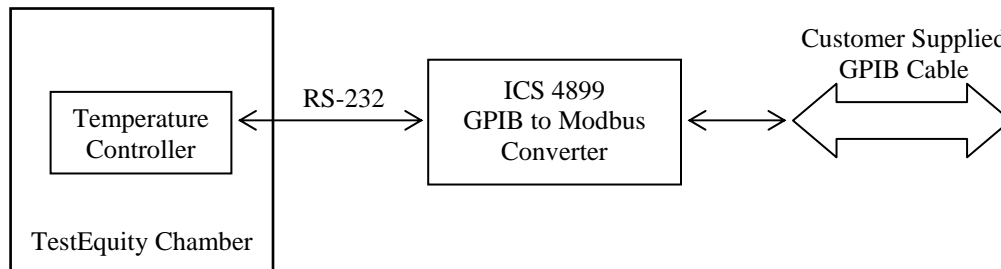


TESTEQUITY

External GPIB Interface Model TE-1052 Instructions

Introduction

The GPIB communications is achieved through an ICS Electronics 4899 GPIB-to-Modbus Interface Converter. The ICS 4899 converts GPIB commands to serial Modbus commands that are transmitted over RS-232. The ICS 4899 also takes care of calculating block checksums that are required for communications to and from the Series F4 or Series 96 Temperature Controller.



Communications Parameters

The ICS 4899 is set at the factory for 9600 baud. The chamber's Temperature Controller must also be configured for 9600 baud. All chambers currently ship configured for 9600 baud. If you need to change the setting on an older 1000 Series, models 105A and 115 chambers, this is located in the F4 Temperature Controller's Main Page\Go to Setup\Communications. If you are unable to enter this menu or change the setting from 19200, then this menu has been locked. If a password is required, call TestEquity. See the "Series F4 User's Manual" for instructions on how to clear the lock and navigate through the menus.

It is important to remember that GPIB interface messages communicate directly with the ICS 4899 as the talk-listen addressed device. The Temperature Controller is NOT the GPIB device. Interface messages that arrive at the ICS 4899 can be interpreted and used locally by the ICS 4899, or interpreted and re-transmitted to the Temperature Controller.

Data returning from the Temperature Controller is received by the ICS 4899 which then examines the block checksum characters, strips them off, and re-transmits the desired data to the GPIB interface for use by the controller.

Modbus Communication

The Temperature Controller relies on a communication protocol called Modbus, which offers multidrop serial capability for serial devices, reliable communication, and standardized commands.

Since Modbus allows more than one device to share communications ports, each device requires a Modbus address. The C command of the ICS 4899 sets the Modbus address of the Temperature Controller. From that point, the ICS 4899 will use that address to communicate to the Temperature Controller. Each Modbus device has its own characteristics and data items that are examined and/or set. Data for the device is organized into registers. Register data is set by sending a Write command to a specific register, and is examined by sending a Read command to a register. The commands to read and write data in registers are performed by commands sent to the ICS 4899. These commands do resemble the final command that the Temperature Controller sees, but the ICS 4899 also generates and sends a block checksum based on all the characters of each message and sends it to the Temperature Controller.

Modbus Register Read / Write Commands:

R? reg, n Read register command
 Reg = Modbus register
 n = number of registers to send

W reg, data Write register command
 Reg = Modbus register
 Data = ASCII data written as 16-bit decimal value

NOTE: The '?' is optional and is included so programs like ICS's GPIB Keyboard control programs can automatically read back and display the response from a query.

Common Modbus Registers

- Actual chamber temperature reading: 100.
- Actual chamber humidity reading: 104. (Model 1007H and 1207C only)
- Static temperature set point: 300.
- Static humidity set point: 319. (Model 1007H and 1207C only)
- Temperature set point during a profile: 4122. (Except model 105 with 97 controller)
- Temperature set point during a profile: 5009. (Model 105 with 97 controller only)
- Humidity set point during a profile: 4123. (Model 1007H and 1207C only)
- Digital Output 1 (Event 1) in static set point mode: 2000. (Not applicable for model 105 with 97 controller)

The entire listing of Modbus registers for the Series F4 Temperature Controller are found in the Series F4 User's Manual, Chapter 7.

The entire listing of Modbus registers for the Series 96 Temperature Controller are found in the Series 96 User's Manual, pages 7.7 and A.3.

Programming Sequences

Communicating to the Temperature Controller with GPIB requires commands to be made in a standardized sequence. The sequence described here is language-independent, meaning that descriptions of all of the possible languages are not given. Instead, given GPIB commands are shown as strings, with an example language given to show program flow.

!	Comments are indicated with exclamation point
<i>Samp_488</i>	Command to GPIB interface card
STRING	Strings sent to GPIB device at specified address
<lf>	Individual linefeed character
<cr>	Individual carriage return character
language	Sample language element

Reset & Initialization Example

This sequence should be performed once before establishing communications to the Temperature Controller.

```
Command: IFC                                ! Reset GPIB interface card
Command: *RST <lf>                          ! Send reset command to ICS 4809
Command: D300<lf>                            ! Set ICS 4899 timeout to 300 ms
```

Reading Chamber Temperature Example

The Temperature Controller sends data over the Modbus interface with an implied decimal point. It is left to the user to remember this, and to scale the data accordingly. The Temperature Controller has been configured by TestEquity to display one decimal point in the temperature reading. If the configuration should change, then the program would not report the correct temperature reading. To remedy this problem, always read the number of decimal points used by Analog Input 1 before running the program.

```
Command: R? 606,1<lf>                        ! Send read command to register 606
Response: iData <lf>                        ! Returned variable iData represents ASCII
                                           ! numeric characters that are converted in this
                                           ! example to integer.
                                           ! 0 = no decimal point
                                           ! 1 = one decimal point
```

```
Command: R? 100,1<lf>                       ! Send read command to register 100
Response: fTemp <lf>                       ! Read the value of register 100. fTemp
                                           ! represents ASCII numeric characters,
                                           ! converted to float type.
```

```
if iData = 1 then                            ! Check iData
    fTemp = fTemp / 10                       ! 1=means that data has decimal and
endif                                         ! we need to divide by ten.
print "Chamber Temp is", fTemp
```

Set Chamber Setpoint Examples

Command: **W300,230 <lf>** ! Send write command to register 300 and
! change the setpoint to 23.0 degrees.

Command: **W300,1005 <lf>** ! Send write command to register 300 and
! change the setpoint to 100.5 degrees.

Command: **W300,-255 <lf>** ! Send write command to register 300 and
! change the setpoint to -25.5 degrees.

Additional Resources

The ICS GPIB Modbus Interface manual provides detailed information on the 4899 Interface Converter. Additional resources and LabView drivers can be downloaded and/or purchased from <http://www.icselect.com>.

IMPORTANT NOTE: The examples shown in documentation from ICS are for illustration purposes only. They do not represent the correct setup or configuration for TestEquity chambers. Sample programs from ICS may change critical setup parameters, resulting in improper chamber operation. They are provided as a guideline for how to write your own programs only. Correct setup parameters are documented in the TestEquity chamber manuals.

ICS Manual Errata

To reflect the standard decimal point configuration of 0.0° in TestEquity chambers, the ICS Electronics manual, Page 3-24, should be corrected as follows:

3.8.4 Writing to the Modbus Device

The nature of the command depends upon the specific Modbus device. Simple writes are handled with the W command. In the following example, a value of 50.0° is written to register 300.
i.e.

W300,500 'sets temperature setpoint

C7; W300,500 'concatenated command

Writes to multiple registers are possible with the WB command.