ICS ELECTRONICS
division of Systems West Inc.

IEEE 488 APPLICATION BULLETIN

CONTROLLING MODBUS DEVICES FROM THE GPIB BUS

INTRODUCTION

ICS's Model 4899, 4809 and 4819 GPIB-to-Modbus Controllers simplify controlling Modbus devices from the GPIB bus by converting simple GPIB commands into Modbus RTU packet messages. This application note describes how these GPIB Controllers generate Modbus messages and how to use them to control Modbus devices. Modbus control is demonstrated with an example GPIB-to-Modbus Temperature Control program that uses a Watlow F4 to control temperatures. The example program is a Visual Basic program that runs on a PC with Windows 95, 98, 2K or XP. ICS's website also has LabVIEW VIs that can do setpoints and ramp profiles.

MODBUS PROTOCOL

The Modicon Corporation, now AEG Schneider, created a serial protocol known as Modbus for controlling industrial process control systems. The Modbus protocol is an extremely reliable protocol for exchanging data in noisy industrial applications. The Modbus protocol is a packet exchange protocol where the controller receives a response to each packet that it sends to a device. Each packet contains the address of the controller that is to receive the information, a command field that says what to do with the data fields and a CRC (Cyclic Redundancy Checksum) field that is used to validate the packet. The Modbus protocol uses two coding schemes, ASCII or HEX characters. The HEX character format is known as the RTU protocol and it is the protocol supported by the 4809, 4819 and 4899 Controllers.

Modbus devices pass data and commands through addressable registers. Each register has a different function as defined by the device designer, i.e. temperature setpoint, reading, alarm setting etc. The registers may be read only, write only or read-write. Figure 1 shows an Modbus HEX RTU message packet. The Address and Command fields are

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

Figure 1 Modbus Packet

8-bits. The Register-Data fields hold multiple 16-bit words and contain the register address and data values. The data field is a signed 16-bit word. The CRC field is a 16-bit binary word. The RTU message packet is transmitted as asynchronous 8-bit serial characters with single start and stop bits. Each character holds 8 bits or 2 HEX characters.

CONTROLLER OPERATION

The 4809, 4819 and 4899 are IEEE 488.2 compatible GPIB devices that can control single or multiple Modbus slave devices. They accept simple GPIB bus commands that are used to create Modbus RTU packets that are transmitted serially to the Modbus slave device(s). The data transmission can be to a single Modbus device over a RS-232 link or to multiple Modbus devices over an RS-485 network. The Controller's SCPI parser accepts IEEE-488.2 common commands, SCPI commands and Modbus commands. The IEEE-488.2 and SCPI commands are used to setup the Controller's Status Reporting Structure or to configure its GPIB and Serial interfaces. Any commands that end in a '?' are a query and the Controller responds by outputting the requested data on the GPIB bus the next time it is addressed as a talker.

The 4809, 4819 and 4899 include a list of commands for controlling Modbus devices. These commands specify the Modbus slave device address, a read or write operation, a register number and data or the number of registers to be read. When the Controller receives a read (R) or write (W) command, it converts the GPIB characters into HEX bytes, assembles the packet, and transmits it to the Modbus device.

If the packet is successfully received by the Modbus device, the Modbus device will generate a response packet that either acknowledges receipt of the message or returns the requested data. The Controller receives the response packet and checks it for a valid checksum and byte count. If the packet is a valid response to a read command, the returned data is held in the GPIB transmit buffer and outputted onto the GPIB bus the next time the Controller is addressed to talk. If the packet is a command acknowledgment message, there is no further action. If the packet is invalid or contains a Modbus Exception Error code, then the code is placed in the Modbus Err Register and the ERR LED lights.

The GPIB Controller expects to receive a response packet from the Modbus device within a preset time period or it declares a timeout error. The timeout period is programmable and is factory set to 100 milliseconds. (Customer tests have shown that 300 milliseconds is a better value for the Watlow F4 Controller as it occasionally gets busy.) If the received packet was not a valid packet, or was an exception message, then the Controller sets the appropriate bit(s) in the Questionable Register and puts a decimal error value in the Modbus Error register. Both registers are part of the Controller's IEEE-488.2 Status Reporting Structure. The GPIB Controller can be programmed to generate an SRQ if an error occurs by setting the appropriate enable bits in the Status Reporting Structure. If an enabled error bit becomes set, the register's summary signal cascades into the Status Byte Register and generates a Service Request by asserting the SRQ line. The SRQ line stays asserted until the unit is serial polled or until the bits that caused the SRQ are reset. (Refer to AB48-11 or to the 4899's Instruction Manual for a description of the IEEE-488.2 Status Reporting Structure and how to use it to generate SRQs.)

