to the GPIB-422CT looks like any other GPIB device and, as such, is configured to respond in certain ways to GPIB commands.

Serial Poll

The GPIB-422CT will respond to a serial poll of the serial device by placing a serial poll response byte on the data lines.

Table 7-7 contains the meanings of the bits in the serial poll response byte.

Table 7-7. 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 - EARG, ECAP, ECMD
3	SERR	Serial error - EPAR, EORN, EFRM
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	RSV	Request service
7	—	Not Used

Service Request

The GPIB-422CT can be programmed to assert Service Request (SRQ) in a variety of cases. After power-on, the GPIB-422CT defaults to never asserting service request. You can program the GPIB-422CT to assert SRQ under the following circumstances:

- When the GPIB-422CT serial port receive buffer is full.

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

Note: You must use the `srqen` function to enable the GPIB-422CT to assert SRQ under the conditions listed above.

Parallel Poll

The GPIB-422CT responds to a parallel poll by driving the line indicated by the PPE command true if the value of its individual status bit matches the sense bit of the PPE command and false otherwise.

Trigger

This has no effect on the GPIB-422CT.

Local

This has no effect on the GPIB-422CT.

Control

In G mode, the GPIB-422CT can act only as a Talker/Listener. It does not make sense for the GPIB-422CT to be passed control, since all programming instructions must be sent to it from another GPIB device. However, the GPIB circuitry in the GPIB-422CT will allow it to be passed control, in which case the GPIB-422CT will immediately assert ATN. This is an error condition that can lock up your system and should therefore be avoided.

Device Clear

When the GPIB-422CT receives the universal Device Clear (DCL) command or when it receives its listen addresses and the Selected Device Clear (SDC) command, it clears both its status buffer and its serial port receive buffer.

Chapter 8
G Mode Functions

This chapter contains descriptions of the G mode functions that you use to program the GPIB-422CT. These functions are in alphabetical order and are formatted to provide you with an easily usable reference.

Points to Remember

- The program examples within the function descriptions are written in Microsoft BASIC Version 3.0, using the GPIB-PC function calls of National Instruments.

- The GPIB-PC function call:

```
WRT$="SPSET"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

  automatically sends to the GPIB-422CT its talk address, and the programming message `spset`, followed by a carriage return. If you are not using the National Instruments GPIB-PC software, be sure your program properly addresses the GPIB-422CT 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.

- Terminate each programming message 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.

- To send more than one programming message per 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 either send two separate strings to the GPIB in two separate GPIB writes:

```
WRT$="SPSET 1200,n,8"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
WRT2$="XON 1,1"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

or, you could put both messages in one string and send it to the GPIB-422CT in one GPIB write:

```
WRT$="SPSET 1200,n,8"+CHR$(13)+"XON
1,1"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

- It is necessary for you to send only enough characters of the function name to distinguish it from other functions. These characters are shown in **boldface** in the syntax portion of the function descriptions.

echo - Echo Characters Received from Serial Port

Type: Serial Port Configuration function

Syntax: **ec**`ho [bool]<CR>`

Purpose: You use `echo` when a terminal is connected to the GPIB-422CT and you wish to display what you type on the screen of the terminal.

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 you call `echo` without an argument, the GPIB-422CT returns the current setting.

By default, echoing is disabled.

In a debugging environment where the success of your communication with the serial device is unclear, you could connect a terminal to the GPIB-422CT instead of connecting the serial device to the GPIB-422CT. Now the data that the GPIB-422CT normally sends to the serial device is displayed on the terminal screen. Also, you can type characters on the terminal to send to the GPIB-422CT, just as your serial device sends characters to the GPIB-422CT. With `echo` enabled, the GPIB-422CT echoes back to the terminal what you type, so that you can verify that what the GPIB-422CT receives is exactly what you type.

Examples:

```
1. WRT$="echo 1"+CHR$(13)          'Enable character
   CALL IBWRT(GPIB422%,WRT$)       'echoing.

