ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

CONTROLLING SERIAL TEMPERATURE CONTROLLERS FROM THE GPIB BUS

INTRODUCTION

This Application Note describes how to interface and control industrial temperature controllers with serial interfaces from the GPIB bus. Typical temperature controllers are the Tenny Versatenn or the Watlow 942, 988 and F4 series controllers.

BACKGROUND

Most temperature controllers used in temperature and environmental chambers have the ability to be remotely controlled over a RS-232 and RS-422 or RS-485 serial link. In many manufacturing and test applications, it is convenient to control the temperature chambers from the same GPIB bus that is being used to control other instruments and collect test data.

Two major temperature controller manufacturers are Tenny (Lunaire) and Watlow Controls. Their controllers employ unique serial command protocols such as X-on/X-off, ANSI X3.28 (ACK-NAK), and the Modbus RTU Packet Protocol. Each protocol requires special programming considerations if it is going to be handled in the test program. This application note describes the important features of each protocol and solutions to simplify the programming effort. It also covers how to connect various interfaces to the Temperature Controller, how to test the connection and how to program the application.

ICS manufactures several GPIB-to-Serial Interfaces that can be used to convert GPIB commands to serial strings and serial data back into GPIB responses. Table 1 lists ICS's serial interfaces and their protocol capabilities. For more information on a particular interface contact ICS Electronics or download the product data sheet from www.icselect.com.

TABLE 1 ICS Serial Interfaces and Protocols

Model#	Type	Comments	Protocol
4804B	Board	Transparent	X-ON/X-OFF
4894B	Box	Transparent	X-ON/X-OFF
4814*	Board	Transparent	X-ON/X-OFF
		Option Sw On	ANSI X3.28
4804B-7 w/F30145	Board	Fixed Option	ANSI X3.28
4894B-7 w/F30146	Box	Fixed Option	ANSI X3.28
4809A	Board	Fixed Option	Modbus RTU
4819A	Board	Fixed Option	Modbus RTU
4899A	Box	Fixed Option	Modbus RTU

* The 4814 is obsolete and only shown for reference purposes

THE X-ON/X-OFF PROTOCOL

The X-on/X-off protocol is for point-to-point communication and requires a carriage return character (<CR>) at the end of each command. The controller responds with a an X-off and then sends a X-on character when ready for the next command. With this protocol, the user should set the ICS GPIB-to-Serial interfaces to operate with the X-on/X-off protocol and to add a line feed (<LF>) character to the temperature controller responses before passing them back to the GPIB controller. ICS GPIB-to-Serial interfaces will automatically strip the X-on and X-off characters from the temperature controller responses so they do not go back to the GPIB controller. The added linefeed character terminates the response string and makes it easier for the GPIB controller program to input the response string.

The user should start the dialog by sending the temperature controller a single X-on character (DC1 or a 0x11 character).

The problem with the X-on/X-off protocol is that the user does not know when the temperature controller is ready for another command. To avoid overrunning the temperature controller with multiple commands, insert a nominal delay of 100-200 milliseconds between commands. After the program runs successfully, the delay time can be shortened until the controller hangs up or displays a protocol error. Then increase the delay time by 20 milliseconds for a safe value.

A typical X-ON/X-OFF dialog is shown below. *G i*ndicates GPIB Controller, *T* indicates the temperature controller. <> indicates a non-printable character.

Send Command		
G:	=<sp>SP1<sp>500<CR>	Set temp to 50
Query		
G:	?<sp>C1<CR>	Request sensor #1 reading
T:	500<CR><LF>	Value = 50.0

Figure 1 X-ON/X-OFF Protocol Example

ANSI X3.28 (ACK-NAK) PROTOCOL

The ANSI X3.28 or ACK-NAK Protocol is more robust and has the advantage of providing a response to every command. This lets the user know when the temperature controller is ready for the next command. The protocol can also be used for multiple temperature controllers on the same serial link.

The protocol rules are:

1. Initiate communication by addressing a specific temperature controller.
2. Start every message with a <STX> character and end the message with a <ETX> character.
3. Use the <EOT> character to give the temperature controller permission to respond. Acknowledge the response message with an acknowledge (<ACK>) character.
3. Restart communication if the temperature controller does not respond within a reasonable time.

Figure 2 shows an example of the ANSI X3.28 protocol for a temperature controller at address 0. *G* indicates a GPIB Controller message, *T* indicates the temperature controller response. Non-printable characters are indicated by the < > symbols. The user has to handle the non-printable characters used in the ANSI X3.28 protocol in his program.

This protocol, while not hard to generate in a PC, it is somewhat difficult to use with GPIB software and most GPIB-to-Serial Interfaces. Because the controller responses do not end with a linefeed, the user's choices are: 1) switch the terminating character to the ETX character, 2) anticipate the number of characters expected or 3) create a subroutine that reads individual characters from the GPIB interface until the serial poll status indicates the buffer is empty.