MODEL 4809 and 4899 DIFFERENCES

The 4809 and 4899 are functionally identical but physically different units. The 4809 is a PC board assembly that can be mounted inside a host chassis. The 4899 is packaged in ICS's small Minibox case and can be rack mounted or run on a benchtop.

MODEL 4819 DIFFERENCES

The Model 4819 is a PC board assembly that is intended to be mounted perpendicularly on the rear panel of the host chassis so that its GPIB connector can protrude though the rear panel. It also has a RS-232 interface connector that brings the serial Modbus signals out to the rear panel. The 4819's internal serial interface only has RS-232 signals so it can only drive one Modbus Controller. Its firmware functions are the same as those in the 4899 and 4809 Controllers.

MODBUS CONNECTIONS

The 4809/4819/4899's RS-232 interface connects to a single slave device over distances of less than 100 feet. The 4809 and 4899's RS-422/RS-485 interface can drive multiple slave devices at distances up to 1200 feet. Terminating networks with pullup and pulldown resistors are required on RS-422 or RS-485 networks that use tristated transmitters. The 4809/4819/4899 Instruction Manual shows how to connect the GPIB-to-Modbus Controllers to various Modbus devices.

OVERALL SYSTEM

Figure 2 shows a typical system configuration. The PC is the system controller and with a GPIB controller card installed in it. A standard GPIB cable connects the PC to the 4899 GPIB-to-Modbus Interface. (It could also be a Model 4809 or a 4819 GPIB-to-Modbus Interface.) The 4899's RS-232 serial interface is connected to a Watlow F4 Controller as shown in Figure 9.

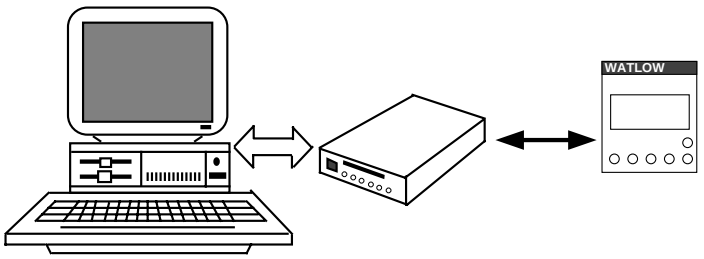

Figure 2 System Configuration

MODBUS COMMAND SET

Although the 4809, 4819 and 4899 are IEEE-488.2 compatible devices and incorporate a SCPI parser for their own functions, they have a very simple set of commands for controlling Modbus devices. (See Table 1 on page 3.) The 'C' command sets the Modbus device address for addressing multiple devices on the serial network. The 'R' command reads a Modbus register. The 'W' command writes a value to a Modbus register. The register numbers and functions come from the Modbus device's manual. Some examples for the Watlow F4 Controller are:

C 1	'sets 4899 to address Modbus device #1
R 0,1	'reads model # at register 0
R 100,1	'reads input value #1
W 300,125	'sets setpoint #1 to 125 ℃