2. WRT$="ECHO 0"+CHR$(13)          'Disable character
   CALL IBWRT(GPIB422%,WRT$)       'echoing.

3. WRT$="echo"+CHR$(13)            'What is the
   CALL IBWRT(GPIB422%,WRT$)       'current echo
   CALL IBRD(GPIB422%,RESP$)       'status?
                                   'RESP$ contains
                                   '0<CR><LF>
                                   '(character echo is
                                   'disabled).
```

eos - Change/Disable GPIB EOS Termination Mode

Type: GPIB Configuration function

Syntax: **eos** `[[X][B]eoschar]`<CR>

or

eos `D`<CR>

Purpose: You use `eos` to enable the GPIB-422CT to add the GPIB END message to the data string sent by the serial device when the data string contains the specified end-of-string character.

Remarks: `eos` applies only when the GPIB-422CT has received the serial device talk address and is sending serial data to the GPIB. It does not apply when the GPIB-422CT 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.

Table 8-1. Data Transfer Termination Methods

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-422CT automatically sends the END message along with `eoschar`

when performing GPIB writes of serial data. That is, when the GPIB-422CT receives `eoschar` over the serial port and sends it on to the GPIB, it will also assert 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-422CT returns to you the current `eos` settings.

The assignment made by this function remains in effect until you call `eos` again, you call `onl`, or you turn the GPIB-422CT off.

See Also: Chapter 7, the sections entitled *GPIB Read and Write Termination*, and *Operation of the GPIB-422CT with a Serial Device;* and the `srqen` function in this chapter.

Examples:

```
1. WRT$="EOS X,13"+CHR$(13)      'Send EOI with <CR>.
   CALL IBWRT(GPIB422%,WRT$)     'Compare 7 bits.

2. WRT$="eos"+CHR$(13)           'What are the
   CALL IBWRT(GPIB422%,WRT$)     'current EOS
                                 'settings?
   CALL IBRD(GPIB422%,RESP$)     'RESP$ contains
                                 'X,13<CR><LF>.
```

id - Identify System

Type: General Use function

Syntax: `id<CR>`

Purpose: You use `id` if you wish to know the revision level of your software, or if you wish to know how much RAM is installed in your GPIB-422CT.

Remarks: The identification is returned in three strings. The first two strings identify the company product model, the software revision level, and a copyright notice. The third string identifies the number of bytes of RAM in the GPIB-422CT.

Examples:

1.
```
WRT$="id"+CHR$(13)      'Get system
                        'identification.
CALL IBWRT(GPIB422%,WRT$)
```

typical response:
```
GPIB-422CT Rev. A.2<CR><LF>
(c)1990 National Instruments<CR><LF>
64K bytes RAM<CR><LF>
```

onl - Place the GPIB-422CT Online/Offline

Type: General Use function

Syntax: `onl [bool]<CR>`

Purpose: You use `onl` to disable communication between the GPIB-422CT and the serial port, or to re-initialize the GPIB-422CT characteristics to their default values.

Remarks: If the argument `bool` is 1, the GPIB-422CT places itself online. If the argument `bool` is 0, the GPIB-422CT places itself offline. By default, the GPIB-422CT powers up online.

If you call `onl` without an argument, the GPIB-422CT returns the current status of the GPIB-422CT, which is 0 if the GPIB-422CT is offline and 1 if the GPIB-422CT is online.

Placing the GPIB-422CT offline can be thought of as disconnecting its serial cable from the serial device. When placed offline, no data is sent out the GPIB-422CT serial port and data coming in to the GPIB-422CT serial port is not saved until `onl` is called with `bool=1`.

Placing the GPIB-422CT online enables data to be sent and received over its serial port. Placing the GPIB-422CT online also restores all GPIB-422CT settings to their power-on values.

See Also: Tables 7-1 and 7-2 for the GPIB-422CT power-on settings.

Examples:

```
1. WRT$="onl 1"+CHR$(13)          'Put the GPIB-422CT
   CALL IBWRT(GPIB422%,WRT$)      'online, and restore
                                  'its power-on
                                  'settings.

2. WRT$="ONL 0"+CHR$(13)          'Put the GPIB-422CT
   CALL IBWRT(GPIB422%,WRT$)      'offline.
```

spign - Ignore Serial Port Errors

Type: Serial Port function

Syntax: `spign [bool]<CR>`

Purpose: You use `spign` at the beginning of your program if you wish 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-422CT to ignore or not to ignore the occurrence of serial port errors. By default, the GPIB-422CT ignores serial port errors.

Remarks: If the argument `bool` is 0, the GPIB-422CT will not ignore serial port errors. When `bool` is 0, the GPIB-422CT does not store characters that contain serial errors. A list of serial port errors are listed in Appendix B.

The serial port errors include parity, overrun, framing, and overflow errors.

If the argument `bool` is 1, the GPIB-422CT stores all serial port errors occur. Also, no serial error code will be reported by the `stat` function.

Examples:

1.
```
WRT $,"spign 0"+CHR$(13)     'Do not ignore
CALL IBWRT(GPIB422%,WRT$)    'serial port errors.
```

2.
```
WRT $,"spign 1"+CHR$(13)     'Ignore serial port
CALL IBWRT(GPIB422%,WRT$)    'errors.
```

spset - Change Serial Port Parameters

Type: Serial Port Configuration function

Syntax: **spset** `[baud] [parity] [databits] [stopbits]<CR>`

Purpose: You use `spset` at the beginning of your program to set up the GPIB-422CT serial port characteristics (baud rate, parity, data bits, and stop bits) to 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).

Until you call `spset` the following characteristics are in effect:

```
9600 n 8 1
```

meaning the baud rate is 9600 bits per second (bps), parity is disabled, there are 8 data bits and 1 stop bit.

If you must reconfigure the GPIB-422CT's serial port, do so only when communication with the serial port is not taking place.

If you call `spset` without an argument, the GPIB-422CT returns to you its current serial port configuration.

Examples:

1.
```
REM Set up the serial port of GPIB-422CT to keep
REM its current baud rate, current parity, and
REM to use 7 data bits and 2 stop bits.
WRT$="spset 7 2"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

2.
```
REM What are the current GPIB-422 serial port
REM settings?
WRT$="SPSET"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
REM RESP$ will contain 19200,E,7,2<CR><LF>
REM (19200 baud, even parity, 7 data bits,
REM 2 stop bits).
CALL IBRD(GPIB422%,RESP$)
```

3.
```
REM Set the GPIB-422CT serial port to 1200 baud,
REM no parity, 8 data bits, and 1 stop bit.
WRT$="spset 1200 n 8 1"+CHR$(13)
CALL IBWRT(GPIB422%,WRT$)
```

srqen - Enable/Disable Setting of SRQ

Type: GPIB Configuration function

Syntax: `srqen [mask]<CR>`

Purpose: You use `srqen` when you wish to allow the GPIB-422CT to assert SRQ under the conditions described in Chapter 7 in the section entitled *Operation of the GPIB-422CT with a Serial Device.*

Remarks: When the argument `mask` is 0, the GPIB-422CT will never assert SRQ. When the argument `mask` is > 0, the GPIB-422CT will assert SRQ under the conditions represented by each bit in the `mask`. The `mask` bits are as follows:

Table 8-2. SRQ Mask Bits

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 bit 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 will 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 will never be asserted.

If you call `srqen` without an argument, the GPIB-422CT returns a decimal string that indicates the decimal value of the current setting.

See Also: `eos`.

Examples:

```
1. WRT$="srqen 0"+CHR$(13)        'Never assert SRQ.
   CALL IBWRT(GPIB422%,WRT$)

2. WRT$="srqen 4"+CHR$(13)        'Assert SRQ when
   CALL IBWRT(GPIB422%,WRT$)      'ENOL occurs.
```

stat - Return GPIB-422CT Status

Type: General Use function

Syntax:
```
stat [[c] n]<CR>
   or
stat [c] s<CR>
   or
stat [c] n s<CR>
```

Purpose: You use `stat` to obtain the status of the GPIB-422CT to see if certain conditions are currently present. You use `stat` most often to see if the previous operation resulted in an error.

Remarks: The GPIB-422CT returns status information to you in a form depending on the mode or combination of modes you chose. `n` indicates that the status information will be returned as numeric strings. `s` indicates that the status information will be returned in symbolic format, that is, as mnemonic strings. `c` specifies that the status will be returned after each programming message, eliminating the need to call `stat` after each programming message.

Normally, you 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.

The status information returned by the GPIB-422CT contains four pieces of information: the GPIB-422CT status, a GPIB-error code, a serial-error code, and a count. The GPIB-422CT returns a <CR><LF> following each piece of the response. It asserts EOI with the final <LF> that comes after `count`.

Status represents a combination of GPIB-422CT conditions. Internally in the GPIB-422CT, 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. Since more than one GPIB-422CT 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-422CT detects either a GPIB error or a serial port error.

Consequently, when status is negative, an error condition exists; when status is positive, no error condition exists.

GPIB-error represents a single GPIB error condition present.

serial-error represents a single serial error condition present.

count is the number of bytes currently contained in the GPIB-422CT's serial port receive buffer.

Table 8-3. GPIB-422CT Status Conditions

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

(continues)

Table 8-3. GPIB-422CT Status Conditions (continued)

Numeric Value (n)	Symbolic Value (s)	Description	Bit
4	—	Reserved	2
2	—	Reserved	1
1	—	Reserved	0

Table 8-4. GPIB Error Conditions

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 8-5. Serial Error Conditions

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 B.

The GPIB-422CT updates `status` and `count` at the end of each programming message. It updates `GPIB-error` and `serial-error` whenever a new error occurs. `GPIB-error` and `serial-error` are cleared only after you have requested status.