None of these choices are very good. ICS's 4894B and 4804B GPIB-to-Serial Interfaces can be set to generate an EOI when the last character is read out of the buffer. This lets the user read the response strings with a standard GPIB input, receive or enter command. A better and easier alternative is to use a GPIB-to-Serial Interface with special firmware that handles the ANSI X3.28 protocol.

Figure 2 shows the three main functions in the ANSCI protocol. They are open the serial communication link, write a command, a query and the disconnect sequence. The characters inside the <> brackets indicate a single serial character. The character values are listed in most ASCII code charts. See the Appendix 1 section in your 48xx manual for these characters.

Open Link		
G:	0<ENQ>	Open link with temp ctlr #0
S:	0<ACK>	Acknowledge link open
Send Command		
G:	<STX>=<sp>SP1<sp>100<ETX>	Set temp setpoint
S:	<ACK>	Acknowledge command
Query		
G:	<STX>?<sp>C1<ETX>	Request sensor #1 reading
S:	<ACK>	Acknowledge query
G:	<EOT>	Okay to send response
S:	<STX>500<CR><ETX>	Value = 50.0
G:	<ACK>	Acknowledge
S:	<EOT>	End of query
Disconnect		
G:	<DLE><ENQ>	End link, no response

Notes: 1. If the temperature controller returns an NAK character, the master should resend the last transmission.
2. G = GPIB Bus message, S = serial device response

Figure 2 Raw ANSI X3.28 Protocol Sequence

4814, 4804B-7 and 4894B-7 ANSI X3.28 Firmware

ICS's Model 4814, 4804B-7 and 4894B-7 GPIB to Serial Interfaces include firmware that automatically handles the ANSI X3.28 ACK-NAK protocol and simplifies the GPIB communication by eliminating the non-printable ENQ, STX and ETX characters. The 4804B-7 and the 4894B-7 are updated versions of ICS's older 4804A-7 and 4894A-7 interfaces. The Model 4814 was ICS's original GPIB to ANSI interface and was only a IEEE-488.1 device. The 4804x and 4894x interfaces are IEEE-488.2 compatible interfaces.

With these interfaces, the user's program only has to generate the temperature controller commands or query strings and wait for GPIB responses that are terminated with a linefeed. The interfaces accept GPIB messages that are terminated with a line feed and prepare responses that are terminated with a carriage return-line feed sequence (<CR><LF>). On the 4814, the ANSI firmware option is enabled by setting the OPT rocker on the Address Switch to Logic 1 and <CR><LF> by setting the CRLF rocker on SW 2 to On. In the 4804B-7 and 4894B-7, the ANSI firmware is always on.

The 4814 is an older 488.1 interface and is no longer available. The 4804B-7 with program F30145 and the 4894B-7 w/F30146 are IEEE-488.2 compatible and should be used as replacements and in new designs. Refer to the 4804B-7 Manual Addendum on the ICS Electronics website for directions on setting the 4804B-7 and the 4894B-7 serial and GPIB configurations.

The ANSI X3.28 firmware, in the 4814, 4804B-7 and 4894B-7 Interfaces, reduces the raw ANSI X3.28 strings in Figure 2 to the simpler, printable character strings shown in Figure 3. The user starts the dialog by sending the interface a GPIB Device Clear command to reset the Temperature Controller followed by the Temperature Controller address number (typically 0) to initiate communication with the controller. The user should set the GPIB read and write calls in his program to terminate input and output strings with the linefeed (<LF>) character. Use the Device Clear-Address sequence to restart the dialog if communication with the Temperature Control is ever lost. Use ICS's GPIBkybd when manually sending these commands as National Instruments' MAX and Agilent Connection Expert cannot generate Device Clear commands.

Figure 3 shows how to open the serial link to the Temperature Controller in the chamber, how to set a temperature and how to query a reading. Consult the Temperature Controller manufacturer for a complete list of Temperature Controller commands.

Open Link		
G:	Device Clear	
G:	0<LF>	Open link with temp ctlr
R:	0ACK<CR><LF>	Acknowledge link open
Send Command		
G:	=<sp>SP1<sp>100<LF>	Set temp setpoint
R:	ACK<CR><LF>	Acknowledge
Query		
G:	?<sp>C1<LF>	Request sensor #1 rdg
R:	500<CR><LF>	Value = 50.0
Disconnect		
G:	DLE<LF>	End link, no response.

Notes: 1. The disconnect command is not required if there is a separate GPIB interface for each temperature controller.
2. G = GPIB Bus message, R = GPIB response.

Figure 3 Simplified ANSI X3.28 Protocol Sequence

MODBUS PROTOCOL

The Modbus RTU Protocol is a very robust protocol that was designed for industrial control applications. The Modbus RTU Protocol consists of message packets made up of 8-bit binary bytes. Each packet has a device address byte, a command byte, register data bytes and a two byte CRC checksum. The Modbus slave device responds to each command packet with an ACK, error or response packet.