TABLE 1 MODBUS COMMANDS

Syntax	Default	Meaning
C addr	1	Modbus Address Command. Sets Modbus slave device address for subsequent commands. Value for *addr* is 1 to 255.
L[?] w	-	Loopback Command. Writes a 16-bit word, *w*, out to a Modbus device and returns a single response word to the GPIB bus. The question mark is optional. Value for *w* is 0 to 65535.
R[?] reg, num	-	Read Register Command. Reads one or multiple Modbus device registers. User specifies starting register *reg* and number of registers to be read *num*. The [?] is an optional symbol so programs like ICS's GPIBKybd program can recognize the comand as a query and automatically read the response. Values for *reg* are 0 to 32767. Values for *num* are 1 to 64. Responses are returned as 16-bit decimal or HEX values separated by commas. Output format selected with the Format command. i.e. R? 0,1 reads Watlow Model Number. Response is 5270 for Watlow Model F4 R? 0,3 reads three successive registers. Response is 5270,0,123 for the Watlow F4 Controller.
W reg, w	-	Write Register Command. Writes a 16-bit value, *w* to a single Modbus device register, *reg*. Values for reg are 0 to 32767. Values for *w* are 0 to 65535. An example is: W 100, 55 writes the decimal value 55 to register 100.
WB reg, num, w(0).w(n)		Write Block Command. Writes multiple 16-bit words, *w(i)* to multiple registers. Starting register, *reg*. Number, *num* specifies how many words are to be written. Values for *reg* are 0 to 32767. Values for *num* are 1 to 64. Values for *w* are 0 to 65535.
D time	100	Timeout Command. Sets timeout value of Modbus response message in milliseconds. Timeout is the total time for the message to be received by the 4899 or 4809. Value for *time* is 1 to 65,535 milliseconds. Default is 100.
D?		Queries the current timeout setting.
E?	-	Read Error Command. Reads and clears the Modbus Error Register and bit 6 in the Event Status Register. Returns a error code whose value is 0 to 255. Error values are: 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 2nn Partial or corrupted message nn is number of received bytes.

VISUAL BASIC EXAMPLE SOFTWARE

ICS has developed an example Visual Basic program, called TempCtlr, that shows how a 4809, 4819 or a 4899 GPIB-to-Modbus Interface can be used to control a Watlow F4 Temperature Controller. The example routines can be easily incorporated in a final application program. The program is written in Microsoft Visual Basic version 6 and can be run by anyone with Visual Basic version 6 or the VB6 Runtime File. Although the program description describes how the program works with a Model 4899, it works the same with the 4809 or 4819 Interface cards. The program uses the National Instruments command set and requires an ICS, Measurement Computing or National Instruments GPIB Controller card installed in the PC.

The program has three forms: a Main Form, a GPIB Commands and Setup Form and a Profile Generator Form. The Main Form contains controls that send commands to the GPIB interface to control the Modbus device, display its responses, set static temperatures, set humidity and run temperature profiles. Figure 3 shows the Main Form when the program is first started. The GPIB Commands Form contains controls that query and set the 4809/4819/4899's GPIB address, its 488.2 Status Reporting Structure and its serial interface. The Profile Generator Form has controls that create temperature controlling profiles with ramps and soak times.

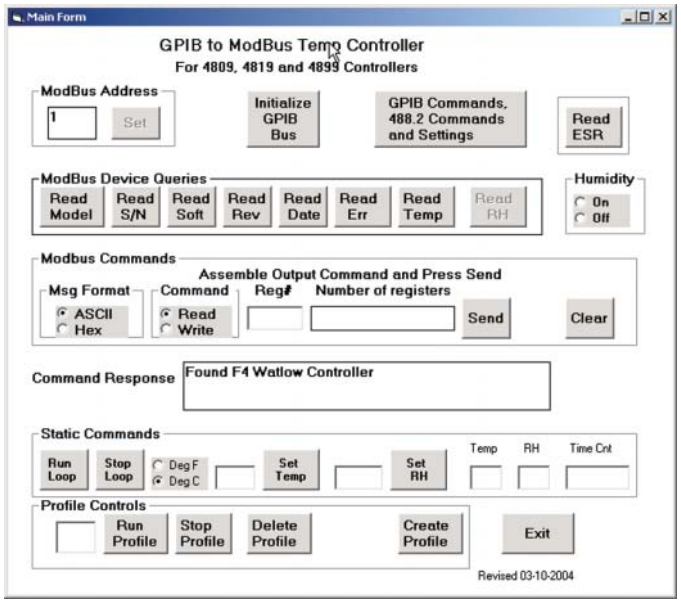

Figure 3 TempCtlr Main Form

Table 1 Notes:
1. All values are in decimal. To enter HEX values, use #h value.
2. Response parameter format set by SCPI FORMat command.
3. Negative values start with a minus '-' sign.
4. Separate multiple values by a comma.

