GPIB Device Simulator EPROM Programming Manual

November 1993 Edition

Part Number 320638-01

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
(800) IEEE-488 (toll-free U.S. and Canada)
Technical support fax: (512) 794-5678

Branch Offices:
Australia 03 879 9422, Austria 0662 435986, Belgium 02 757 00 20,
Canada (Ontario) 519 622 9310, Canada (Québec) 514 694 8521,
Denmark 45 76 26 00, Finland 90 527 2321, France 1 48 65 33 70,
Germany 089 714 50 93, Italy 02 48301892, Japan 03 3788 1921,
Netherlands 01720 45761, Norway 03 846866, Spain 91 640 0085,
Sweden 08 730 49 70, Switzerland 056 27 00 20, U.K. 0635 523545

Contents

Figures

GPIB Device Simulator EPROM

You can simulate the operation of a GPIB instrument by replacing the EPROM inside a GPIB-232CT with the GPIB Device Simulator EPROM (part number 702295-01). The GPIB-232CT will then act as a GPIB instrument.

The Simulator functions like a GPIB oscilloscope in that it is capable of returning waveforms in different configurations: ASCII, 8-bit unsigned binary, and 16-bit signed binary. It is also capable of returning single values in ASCII such as those returned by a multimeter.

Setting the Address

The default address of the GPIB Device Simulator EPROM is set using the first five switches of the U20 dip switch inside the GPIB-232CT. All other switch settings are ignored.

The first 5 switches set the address. Switch 1 is the least significant bit (LSB) and switch 5 is the most significant bit (MSB). Figure 1 shows an example of the Simulator set to address 3.

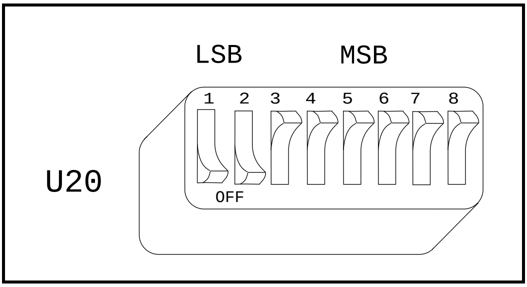


Figure 1. GPIB Device Simulator Set to Address 3

You change the address using the dip switches or by the sending the Simulator the `SADDR primary,secondary` command. For example, `SADDR 1` sets the Simulator to address 1, and `SADDR 2,4` sets the Simulator to primary address 2 and secondary address 4.

Data Formats

Waveform Format

The Simulator returns a 128 point waveform in either ASCII or binary. ASCII waveforms are preceded by the header CURVE. Binary waveforms are preceded by a pound sign (#) and the number of bytes that are in the waveform. All waveforms are terminated by a line feed `<LF>` character.

Floating Point ASCII (Default)

```
CURVE<space>num0,num1,...,num127<LF>
```

8-bit Unsigned Binary

```
#3128<Byte 0><Byte 1>...Byte<127><LF>
```

16-bit Signed Binary

```
#3256<MSB 0><LSB 0><MSB 1><LSB 1>...<MSB 127><LSB
127><LF>
```

Floating-Point Number Format

```
[+][-]1.2345E[+][-]0
```

Simulator Commands

The Simulator uses SCPI-like commands. The commands are shown in long form; however, the Simulator only accepts the short form of the command. In other words, only send the part of the command that is in uppercase characters. You can send multiple commands to the Simulator by separating them with a semicolon (;).

Address Command

`SADDRess primary, secondary`		Sets the address (power-on default – switch setting)
Example:	`SADDR 2`	Set the address to 2
	`SADDR 3, 4`	Set the primary address to 3 and the secondary address to 4

Waveform Format Commands

These commands format how the waveform data is returned by the Simulator.

`FORMat:DATA`	`ASCii`	Floating point (Default)
	`INTeger,8`	8-bit unsigned binary
	`INTeger,16`	16-bit signed binary
`FORMat:DATA?`		Returns the current waveform format

The following command changes the order of the bytes returned by `INTeger,16` encoding.