Packet Format:	Addr	Cmd	Registers.... .data	CRC

While the raw Modbus Protocol is harder to handle over the GPIB bus than the ANSI X3.28 ACK-NAK protocol it is not impossible. It requires a more sophisticated PC program because of the binary data conversions, the CRC calculations and the lack of defined terminators. The program must also be able to input and verify the response packets which the Modbus slave device sends back after every command.

GPIB to MODBUS CONTROLLER SOLUTION

ICS has three GPIB-to-Modbus Controller products that simplify the Modbus temperature control program by generating the Modbus RTU Packets and automatically handling the response packets. These products are the Model 4899A which is a stand-alone small box product for bench top use, and two board products, Models 4809A and 4819A, which are designed to be mounted inside the temperature chamber. All three interfaces are IEEE-488.2 compatible.

Modbus slave devices like Watlow's F4 Temperature Controller use addressable registers to control various functions (i.e. temperature setpoints, alarms, rates etc.) and to hold current readings. Commands are sent to the Modbus device by writing to a register. Quires are made by reading the value in a data register. ICS's GPIB-to-Modbus Controllers provide the user with simple commands to write to registers in the Modbus slave device and to read back register contents. Figure 4 shows example Write register (W) and Read register (R) commands. For more information about the 4899A, 4809A and 4819A GPIB-to-Modbus Controllers refer to the product data sheets.

Set Controller Number		
G:	C 0<LF>	Sets controller number
R:	no response	
Send Command		
G:	W 300, 50<LF>	Sets temperature setpoint
R	no response	
Query		
G:	R? 100,1<LF>	Reads from register 100
R:	21.4<LF>	Value = 21.4

Figure 4 Modbus Protocol Sequence

CONNECTING TO THE CONTROLLER

The Model 4814 GPIB-to-Serial Interface has three connectors: a GPIB connector, a RS-422 connector and a RS-232 Connector. Figure 5 shows the 4814 connector and switch locations. The Rocker Switches are shown on the next page.

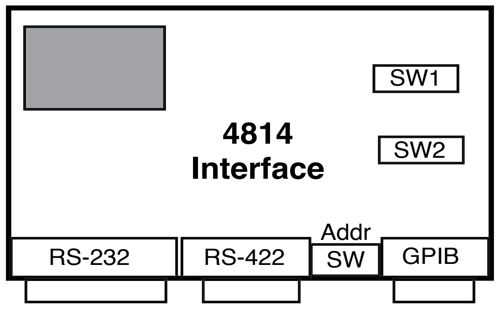


Figure 5 4814 Connector Layout

RS-232 single ended signals are good for short distances up to 50 feet or longer in electrically quiet environments. The connections are simple and require just three wires. The transmitter of one device is connected to the receiver of the other device.

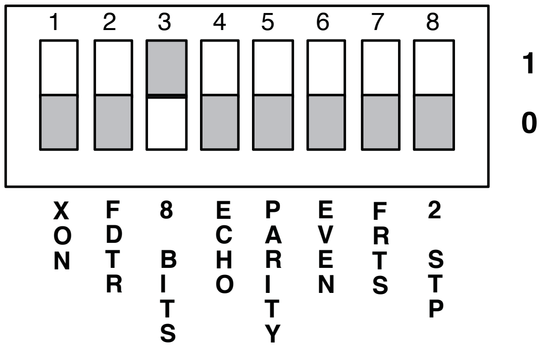
RS-422 differential signals have better common mode noise rejection and can be used over longer distances than can RS-232 signals. RS-422 connections have two pairs of wires to connect. RS-422 signal names are confusing as they vary from full names like Send-Data to abbreviations such as TX and RX. They often end in (A), (B) or + and -. (A signals are positive true and normally mate to + signals). Figure 6 shows how the 4814 is connected to a Watlow 942 Controller with RS-422 signals. The dotted ground wire is optional and is not needed for the serial signals. Use a cable with two twisted pairs of wires. Use one pair of wires for each pair of signals. For long runs, use a shielded cable and connect the shield to ground at one end of the cable.

4814 Switch Setting Reference

The 4814 has three on-card rocker switches that must be set to match the serial device which in this case is the temperature controller. Figure 5 on the prior page shows the switch locations and reference designations. Logic 1 is on, logic 0 is off. Use the silkscreen legends for rocker identification.

Switch 1 - Character Format Switch

Sets serial character data bits, stop bits and parity. Also forces DTR and RTS lines on. Normally set for 8 data bits, 1 stop bit and no parity. Some temperature controller applications set the 4814 to 7 data bits, odd parity and 1 stop bit.