RUNNING THE PROGRAM

To run the TempCtlr program, click on TempCtlr.exe. When the Main Form appears, click the Initialize Button which initializes the GPIB Controller Card in the PC, checks for a 4809/4819/4899 type Interface, checks for presence of the F4 Controller and then enables the other controls. An error message is displayed if the F4 Controller is not found.

If this is your first time in the program, or if the 4899 is set to an address other than 4, click on the GPIB Commands and Setting Button to setup the 4899.

GPIB COMMANDS AND SETUP FORM

The GPIB Commands Form contains controls that sets the GPIB address the program uses for the 4899, that modifies the 4899's serial interface settings and configures the 4899's 488.2 Status Reporting Structure. Click the IDN? button to confirm that you have GPIB communication with the 4899. If the 4899 is not responding to the program, check its GPIB address and change the address in the GPIB Address box to match the 4899's current setting and click Set. (This changes the address used by the program, not the 4899's actual GPIB address.)

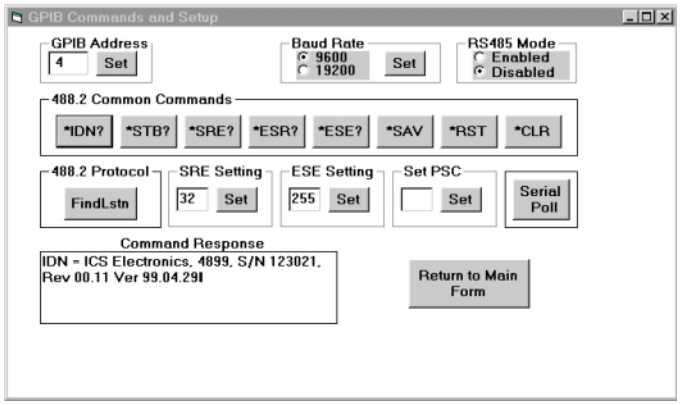

Figure 4 GPIB Commands and Setup Form with IDN Message Response

If the serial baud rate needs to be changed, click the correct baud rate button and then the Set button. Leave RS485 disabled when using RS-232 signals. Enable RS485 when controlling multiple F4s on an RS-485 network. RS-485 changes will not take affect until the interface sends the next serial message.

Click ESE? to confirm that bit 6 in the ESE register is enabled. The response is a decimal value equal to the sum of the binary weights of the enabled bits. Bit 6 has a decimal value of 64 and needs to be on to activate automatic Modbus Error warning on the Main Form. If not, use the ESE Setting box to enable bit 6. Refer to Section 3 of the 4899's Instruction Manual for more information about the other 488.2 controls shown on the GPIB Commands and Setup Form. When done, set PSC to 0 to save the SRE and ESE settings. Use the *SAV control if you want to save any other parameters. Click the Return to the Main Form button to exit the form.

MAIN FORM

On the Main Form, verify that the Current Address box value matches the address of your Modbus device. (Watlow F4s default to address 1). If not, change the address and click the Set Modbus Address control to enable the remaining Modbus controls.

The Modbus Device Queries box has controls that generate preprogrammed queries to the Modbus device. Click on Read Model (Number) to verify communication with your Modbus device. Read Model reads register #0 and displays the result in the Command Response text box. Figure 5 shows the response (5270) from a Watlow Model F4 controller. The other controls, Read S/N, Read Soft(ware), Read Rev(ision) and Read Date query the remaining F4 identification registers (registers 1 thru 5). Change the Msg Format setting from ASCII to HEX to see the difference in the query response values. This control also affects the output data so leave it in ASCII when done unless you want to send and receive HEX values.