`FORMat:BORDer`	`NORMal`	Low byte first (Default)
	`SWAPped`	High byte first
`FORMat:BORDer?`		Returns the current format of the byte order

Example:	`FORM:DATA INT,16`	Set the waveform format as 16-bit integers
	`FORM:DATA?`	Query the current waveform format. For example, if the command was issued after the preceding command, it would return `FORM:DATA INT,16<LF>`

Waveform Generation Commands

These commands generate a 128 point waveform of the specified type. The number of cycles in the waveform is random. It can take 5 to 15 seconds to generate the waveform depending on the format and type of the waveform. Typically ASCII waveforms take longer than integer waveforms.

`SOURce:FUNCtion`	`SINusoid`	Sine waveform (Default)
	`SQUare`	Square waveform
	`NOISe`	Noisy sine waveform
	`RANDom`	Random noise waveform
	`PCHirp`	Chirp waveform
`SOURce:FUNCtion?`		Returns the current waveform type

Example:	`SOUR:FUNC SIN`	Generate a sinusoid waveform
	`SOUR:FUNC?`	Query the current waveform type. For example, if the command was issued after the preceding command, it would return `SOUR:FUNC SIN<LF>`

Waveform Query Commands

`SENSe:DATA?`	Returns the waveform data in the format specified by the waveform format commands
`SENSe:VOLTage:RANGe:OFFSet?`	Returns the Y offset for the waveform in ASCII floating point
`SENSe:VOLTage:RANGe?`	Returns the Y multiplier for the waveform in ASCII floating point
`SENSe:SWEep:TIME?`	Returns the X increment (1E-3) in ASCII floating point

`SENSe:VOLTage:HEADer?`	Returns all of the waveform scaling information in the format `OFFSET=x.xxxxE+x, RANGE=x.xxxxE+x, TIME=1E-3<LF>`

For integer-formatted waveforms, the offset and range is used to scale the raw integer data–for example,
`scaled.point(i) = (waveform.point(i) + offset) * range`

Example:	`SENS:DATA?`	Query Simulator for the waveform
	`SENS:VOLT:HEAD?`	Query Simulator for the waveform scaling information

"Multimeter Configuration" Commands

These commands simulate the operation of a meter. They return one value in ASCII floating point.

`MEASure:DC?`		Returns a random value between 0 to +x in floating point ASCII. The range of x depends on the `CONFigure:DC` command
`CONFigure:DC`	`DEFault`	`MEASure:DC?` returns a number between 0 and 10
	`MIN`	`MEASure:DC?` returns a number between 0 and 1
	`MAX`	`MEASure:DC?` returns a number between 0 and 100
`CONFigure:DC?`		Returns the current configuration setting

Example:	`CONF:DC MAX`	Set the maximum range
	`CONF:DC?`	Query the current DC range. For example, if the command was issued after the command above, it would return `CONF:DC MAX<LF>`
	`MEAS:DC?`	Queries one value, for example `1.2308<LF>`

Other Commands

*IDN?	Returns National Instruments GPIB Device Simulator Rev A.x <LF>
*RST	Resets the Simulator to its default state
*TRG	Triggers the Simulator and returns one random reading (same as MEAS:DC?)
*TST?	Simulates testing the Simulator. Returns OK
*OPC	Sets the operation complete bit in the Standard Event Status Register (ESR)
*OPC?	Returns the value of the OPC bit in the ESR register
*ESR?	Returns value of Standard Event Status register as specified by FORM:SREG

Figure 2 illustrates the bits defined by the Simulator for the ESR register–bit 7 (Power On), bit 5 (Command Error), and bit 0 (Operation Complete). Bit 7 is set when the Simulator is powered on; bit 5 is set when the Simulator receives an invalid command; and, bit 0 is set when the Simulator receives the *OPC command. You can use the *ESR? command to query the value of the ESR register. The value returned is in either ASCII or HEX, as specified by the FORMat:SREGister command. The ESR register is cleared after you read it.