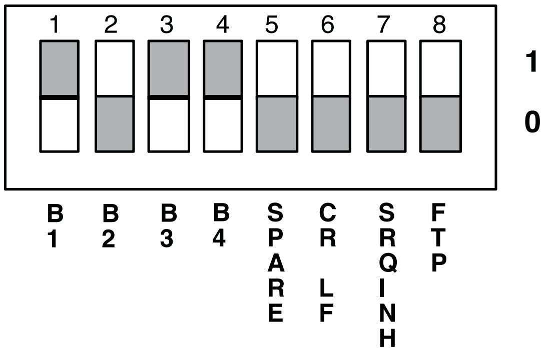


The Character Format Switch rocker functions are self explanatory. Switch is shown set for 8 data bits, 1 stop bit and no parity. Tenny factory settings are 1200 baud, 7 data bits, 1 stop bit and odd parity.

ECHO causes the received data to be echoed back to the serial device. Set Echo off for use with a Temperature Controller.

Switch 2 - Baud Rate Switch

Sets serial baud rate, 4814 response terminator and inhibits SRQ generation. The FTP rocker is for factory test and must be set off.

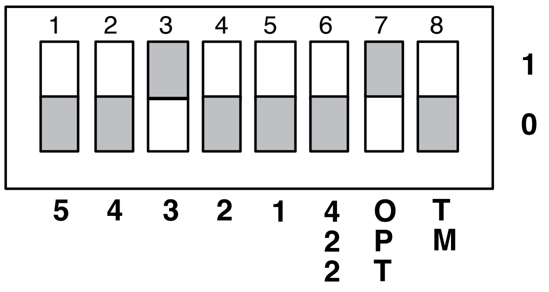


Baud Rate Table

B1	B2	B3	B4	Baud Rate
0	0	0	0	57.6K
1	0	0	0	50
0	1	0	0	75
1	1	0	0	110
0	0	1	0	134.5
1	0	1	0	150
0	1	1	0	300
1	1	1	0	600
0	0	0	1	1200
1	0	0	1	2400
0	1	0	1	4800
1	0	1	1	9600 Factory Setting
0	1	1	1	19200
1	1	1	1	38400

Switch 3 - GPIB Address Switch

Sets GPIB Bus Address, selects the active serial port and enables the ANSI Protocol Option. TM is a test mode rocker that echos serial data back to the serial device. Logic 1 is away from the edge of the PC board.



GPIB Address is set by rockers 5 to 1. From left to right, bit weights are 16, 8, 4, 2, and 1. Rocker 5 is the MSB. Switch is shown at address 4.

422 rocker must be Off to use the RS-232 port and On to use the RS-422 port.

OPT rocker is Off for normal serial data but must be On to enable the ANSI Protocol Option used with the Tenny and Watlow Temperature Controllers.

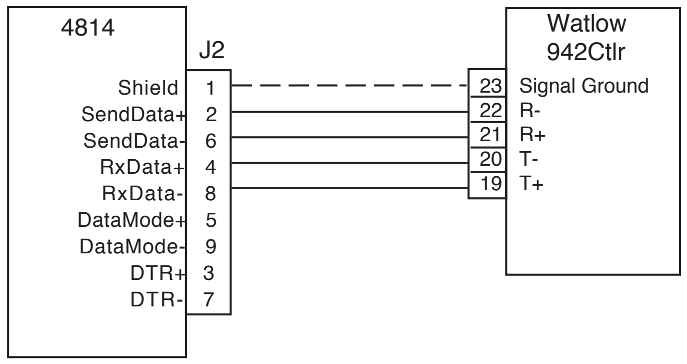


Figure 6 4814 to Watlow 942 RS-422 Connections

RS-232 connections are recommended with the Versatenn and the Watlow 942 controllers as they can be successfully used at rates of 9600 baud or faster. RS-422 connections with these controllers have not been reliable above 1200 baud. Figure 7 shows the RS-232 to Versatenn connections.

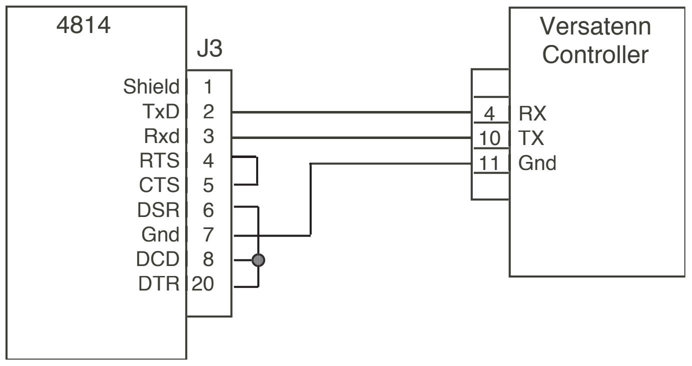


Figure 7 4814 to Versatenn RS-232 Connections

Set the 4814's Address Switch to the desired GPIB address. (The address switch is located between the GPIB and RS-422 connectors. A logic 1 setting is up (away from the PC board). Refer to page 3-4 in the 4814 manual for the rocker assignments and locations.) Set the Address Switch rockers 6 and 7 on to enable the RS-422 signals and the ANSI X3.28 ACK-NAK option firmware. Set the Character Format Switch (SW1) to 8 data bits, no parity and X-on off. Set the Baud rate switch to 9600 baud and CR-LF on. If the 4814 is to be used with the X-on/X-off protocol, set the X-on rocker on and turn the OPT rocker on the Address Switch off.