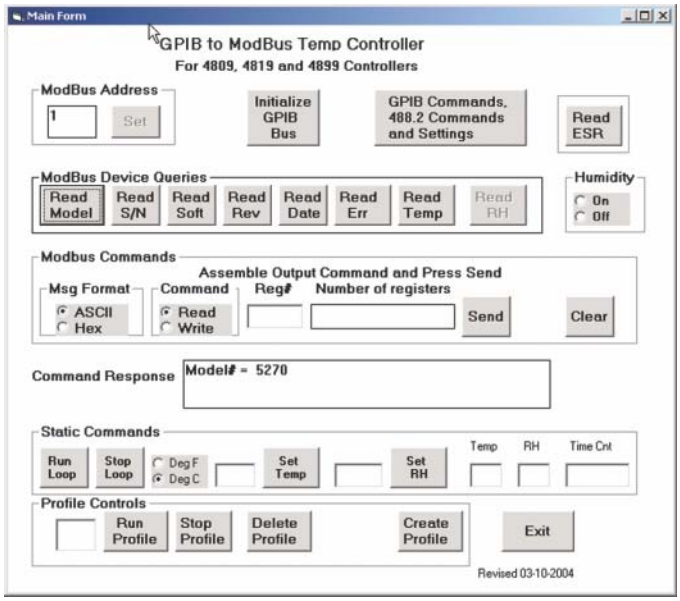

Figure 5 Main Form with F4 Model Number

The remaining Modbus Device Query buttons let you read the 4899's Modbus Error Register, read Temperature (register

100) and Humidity (register 104). You can easily modify these routines to read other Modbus registers.

The Modbus Commands box lets you manually generate Modbus read or write commands for any Modbus device. To read a register, click the Read Option Button. This sets up the Reg# and Number-of-Register boxes. Enter the number of the register to be read in the Reg# Box. Enter the number of sequential registers to be read in the Number-of-Registers Box. If you make a mistake, use the Clear Entry or the cursor to clear out the boxes. Click the Send Command button to send the read message to the Modbus device. The response will appear in the Command Response box.

To write to a register, click the Write Option Button. Enter the number of the register to be written to in the Reg# Box. Enter the data in the Output Data Box. When writing to multiple sequential registers, separate the values with a comma. Click Send Command to send the write message to the Modbus device.

If a Read ERR Reg flag appears, the 4899 detected a Modbus error. Click the Read Err button to read the 4899's Modbus Error Register. If the value is 0, the 4899 may have detected a command or user error. Use the Read ESR control to read the 4899's ESR register. Refer to the ESR register bit definitions in Figure 3-1 of the 4809/4819/4899 Instruction Manual to determine the cause of the error. Figure 6 shows the Modbus Error Register message after writing an out-of-range value to register 4003 in the Watlow F4 Controller.

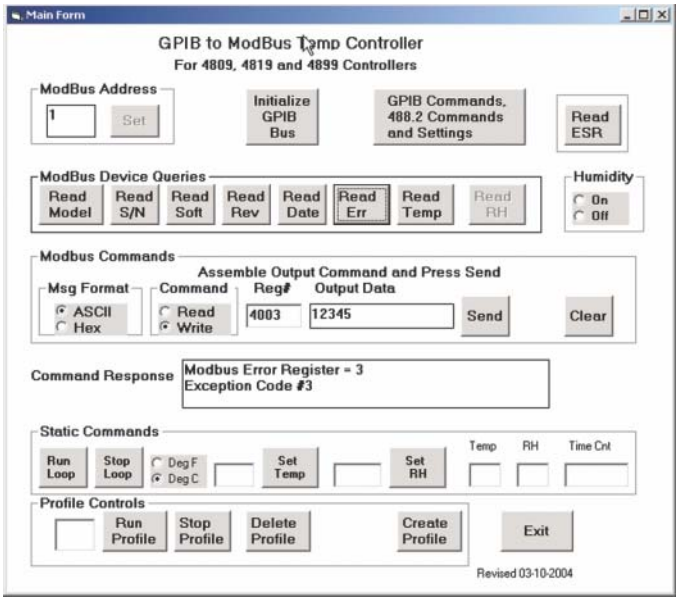

Figure 6 Main Form with Modbus Error Register Response

STATIC TEMPERATURE AND HUMIDITY

The Static Commands box contains controls that set temperature and humidity setpoints and display the current values and time readings. To verify the temperature control functions, the F4 was connected to a AC relay which powered a lamp that was used to heat a small Temperature Chamber. A type J thermocouple was used to sense the chamber temperature. Figure 9 shows the F4 test setup.

Note that this routine does not modify any control loop parameters. Those parameters are set by your temperature chamber supplier and should only be changed after consulting with your supplier.