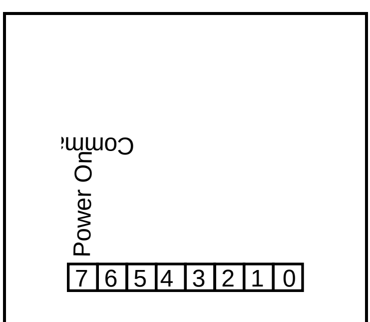


Figure 2. Three ESR Bits Set by the Simulator

`*ESE 0x##` (zero, x, mask in hex)	Sets value of Standard Event Status Enable register
`*ESE?`	Returns value of Standard Event Status Enable register as specified by `FORM:SREG`
`*STB?`	Returns value of Status Byte register as specified by `FORM:SREG`
`*SRE 0x##` (zero, x, mask in hex)	Sets value of Service Request Enable register
`*SRE?`	Returns value of Service Request Enable register as specified by `FORM:SREG`
`*WAI`	Does not do anything; included to make the Simulator IEEE 488.2 compatible
`FORMat:SREGister ASCii`	Specifies the output of ESR, ESE, STB, and SRE registers as an ASCII string (default)
`HEX`	Specifies the output of ESR, ESE, STB, and SRE registers in hex
`FORMat:SREGister?`	Returns the current format of the registers
`SYStem:HELP?`	Returns a list of all of the commands. Refer to *Command Summary* section.

Command Summary

```
SADDR
FORM:DATA ASC | INT,8 | INT,16 (?)
FORM:BORD NORM | SWAP (?)
SOUR:FUNC SIN | SQU | RAND | PCH (?)
SENS:DATA?
SENS:VOLT:RANG:OFFS?
SENS:VOLT:RANG?
SENS:SWE:TIME?
MEAS:DC?
CONF:DC MIN | MAX | DEF (?)
*IDN?
*RST
*TRG
*TST?
*OPC
*OPC?
*ESR?
*ESE 0x##
*ESE?
*STB?
*SRE 0x##
*SRE?
*WAI
FORM:SREG ASC | HEX (?)
SYS:HELP?
```

`|` – separates options for the command
`(?)` – indicates the command can be used to query the current state

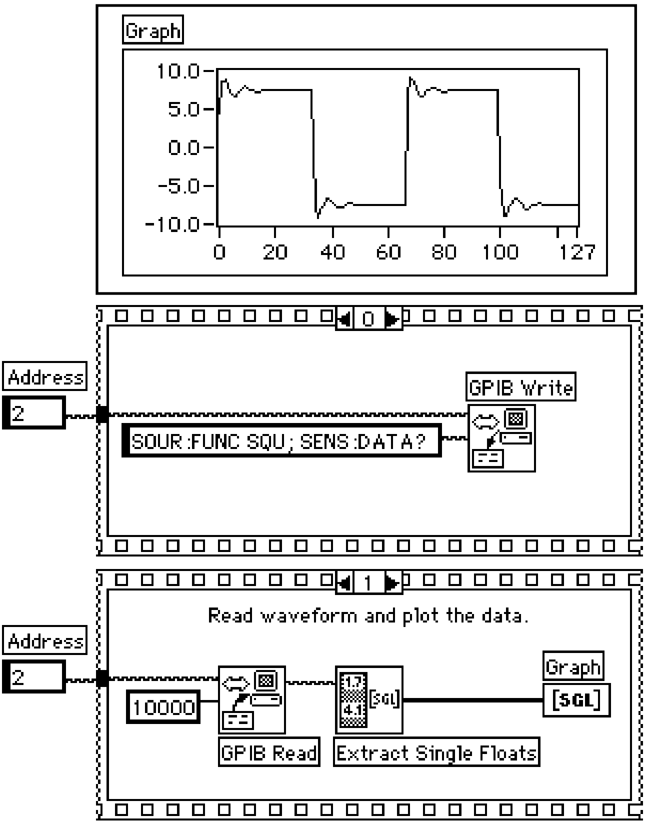
Compatibility Commands

The new Simulator EPROM supports the following commands for compatibility with the older EPROM. However, if multiple commands are sent together, they must be separated using a semicolon (;).