4804B-7 AND 4809A CONNECTIONS

The 4804B-7 and 4809A boards have two headers with GPIB signals. The GPIB header (J2), closest to the board edge, has GPIB signals and Address Switch inputs, and mates with the flat ribbon cable from an ICS GPIB Connector/Address Switch Assembly. This assembly mounts a GPIB Connector and a rocker switch on the rear panel of the chassis. The internal GPIB header (J1) has only GPIB signals and mates to a rear panel GPIB connector with a flat ribbon cable. On the serial side, header (J3) and the 25-pin connector (J4) have the same RS-232 or RS-422/RS-485 signals and either one can be used for the serial connection to the Temperature Controller. Refer to the 4804B or 4809A Instruction Manual for the signal pinouts.

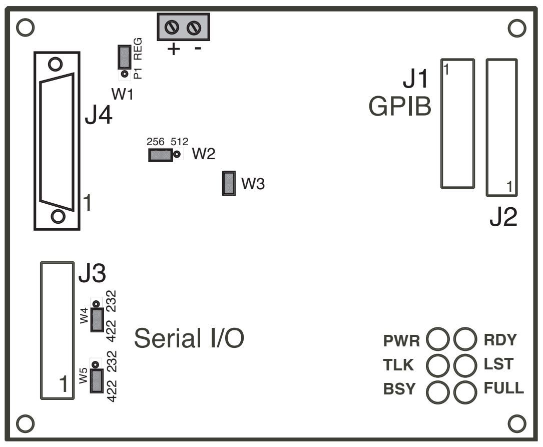


Figure 8 4804B Board Layout

For RS-232 signals, leave the two jumpers by J3 in the 232 positions as shown in Figure 8. For RS-422 signals, move the two jumpers to the inboard (422) positions. For RS-485 operation, select RS-422 signals and use a SCPI command to enable RS-485 operation. Apply DC power to the 4804B at the terminals on the top of the board. Set jumper W1 to P1 for 5 Vdc power or to REG for unregulated 5.5 to 15 Vdc power.

4894B AND 4899A CONNECTIONS

The 4894B and 4899A and physically similar boxes. They have a 25-pin connector on the rear panel with the serial interface. Three jumpers inside the box next to the serial connector select RS-232 or RS-422 signals. For RS-485 operation, select RS-422 signals and use a SCPI command to enable RS-485 operation. Use the wiring diagrams shown in the 4899A/4809A/4819A Instruction Manual to connect the GPIB Interface to the temperature Controller.

DEBUGGING THE CONNECTION

The following debugging procedure includes general comments that apply to all controller connections and then some protocol specific steps.

GENERAL COMMENTS

To debug the connections, first verify that the Temperature Controller, the GPIB Interface and the GPIB bus are correctly connected. For 4804B and 4809A boards, verify that the GPIB Connector/Address Switch cable is **not** plugged into the serial interface header. If RS-232 signals are being used, recheck the RS-232 signal connections. The TxD signal from the Interface Card should be connected to the Temperature Controller's RxD input and the Temperature Controller's TxD output should be connected back to the Interface Card's RxD input. Use a voltmeter or a LED indicator to verify the signal connections and that the two Tx lines are **not** connected together. To test with a voltmeter, open the Tx/Rx line connections and measure the signals. The Tx lines should be -6 to -12 Vdc.

Secondly check to see if the Temperature Controller settings match those of GPIB Interface. Depending upon the Temperature Controller model you can verify the signal type selection setting, controller address, baud rate, number of data bits, parity and protocol from its front panel or internal switches. Make any necessary changes to the controller and GPIB Interface settings. If any GPIB Interface settings were changed with a GPIB command, they should be saved with the *SAV 0 command before proceeding.

Connect the GPIB controller directly to the Temperature Chamber. It is best to keep things simple and leave the other devices off of the GPIB bus while checking out the Temperature controller. Then use a live keyboard program such as ICS' GPIBkybd program to send commands over the GPIB bus and to read back the responses from the temperature controller. ICS's GPIBkybd program runs GPIB controller cards from ICS, Measurement Computing and National Instruments. A copy of the GPIBkybd program can be downloaded from ICS's website at www.icselect.com. Note that National Instruments' Measurement and Automation Explorer and Agilent's Connection Expert cannot send the GPIB Device Clear command and cannot initialize the X3.28 protocol connection.