Before running the Static Command Loop, select the temperature units and set the desired temperature and humidity by entering a value in the box and clicking the Set Temp or the Set RH button. Clicking Run Loop enables the Visual Basic's timer function to read and display temperature, humidity and time from the F4 once a second. If the temperature setpoint value was entered with a decimal point, the F4 display and the response will show a decimal point. If the setpoint was entered without a decimal point, the F4 and the response show just integers. Figure 7 shows a 40 °setpoint and the Temperature and Time displays active.

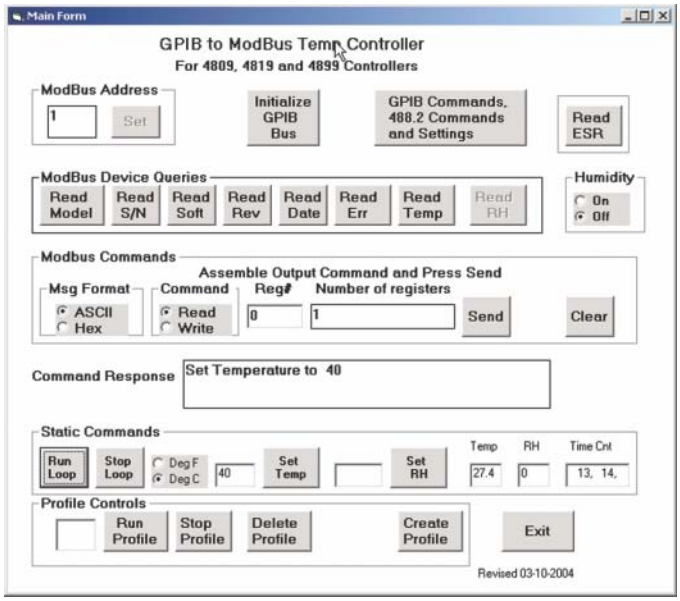

Figure 7 Main Form with a Static Setpoint and Temperature Display

You can stop the static set point operation by entering OFF in the temperature box and clicking the Set SetPt control. The reading loop is stopped by clicking the Stop Loop button.

GENERATING A PROFILE

The Profile Controls box on the Main Form contains controls to create, run or delete temperature profiles. Click the Create Profile button to bring up the Profile Generator form shown in Figure 8.

To create a profile, enter an unused profile number in the box (start with number 1) and click Set Profile. If the F4 responds that this is a valid unused profile number, the Step Generator box controls are enabled. If not, go back to the Main Form and delete ALL profiles by entering ALL into the profiles text box and clicking Delete.

The Step Choice coding in the TempCtlr program follows the Watlow suggested flowchart for programming a profile. The example program implements the Autostart, Time Ramp, Ramp Rate, Soak and End steps. The user can add the Jump step if it is needed in his application.

To start a profile, you can use either Autostart which starts the profile at a preset time of day (like an alarm clock) or Time. Time is really the Time Ramp function but it will do an immediate start profile step when programmed for a short period of time. The TempCltr program defaults to 10 seconds.

Click the Time option to create the first step. This displays the Start Time boxes. Leave the start time at 00:00:10 for an immediate start or enter a short delay time. Click Set. Enter the current temperature in the Temperature text box and click Set Temp to complete step 1. This enables the Rate and Soak steps. Each subsequent step increments the Step number in the Step Number box.

For a temperature ramp you can use either a Time or Rate step. Time works as described above where you enter a ramp time and ending temperature. Follow the above instructions but enter a real ramp time and an ending temperature. Click either the Ramp or Soak options to add more steps to the profile. For a Rate Ramp step, enter the Ramp rate in degrees/minute * 10. (e.g. Enter 50 for 5 degrees per minute). Click Set. Enter the end temperature value and click Set Temp. Figure 8 shows the Profile generator form making a 5 degree/minute Rate ramp.

For a Soak step, enter the Soak Time and click Set.

To end the profile, click End which makes the last step and saves the profile. Return to the Main Form to run the Profile.