`E0xh0`	(E zero, x, mask in hex) Causes the box to assert SRQ whenever it has finished generating data in response to a W command. The serial poll status is specified in h0.
`E0x0`	(E zero, x, zero) Disables asserting SRQ
`G0`	Output data as 2-byte integers
`G1`	Output data as ASCII floats separated by a comma
`G2`	Output data as ASCII floats separated by a comma
`W1`	Output a noisy square wave
`W2`	Output a sine wave
`W3`	Output a noisy sine wave
`W4`	Output random data
`W5`	Output a chirp waveform
`Od0`	(Letter O) Output d0 random 2-byte integers one at a time

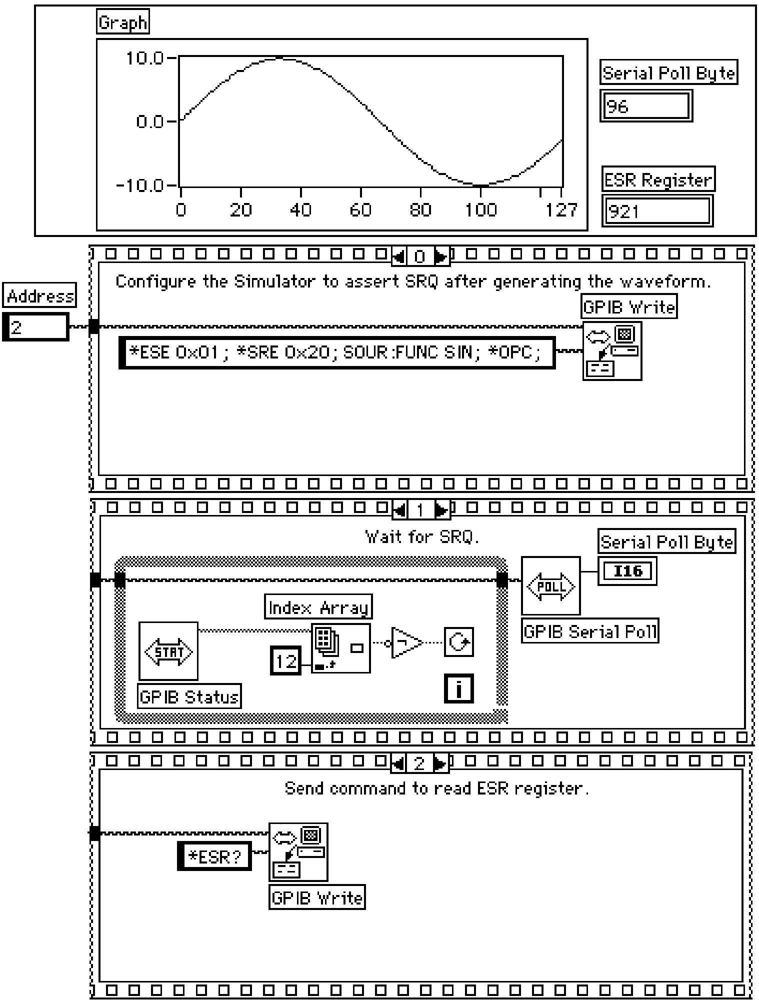
LabVIEW Examples
Example 1

The following LabVIEW example shows how to set up the Simulator to generate a square waveform, read the waveform, and plot the waveform on a graph.