When the GPIBkybd program is started, it finds the GPIB addresses of all of the devices connected to the GPIB bus. The GPIBkybd program sets the GPIB Controller to address 0 or 21. If multiple devices are found, enter the address of the Temperature Controller interface into the Device Address box and press SET. Never set a bus device to the same address as the GPIB Controller. To send a text command, enter the command string in the Device Command text box and press SEND. To read a response press the READ DEVICE RESPONSE button. To send a 488.1 command like Device Clear, just press the DEVICE CLEAR button. Do **not** use the 488.2 commands All Poll or Find SRQ during the debugging process.

For all GPIB Interfaces except the 4814, test out the GPIB-to-Interface Card connection by sending the GPIB Interface an "*IDN?" query. The GPIB Interface's response message will appear in the Device Response window. A typical response is "ICS Electronics, 4894B, S/N 123456, Rev 06-01-02". If the message does not appear check for the wrong device address. Press the Send IFC button to clear the interface and retry the "*IDN?" query. For the 4814, all you can do is Serial Poll it by pressing the SERIAL POLL button. You should get a numeric response.

The GPIBkybd program cannot send non-printable characters and so cannot be used to checkout the X-ON/X-OFF protocol.

ANSI X3.28 PROTOCOL DEBUGGING

To checkout the ANSI X3.28 protocol follow the example sequence shown in Figure 3. Be sure the 4814 OPT switch is set On. Start the dialog by sending the interface a Device Clear command followed by the temperature controller's address number, typically a 0. Press the READ DEVICE RESPONSE button to read the controller's response. An "ACK" response indicates the Temperature Controller recognized the command and is ready to receive commands. A "NAK" response indicates a communications failure from the GPIB Interface to the Temperature Controller. Retry the above sequence. If it fails a second time, go back and recheck the serial connections and settings. Be sure you are using the correct connectors and that the Temperature Controller jumpers and switch (page 6) are in the correct position.

Next read a response from the Temperature Controller by sending it a query command. Use "? <space> mdl " for the model number or "? <space>C1" to read the current temperature. The response will automatically appear in the Response text box. Next refer to the Temperature Controller's manual for other commands to use with your controller. Press the READ DEVICE RESPONSE button after every command to read back the "ACK" response. Query responses automatically appear in the text window since they include a question mark.

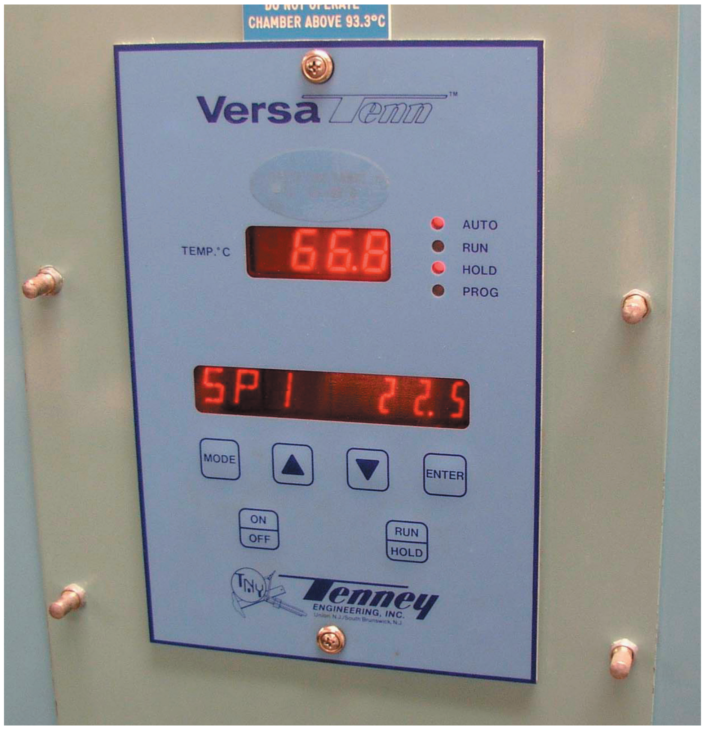


Figure 9 Versatenn Front Panel

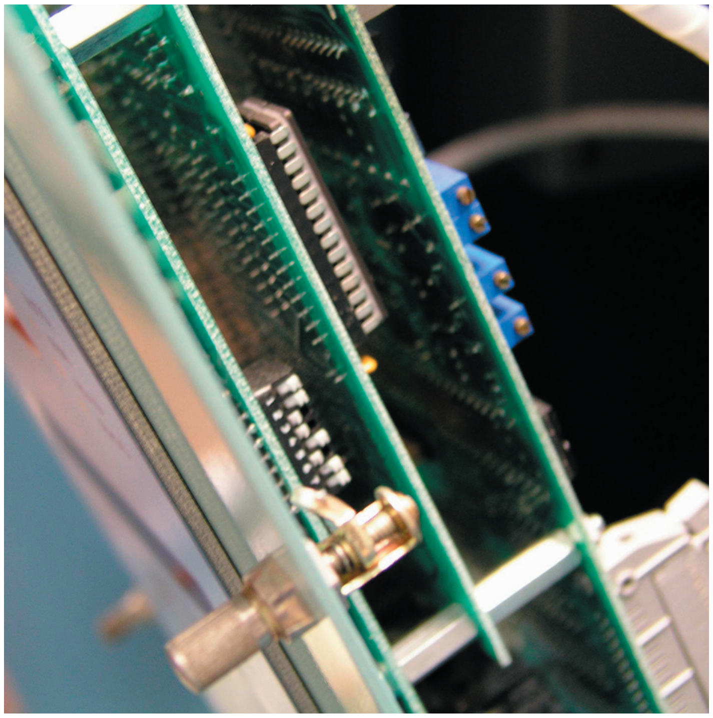


Figure 10 Internal Dip Switch

If you cannot initialize serial communication with the Tenny Versatenn Controller after checking the serial settings, check the Versatenn's internal switch setting. To check the Versatenn's internal switch, loosen the front panel hold down screws shown in Figure 9 below or otherwise open the front panel of the Temperature Chamber. Pull down the Versatenn's front panel to expose the internal dip switch as shown in Figure 10.

Rocker number 2 is the rocker that enables remote communication. Set the rocker in the down position to enable remote communication. Figure 10 shows the rocker up which is the wrong position for remote communications.

MODBUS PROTOCOL DEBUGGING

Checkout the Modbus protocol by following the example in Figure 4 to checkout the Temperature Controller connections. Use the "C" command to set the Temperature Controller number in the GPIB Interface. Use the "W" command to write a value to a register. Use the "R?" command to read a register value. In the case of the Watlow F4 Controller, register 0 is the model number and register 100 is the current temperature reading. Note that the Modbus protocol does not provide a response to "C" and "W" commands.

If the ERR LED on the GPIB Interface illuminates, it means an error bit is set in the Interface's ESR Register. (Refer to Figure 3-1 in the 4899A/4809A/4819A Instruction Manual) There may be a problem with the temperature Controller or the Interface may have been sent a bad command. Use the "*ESR?" query to read the GPIB Interface's ESR register and to clear the ERR LED. The response is a decimal value that is the sum of the 'on' bits in the ESR Register. At power turn on, the first ESR reading will show 128. Subsequent readings will be 0. If the response contains a value of 64, bit 6 was set and there is a Modbus error. If the response contains a 32 or a 16, this indicates a command syntax error. Retry the command.

Modbus errors are contained in a separate Modbus Error Register. Use the "E?" query to read the Modbus Error Register.

Error	Description
0	No errors present
1	Exception Code 1
2	Exception Code 2
3	Exception Code 3
100	CRC Error
101	Timeout Error indicates no characters received in response message.
2nn	Partial or corrupted message received. nn is the number of received bytes.

Figure 11 Modbus Errors

PROGRAMMING AIDS

The following sections describe some basic concepts for including control of the Temperature Chamber in a test program.

QUICK BASIC ACK-NAK PROGRAM

The example program in Figure 12 shows how to use a 4814, 4894B-7 or a 4804B-7 to communicate with a Temperature Controller that uses the ANSI X3.28 ACK-NAK protocol. A typical application has the Temperature Chamber on the GPIB bus along with other measuring instruments. All communication to the Temperature Controller is done by addressing the 4814, 4804B-7 or 4894B-7 at its GPIB address. In this example, the 4814 has the ACK-NAK option enabled.

The user initiates the communication with the Temperature Controller in the initialization section of the program by sending the Temperature Controller its address number. The temperature setting, temperature reading and other chamber control instructions and queries are put in the main section of the program. The example Quick Basic program uses ICS's GPIB command set and a 488-PC2 card. The ICS GPIB commands can be easily exchanged for those for another GPIB controller card and/or the program can be converted to another language such as Visual Basic, HP Basic or C/C++.

```
INITIALIZATION
TempCtlr% = 4                             '4814 GPIB address
CmdStr$ = "1"                             'temp ctlr address
CALL ieOutput(TempCtlr%, CmdStr$)         'sends temp ctlr ad
                                                     dress
msDelay 100                               'wait 100 ms
Resp$ = String$(80," ")                   'clear input string
CALL ieEnter(TempCtlr%, CmdStr$)          'read ACK

MAIN SECTION
CmdStr$ = "= SP1 500"                     'set temperature
CALL ieOutput(TempCtlr%, CmdStr$)         'sends string
Resp$ = String$(80," ")                   'clear input string
CALL ieEnter(TempCtlr%, CmdStr$)          'read ACK
msDelay 100                               '100 ms delay
CmdStr$ = "? c1"                          'temp query
CALL ieOutput(TempCtlr%, CmdStr$)         'sends string
Resp$ = String$(80," ")                   'clear input string
CALL ieEnter(TempCtlr%, CmdStr$)          'read temperature
```

Figure 12 Quick Basic ANSI X3.28 Program

The program uses three variables. TempCtlr% is the 4814's GPIB address. You need a separate address variable for each GPIB device. CmdStr$ is the second variable. It is a string that holds the current instruction that the program sends through the 4814 to the temperature controller. Resp$ is the third variable and it is used to read responses from the temperature controller. You can always create multiple response or command string variables if they make sense in your program. Be careful when making up the command strings. Note that the ANSI X3.28 protocol requires spaces after the "?" and before a parameter.

If you need to send the temperature controller multiple messages or query multiple parameters such as temperature, setpoint, error status, etc., it is necessary to put a delay of 100 ms between each query sequence. These delays can be reduced once the program has been debugged to speed up the program.

If you are using a 4894B or 4804B GPIB-to-Serial Interface without the ANSI firmware to communicate with a ANSI X3.28 device, set the GPIB Interfaces to generate an EOI when talking out the last character from the Rx buffer. Use the example sequences shown in Figure 2 to communicate with the temperature controller. Include the non-printable characters shown in the <> brackets in your command strings. Their character values are shown in the ASCII chart in the appendix of the Interface's instruction manual. For Basic programs use the CHR$() function to convert the character's decimal value into a string characters. For C, use '\n' type formats to convert the character's octal value into a string characters (STX = '\2'). Provide a delay after each command so that the GPIB Interface has time to receive the temperature controller's complete serial response before reading the response string. Otherwise the Interface will generate an EOI while outputting only a partial string. An alternative to programming for the raw ANSI X3.28 protocol is to upgrade your interface to a 4804B-7 or 4894B-7. Contact ICS for possible replacement EPROMs or replacement interfaces.

VISUAL BASIC MODBUS PROGRAM

Application Note AB48-25 describes how to connect the 4899A, 4809A and 4819A GPIB to Modbus Interfaces to the Modbus Temperature Controller and lists a Visual Basic program that can be used as the starting point for your Visual Basic program. The program source code is available on ICS's website. If you want to use transparent GPIB-to-Serial Interfaces and handle the Modbus protocol in the computer, Application Note AB48-24 lists a Visual Basic program that works with the ICS 4894 and 4804 GPIB-to-Serial Interfaces.

LABVIEW ANSI X3.28 DRIVER

ICS has developed a LabView driver library for controlling Watlow 942 and Tenny Versatenn Temperature Controllers that use the ANSI X3.28 protocol. The driver works with the 4804B-7 and the 4894B-7 Interfaces and with some modifications, the older 4814. Three demo VIs are available on the ICS website (www.icselect.com) that demonstrate setting a Static Setpoint, Loading temperature steps and Running programmed temperature steps. They were developed in LabView 5.1 with the 4804B-7. If you have a different version of LabView, download the version 5.1 Runtime Engine from the LabView Driver page to try out the demo VIs. When the demo VIs start, they query the interface for its IDN message.

The driver library contains all of the subVIs and editable versions of the demo VIs to show you how the subVIs interconnect. See the data sheet on the website for more information about the driver contents. Order the ANSI X3.28 LabView library as part number 123167. Note: If you are using the 4814, the IDN subVI should be edited out of your program as the 4814 is not a 488.2 compatible device.

LABVIEW MODBUS DRIVER

ICS has developed a LabView driver library for controlling Watlow F4 series Temperature Controllers and other Modbus devices that use the Modbus RTU Protocol. The driver works with the 4899A, 4809A and 4819 GPIB Interfaces. Two demo VIs are available on the ICS website that demonstrate setting a Static Setpoint and Running with a Strip Chart Recorder. They were developed in LabView 5.1 with the 4899A. If you have a different version of LabView, download the version 5.1 Runtime Engine from the LabView Driver page to try out the demo VIs.

The driver library contains all of the subVIs and editable versions of the demo VIs to show you how the subVIs interconnect.

SUMMARY

This application note has described three serial protocols for communicating with Tenny and Watlow Temperature Controllers and has provided serial connections, debugging steps and some programming information. The goal of this application note is to make it easier to control these devices from a GPIB test program.

The X-ON /X-OFF Protocol is straight forward and can be implemented with virtually any ICS GPIB-to-Serial Interface. Its drawback is that it is open ended and does not provide a response to every command.

The ANSI X3.28 ACK-NAK Protocol has the advantage of providing a serial response after every command to verify that the command was received. ICS's Models 4804B-7, 4894B-7 and 4814 GPIB-to-Serial Interfaces include firmware that simplifies the ANSI X3.28 protocol and makes it easier for the user to control the temperature chamber from his test program.

The Modbus RTU Protocol is a more robust packet oriented protocol that uses binary characters and ends each packet with a CRC sequence. The Modbus RTU Protocol is the most difficult protocol to use in its raw format as it requires a significantly more complex program to generate the packets and to handle the response packets. ICS Models 4899A, 4809 and 4819 GPIB-to-Modbus Controllers automatically handle the packet protocol and let the user control Modbus devices with simple commands over the GPIB bus.