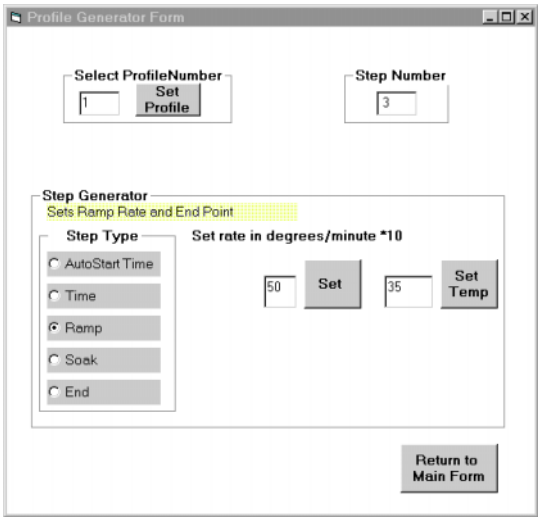

Figure 8 Profile Generator Form making a Ramp

RUNNING THE PROFILE

On the Main Form, enter the profile number in the profile text box and click Run Profile to start the profile. The profile can be stopped by clicking the Stop Profile control. To delete a profile, enter the profile number in the profile text box and click Delete. Click Confirm to finish the delete process. If things get too mixed up, enter ALL in the profile text box and click Delete to erase all profiles.

VISUAL BASIC TempCtlr PROJECT

The TempCtlr program can be downloaded from ICS's website as VB5_TempCtlr.zip. Load the program into a temporary directory and unzip it to extract the TempCtlr files. The TempCtlr.exe file is an executable version of the TempCtlr program that runs with ICS, ComputerBoards or National Instruments GPIB Controller Cards.

Visual Basic programs are referred to as projects. The TempCtlr project contains three .frm files plus the three .bas files: NIGlobal.bas, VBib-32.bas and Module1.bas. The frm files contain the controls (buttons, frames, text boxes etc.) and code to execute the control functions. frmMAIN is the top or main form. The NIGlobal.bas file contains the GPIB constants and variables used in all Visual Basic programs. The VBib-32.bas file contains the library declarations for the calls to the GPIB DLL. The Module1.bas file contains the program's global variables and the SendCmd, ReceiveResp and gpiberr subroutines.

To adapt the TempCtlr program to a GPIB card with another command set, you have to replace the NIGlobal.bas and

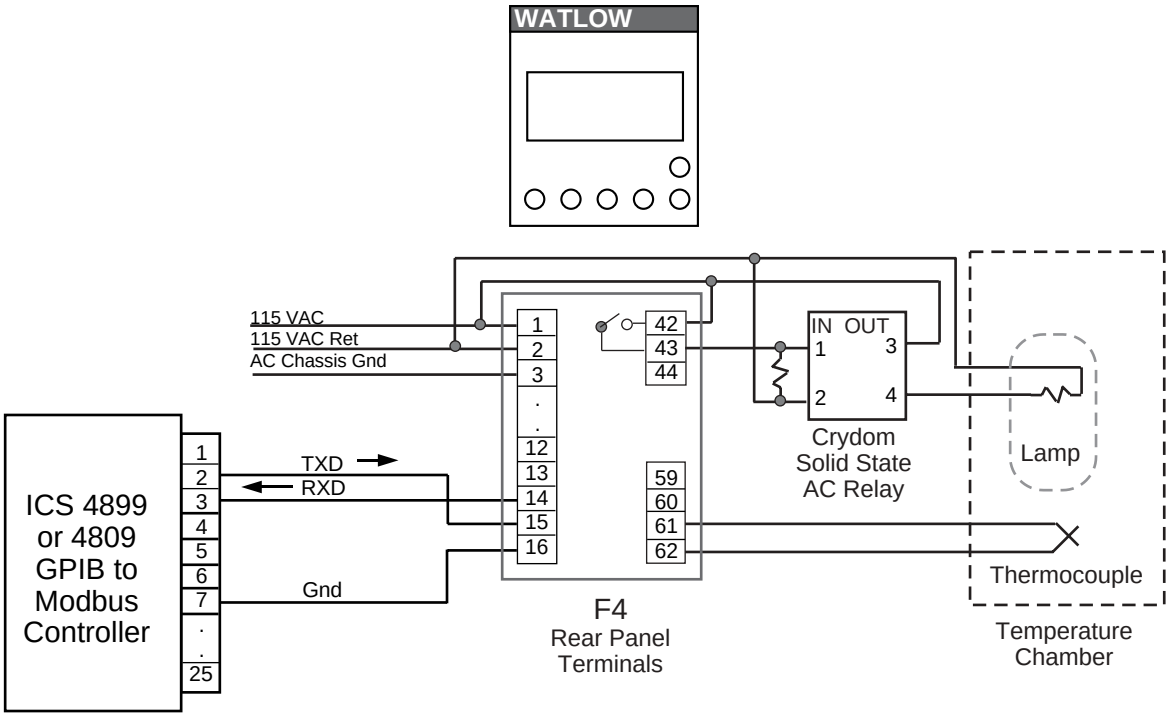

Figure 9 TempCtlr Program Test Connections

VBib-32.bas files with the equivalent file(s) for the other card. You will also have to change the calls in the rest of the program to call the equivalent functions for the new GPIB card. This is fairly simple to do since most of the GPIB calls are to the SendCmd and ReceiveResp subroutines in the Module1.bas file. After changing these two subroutines, all you have to do is to correct the initialization commands in the Main Form and the 488.2 commands in the 488.2 Form. Follow the instructions from the GPIB Card manufacturer when preparing Visual Basic programs for your card.

WRITING THE APPLICATION PROGRAM

The following suggestions apply to an application program for the 4809, 4819 or 4899 GPIB-to-Modbus Interfaces. The initialization routine should set all of the fixed parameters such as timeouts, format, ESE settings, Modbus device address etc. and verify presence of all devices. When some of the Send and Receive commands are executed in the example program, the error variable is tested after the command to check for errors. This only assures the command was correctly executed. It does not mean that the 4809/4819/4899 or the Modbus device liked the command or executed it. Add additional tests to read the Modbus Error Register in the 4809/4819/4899 or to check the 4809/4819/4899's ESR register to determine if it or the Modbus device had a problem with the command. Bit 6 will be on if there was a Modbus error. One method of doing this is to enable SRQ generation when an error occurs and then check the SRQ line after each command. It will be asserted if the 4899/4809/4819 detected a Modbus error. To reduce your coding effort, the SRQ and Modbus error tests can be put into the command subroutines in Module1.bas.

Next set the Modbus device address to match the address of the actual Modbus device and send the C command with the address value. If you have only one device, you will not need a text box to enter the address. The 4809/4819/4899 defaults to a value of 1 which should match the Watlow's F4 default value. If you have multiple Modbus devices, the cmdChgAddr routine can serve as an address example. In either case, the C command and Modbus device address have to be sent to the 4809/4819/4899 before executing a read or write command. You can skip this step if you use the default address of 1 or have saved the Modbus address in the 4809/4819/4899's memory with the *SAV 0 command.

The cmdRdModel or cmdSend routines show how to read or write to a Modbus device register. Be sure the 4809/4819/4899's format is set correctly before executing the R or W commands. The recommendation is to put a format setting command in the initialization procedure after reading the unit's IDN message. Read routines should provide a short delay after sending the query (R? reg,n) before addressing the 4809, 4819 or 4899 to talk to give the Modbus device time to receive the message and to respond to it.

When setting the Modbus timeout, check your Modbus device's manual for its expected response times and set the timeout for the worst case time plus 2 milliseconds per character in the longest message packet. Add a safety margin of 30 to 50 milliseconds. The 4809/4819/4899s default to 100 milliseconds. Some users have reported that 300 ms is a more reliable timeout value for the Watlow F4 Controller as it occasionally gets busy with an internal calibration routine.

The Set Static Setpoint and the Profile generating routines can be incorporated directly into your test program. The example used the simple ON-OFF control option as it was the easiest to implement with the lamp heat source. However in a real system, the user may want to use one of the more sophisticated PID functions.

SUMMARY

This application note has shown how ICS's Model 4809, 4819 and 4899 GPIB-to-Modbus Controllers are used to control Modbus slave devices over a serial link. This note also describes an example Visual Basic program that can be used with a Watlow F4 to control process temperatures. While Watlow's new F4 series Process Controller was used as the Modbus device to develop the program, most of the program and its concepts should work with other Modbus devices. The example program can also be used as a starting point for the user's own temperature control program. The Visual Basic source files and an executable copy of the program can be downloaded from ICS's website at http://www.icselect.com as a self exploding zip file.