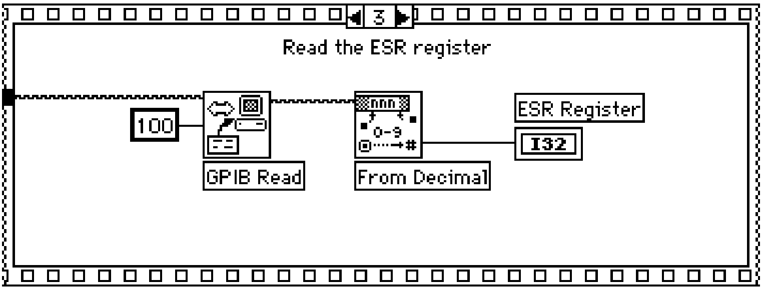


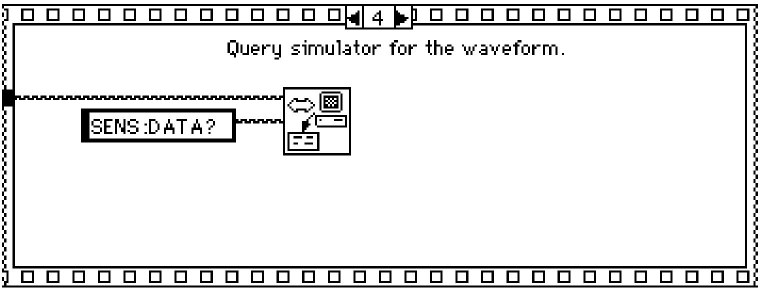
Example 2

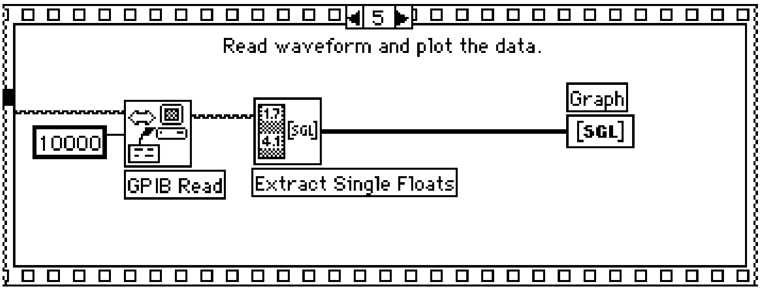
The following LabVIEW example shows how to set up the Simulator to assert an SRQ after it generates a sine waveform, read the waveform, and plot the waveform on a graph.



Example continued on the next page.

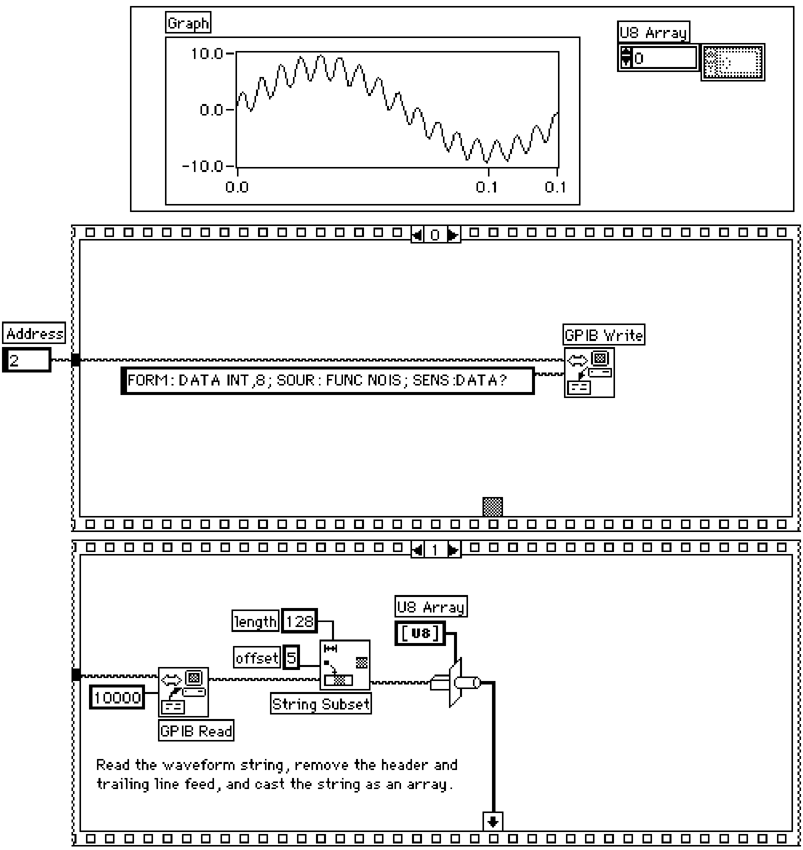

3
Read the ESR register
100
GPIB Read
From Decimal
ESR Register
I32


4
Query simulator for the waveform.
SENS:DATA?

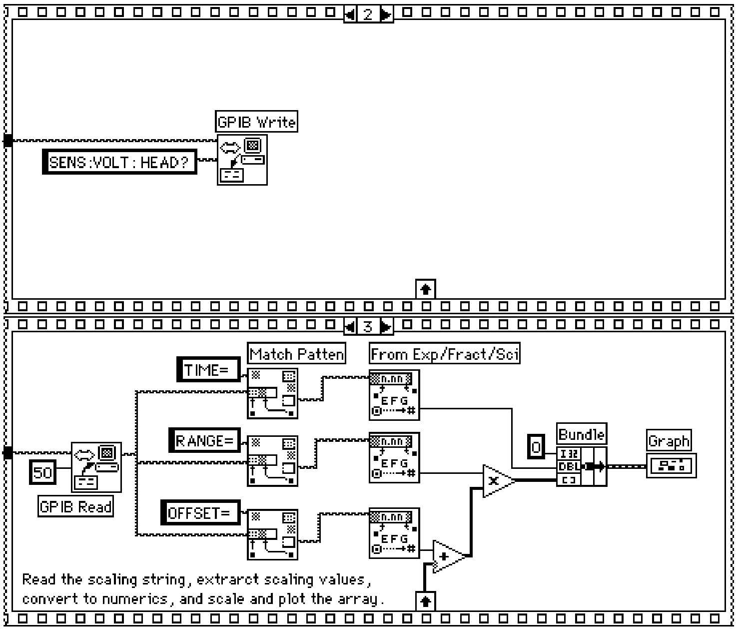

5
Read waveform and plot the data.
10000
GPIB Read
Extract Single Floats
Graph
[SGL]

Example 3

The following LabVIEW example shows how to set up the Simulator to generate a noisy sine waveform in binary format, read the waveform, scale the waveform, and plot the waveform on a graph.



Example continued on the next page.


2
GPIB Write
SENS:VOLT:HEAD?
3
Match Patten
From Exp/Fract/Sci
TIME=
RANGE=
OFFSET=
50
GPIB Read
0
Bundle
Graph
Read the scaling string, extrarct scaling values,
convert to numerics, and scale and plot the array.

LabWindows Examples

Example 1

```
REM The following example shows how to set up the
Simulator
REM to generate a square waveform, read the waveform, and
REM plot it on a graph.

DIM buffer AS STRING * 2000
DIM waveform#(128)

CALL ibfind ("DEV1", simulator%)
CALL ibpad (simulator%, 2) 'Simulator at address 2

CALL ibwrt (simulator%, "SOUR:FUNC SQU; SENS:DATA?")
CALL ibrd (simulator%, buffer$)

REM Discard header and convert ASCII data to
REM floating-point array
n% = Scan (buffer$, "%s[i6]>%128f[x]", waveform#())
uir.err% = YGraphPopup (waveform#(), 128, 4)
```

Example 2

```
REM The following example shows how to set up the
Simulator
REM to generate a noisy sine waveform, read the waveform,
REM and plot it on a graph.  The waveform is transferred
in
REM 2-byte binary.

DIM buffer AS STRING * 2000
DIM waveform%(128)

CALL ibfind ("DEV1", simulator%)
CALL ibfind ("GPIB0", board%)

CALL ibpad (simulator%, 2)    'Simulator at address 2

CALL ibwrt (simulator%, "FORM:DATA INT,16;
                              SOUR:FUNC NOIS;SENS:DATA?")

CALL ibrd (simulator%, buffer$)
```

```
REM Convert the data from binary to numeric (must byte
swap)
n% = Scan (buffer$, "%128d[zi5]>%128d[o10]", waveform%())
uir.err% = YGraphPopup (waveform%(), 128, 1)
```

Example 3

```
REM The following example shows how to set up the
Simulator
REM to assert an SRQ after it generates a sine waveform,
REM read the waveform and plot it on a graph.

DIM buffer AS STRING * 2000
DIM waveform#(128)

CALL ibfind ("DEV1", simulator%)
CALL ibfind ("GPIB0", board%)

CALL ibconfig (board%, 7, 0)  'Disable auto serial
polling
CALL ibpad (simulator%, 2)   'Simulator at address 2

CALL ibwrt (simulator%, "*ESE 0x01; *SRE 0x20;
                                  SOUR:FUNC SIN; *OPC;")

CALL ibwait (board%, &H5000)  'Wait for SRQ

CALL ibrsp (simulator%, spbyte%)   'Get serial poll byte
CALL ibwrt (simulator%, "*ESR?;")  'Read ESR
CALL ibrd (simulator%, buffer$)

CALL ibwrt (simulator%, "SENS:DATA?")  'Query the
waveform
CALL ibrd (simulator%, buffer$)

REM Discard header and convert ASCII data to
REM floating-point array
n% = Scan (buffer$, "%s[i6]>%128f[x]", waveform#())
uir.err% = YGraphPopup (waveform#(), 128, 4)
```