Refer to the following examples for ways in which to use the status information.

When you wish 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.

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

Examples:

1.
```
 10 REM Tell GPIB-422CT to send us numeric
 20 REM status.
 30 WRT$="stat n"+CHR$(13)
 40 CALL IBWRT(GPIB422%,WRT$)
 50 REM Now read the status from the
 60 REM GPIB-422CT.
 70 STATUS$=SPACE$(10) : GPIBERR$=SPACE$(10)
 80 SPERR$=SPACE$(10) : COUNT$=SPACE$(10)
 90 CALL IBRD(GPIB422%,STATUS$)
100 REM Read up to 10 bytes of status.
110 REM The GPIB-422CT returns 4 pieces of
120 REM status, one for each IBRD.  We are set
130 REM up to terminate read on linefeed, which
140 REM is what terminates each piece of status.
150 STATUS% = VAL(STATUS$)
160 CALL IBRD(GPIB422%,GPIBERR$)  'Read GPIB-error.
170 CALL IBRD(GPIB422%,SPERR$)    'Read serial-error.
180 CALL IBRD(GPIB422%,COUNT$)    'Read count.
190 REM Call error routine at 500 if error occurred.
200 IF STATUS% < 0 THEN GOTO 500
    ...
500 REM Print GPIB-error, and serial-error
510 REM values to determine what errors occurred.
520 PRINT "GPIB-error = ";GPIBERR$
530 PRINT "serial-error = ";SPERR$
540 STOP
```

2.
```
 10 REM Turn on continuous status reporting,
 20 REM in numeric format.
 30 WRT$="stat c n"
 40 CALL IBWRT(GPIB422%,WRT$)
 50 REM If we have 3 bytes in the serial port
 60 REM buffer, a typical response would be:
 70 REM 262<CR><LF>0<CR><LF>0<CR><LF>3<CR><LF>
 80 REM Read the GPIB-422CT status; read 30
 90 REM bytes or until EOI is received.
100 RD$=SPACE$(30)
110 CALL IBRD(GPIB422%,RD$)
120 REM Print the status information.
130 PRINT "GPIB-422CT status is: ";RD$
```

3.
```
 10  REM Turn on continuous status reporting,
 20  REM in symbolic format.
 30  WRT$="stat c s"
 40  CALL IBWRT(GPIB422%,WRT$)
 50  REM Read the GPIB-422CT status; read 50
 60  REM bytes or until EOI is received.
 70  RD$=SPACE$(50)
 80  CALL IBRD(GPIB422%,RD$)
 90  REM Print the status information.
100  PRINT "GPIB-422CT status is: ";RD$
```

```
Printed information is:

GPIB-422CT status is : CMPL
NGER
NSER
3
```

xon - Change Serial Port XON/XOFF Protocol

Type: Serial Port Configuration function

Syntax: `xon [booltx][,boolrx]<CR>`

Purpose: You use `xon` at the beginning of your program when your serial device uses the XOFF/XOFF protocol. `xon` allows the GPIB-422CT to communicate over the serial port using the same 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-422CT 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.

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-422CT receives data over the serial port, it sends XOFF over the serial port (if its serial port recive buffer is almost full). This tells the sender to stop sending data. When the GPIB-422CT has room to safely receive more bytes, the GPIB-422CT sends XON over the serial port. This tells the sender to begin sending data again.

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 it, you are in danger of overflowing the serial port receive buffers of the GPIB-422CT and the serial device. Refer to your serial device's documentation to see if it uses a hardware handshake.

Why would you want to enable the protocol in one case and not the other? Some serial devices use XON/XOFF protocol when transmitting data but not when receiving. In this case, you might configure the GPIB-422CT using the second example.

The power-on default is that XON/XOFF for both cases is disabled.

If you call `xon` without an argument, the GPIB-422CT returns to you the current settings (1=protocol enabled, 0=protocol disabled).

Examples:

```
1. WRT$="XON 1,1"+CHR$(13)          'Enable GPIB-422CT
   CALL IBWRT(GPIB422%,WRT$)        'XON/XOFF protocol
                                    'for TX and RX.

2. WRT$="XON 0,1"+CHR$(13)          'Disable protocol on
   CALL IBWRT(GPIB422%,WRT$)        'TX; enable protocol
                                    'on RX.

3. WRT$="XON"+CHR$(13)              'Return current settings.
   CALL IBWRT(GPIB422%,WRT$)
   CALL IBRD(GPIB422%,RESP$)        'RESP$ contains
                                    '0,1<CR><LF>
                                    '(transmit protocol
                                    'disabled, receive
                                    'protocol enabled).
```

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP