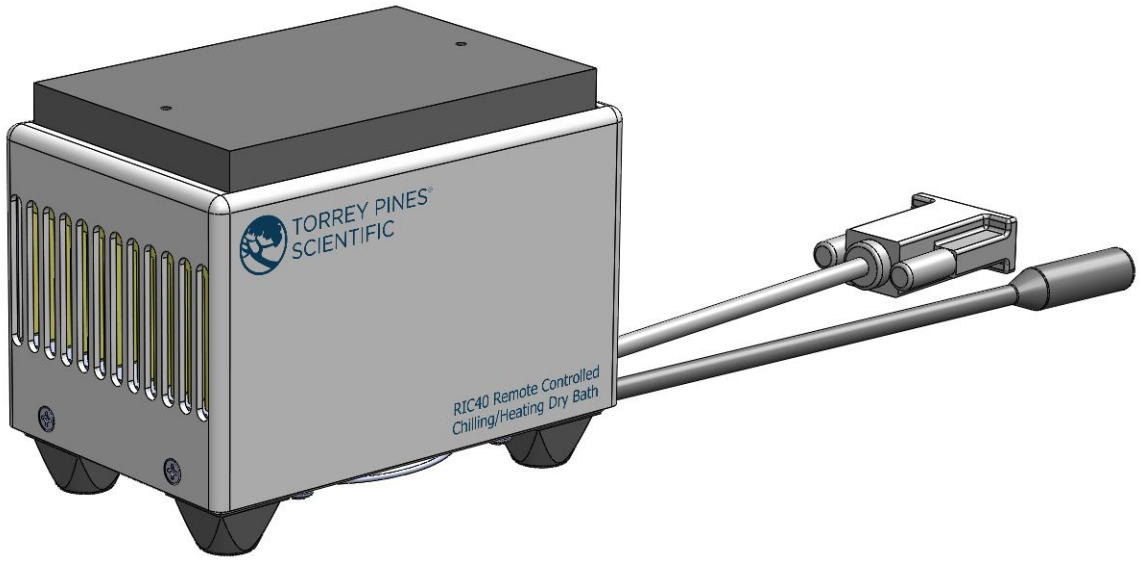
RIC40 Programming Manual

Rev A (June 30, 2020)

Applicable to RIC40 and RIC40XR firmware v1.0



TORREY PINES SCIENTIFIC, INC.
2713 Loker Ave. West
Carlsbad, CA 92010

TELEPHONE: (760)-930-9400
TOLL FREE: (866)-573-9104
FAX: (760)-930-9480
E-Mail: info@torreypinesscientific.com
Web site: www.torreypinesscientific.com

Features

Identification	Unit Model and version Unit Serial Number User Defined String (10 chars max)
Set Point (C)	Tenth Degree Precision Settable Range: -10.0 to +100.0 “off” (idle mode—temp controller turned off)
Plate Temperature (C)	Tenth Degree Precision
Calibration	Calibrated at Factory 2 Point User Settable
Timer	User Settable Start Time (format -- hh:mm:ss) Range: 00:00:00 to 24:59:59 Count Down (00:00:00 min) Count Up (24:59:59 max)
LED Event Notification	LED Blinking → Plate Temp Changing LED Steady On →Plate Temp Within 0.2C of Set Point for 60 seconds
Serial Event Notification	Plate Temp (User settable period 0-99:59) TEMP_STEADY (User settable enable/disable) TIMER=0 (User settable enable/disable)
Non-Volatile Memory	On Power Down, the following values are stored: • set point • high and low calibration values • event broadcast settings

Manual Conventions

<CR>
The notation "<CR> " refers to the ASCII character for "carriage return" which is decimal 13 or hex D. Every valid RIC40 command must be terminated by this character. In HyperTerminal the character is sent when the "enter" key is pressed. Programs written in C for example, typically send this character when "\r" is appended to the transmitted command string.

<LF>
The notation "<LF> " refers to the ASCII character for "line feed" or "new line" which is decimal 10 or hex A. Every string that is returned from the RIC40 will be terminated with this character. Actually, every string that is returned from the RIC40 will be terminated with a <CR> then a <LF>. In HyperTerminal the combined characters will cause the cursor to return to the beginning of the next line. Programs written in C for example, can use this character (often "\n") for parsing returned strings.

(string)
Characters within parenthesis are strings consisting of 7-bit ASCII characters and the string itself is the argument for a command. The length of the string is clarified in the discussion of the command and the two parenthesis are not included in the transmission of the command string.

Serial Interface

The RIC40 line of products are controlled using a simple serial RS232 interface. Each command sent to the RIC40 must be terminated with an ASCII carriage return character and every response string from the RIC40 will be terminated with a character pair consisting of an ASCII carriage return and ASCII linefeed character. All commands described within this manual are case sensitive. When a command is successfully received, an appropriate string will be returned. If a command is received with a syntax error, the character "e" will be returned to indicate the error.

The command set was developed to provide functionality that will enable multiple interface and programming scenarios. For example, if a user wishes to use a simple application such as HyperTerminal to control a single unit, linefeed and carriage control characters included in the command and return strings enable clean formatting within the HyperTerminal User Interface. Additionally, commands are included to enable different programming methodologies such as polled response or event driven responses. For example, a program could be written to set a new set point then read the plate temperature at some time interval (polling) and take specific actions when the plate reaches various values along the way. Or a program could be written to set a new set point then the program could do something else until the set point is reached and the "TEMP_STEADY" message is received from the unit (event driven). **NOTE: If the user written controller is event driven and it "misses" the TEMP_STEADY event message, it may hang waiting for the event that it missed. Asynchronous Serial communication "misses" are not that uncommon so it is recommended that the controller algorithm have provisions like a wait-for-event timeout and if a timeout occurs, code to check the plate temperature and compare that to the set point temperature then take the appropriate action. As a rule of thumb, the most robust coding approach is to set the set point, read and verify the new set point, then continuously poll the plate temperature for routines that do things like determine the user established steady condition and for routines that continuously monitor the temperature. Code written using set and verify with polling will recover easily if an occasional corruption of the serial communication occurs.**

When the Serial COM settings are configured correctly on the external control computer, every properly formatted command sent to the RIC40 will return either the requested data string or "ok" to acknowledge the successful reception of the command. The character "e" is returned when the RIC40 does not recognize the received command string. The code running in the external computer can use the returned strings for handshaking and/or to verify that the command was properly executed.

If a serial terminal emulator application like HyperTerminal is used to send commands and receive responses, the command x<CR> will put the RIC40 into "Terminal Mode". Terminal Mode will enhance the screen data captured by HyperTerminal by displaying the RIC40 response on the next line from the line that issued the command. Terminal Mode is cancelled when the RIC40 is powered off.

COM Settings

9600 baud
1 stop bit
no parity
no hardware handshake
50ms delay after each line sent (after each ";")

Unit Identification Commands

Command: **v**
Function: **Return RIC40 Model and Version**
Description: When the command v<CR> is received by the RIC40 unit, the model number and the firmware version will be returned in a text string terminated by <CR><LF>. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example:
send: v<CR>
returned: RIC40 v1.00 <CR><LF>

Command: **V**
Function: **Return Serial Number**
Description: When the command V<CR> is received by the RIC40 unit, the 8 character serial number will be returned in a text string terminated by <CR><LF>. The serial number for every RIC40 unit is unique. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Return the 8 char serial number:
send: V<CR>
returned:12345678 <CR><LF>

Command: **>(user defined string)**
Function: **Store User ID String**
Description: Each unit has a unique serial number (ref cmd V) but it may be helpful to assign a custom "name" to identify a unit. This command enables a string of up to 10 characters to be stored with the unit. When the command >(user defined string)<CR> is received by the RIC40 unit, the string is stored and the unit returns ok<CR><LF>. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example—assign RIC40 the name "Unit 1"
send: Unit 1<CR>
returned: ok <CR><LF>

Example—verify that the name is "Unit 1"
send: ><CR>
returned: Unit 1<CR><LF>

Command: **>**
Function: **Return User ID String**
Description: When the command **>**<CR> is received by the RIC40 unit, the User ID String will be returned in a text string terminated by <CR><LF>. If no string has been stored, 10 space characters will be returned followed by <CR><LF>. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example—Return the string that was stored using the >(user defined string) command:
send: ><CR>
returned: Unit 1<CR><LF> if "Unit 1" was previously stored
returned: <CR><LF> if no user string has been stored

Temperature Commands

Command: **s**
Function: **Return Set Point Temperature**
Description: When the command s<CR> is received by the RIC40 unit, the current set point temperature will be returned in a text string terminated by <CR><LF>. The setpoint format length is variable depending on the number of digits in the setpoint temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Note: Putting the RIC40 into Idle Mode (see "Command: i") will turn off the temperature controller and the set point will be read as "off".

Example—Return the current set point:
send: s<CR>
returned example 1: -10.0 <CR><LF>
returned example 2: 9.3 <CR><LF>
returned example 3: 100.0 <CR><LF>
returned example 4: off<CR><LF> if unit is in Idle Mode (see "Command: i")

Command: **n**
Function: **Set and Store New Set Point Temperature**
Description: When the command n(new_temperature)<CR> is received by the RIC40 unit, the set point will be changed to new_temperature and the text string "ok" will be returned terminated by <CR><LF>. The format for new_temperature is variable depending on the number of digits in the desired new set point temperature and whether the value is negative. The value must always include a decimal point and one character representing the fractional tenth value. The settable range is -10.0C to 100.0C.

If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example1— Change set point to -10.0C:
send: n-10.0<CR>
returned: ok <CR><LF>

Example2 – Change set point to 9.3C :
send: n9.3<CR>
returned: ok <CR><LF>

Example3 – Change set point to 100.0C :
send: n100.0<CR>
returned: ok <CR><LF>

Command: **i**
Function: **Set RIC40 Unit to Idle Mode**
Description: When the command i<CR> is received by the RIC40 unit, the temperature controller will be switched “off”, meaning that the plate will no longer heat or cool. In Idle Mode, the unit will report a set point of “off”. To exit Idle Mode, simply set the set point to a new value using cmd n. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Set Idle Mode:
send: i<CR>
returned: ok <CR><LF>

Example – verify Idle Mode:
send: s<CR>
returned: off<CR><LF>

Example – exit Idle Mode by setting new set point to 25.0C
send: n25.0<CR>
returned: ok <CR><LF>

Example – verify no longer in Idle Mode and set point is 25.0C:
send: s<CR>
returned: 25.0<CR><LF>

Timer Commands

Command: **a**
Function: **Return Current Timer Value**
Description: When the command a<CR> is received by the RIC40 unit, the current timer value will be returned in a text string terminated by <CR><LF>. The timer string is a fixed width of 8 characters in the format hh:mm:ss. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – the current timer value is 1 hour, 32 minutes, 15 seconds:
send: a<CR>
returned: 01:32:15 <CR><LF>

Example – verify that the current timer has been cleared (see Command: ac):
send: a<CR>
returned: 00:00:00 <CR><LF>

Command: **a(hh:mm:ss)**
Function: **Set Current Timer Value**
Description: When the command a(hh:mm:ss)<CR> is received by the RIC40 unit, the current timer value will be set to the new value in the 8 character format hh:mm:ss. The settable timer range is 00:00:00 to 24:59:59. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Set the current timer value to 1 hour, 32 minutes, 15 seconds:
send: a01:32:15<CR>
returned: ok <CR><LF>

Command: **au**
Function: **Start Timer—Count Up**
Description: When the command au<CR> is received by the RIC40 unit, the current timer value will increment every second. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Verify current timer value is 00:00:00
send: a<CR>
returned: 00:00:00 <CR><LF>

Example – Start incrementing timer
send: au<CR>
returned: ok<CR><LF>

Example – Check timer after 5 seconds
send: a<CR>
returned: 00:00:05 <CR><LF>

Command: **ad**
Function: **Start Timer—Count Down**
Description: When the command au<CR> is received by the RIC40 unit, the current timer value will decrement every second. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Verify current timer value is 00:30:00

send: a<CR>
returned: 00:30:00 <CR><LF>

Example – Start decrementing timer
send: ad<CR>
returned: ok<CR><LF>

Example – Check timer after 5 seconds
send: a<CR>
returned: 00:29:55 <CR><LF>

Command: **ap**
Function: **Pause or Stop Timer**
Description: When the command ap<CR> is received by the RIC40 unit, the current timer value will stop decrementing or incrementing. To restart the timer from the current timer value, send either an au or ad command. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Stop Timer
send: ad<CR>
returned: ok<CR><LF>

Example – Read Current Timer Value
send: a<CR>
returned: 00:29:55 <CR><LF>

Command: **ac**
Function: **Clear the Current Timer Value**
Description: When the command ac<CR> is received by the RIC40 unit, the current timer value will be set to 00:00:00. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Clear the Current Timer Value
send: ac<CR>
returned: ok<CR><LF>

Example – Read Current Timer Value
send: a<CR>
returned: 00:00:00 <CR><LF>

Calibration Commands

Command: **R**
Function: **Return the High Calibration Point Temperature**
Description: When the command R<CR> is received by the RIC40 unit, the temperature at which the high calibration point was established will be returned in a text string terminated by <CR><LF>. The high calibration point temperature format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Determine the high temperature calibration point:
send: R<CR>
returned: 75.0 <CR><LF> (the RIC40 was calibrated at 75.0C)
returned: 100.0 <CR><LF> (the RIC40 has the default High Calibration Temp)

Command: **r**
Function: **Return the Low Calibration Point Temperature**
Description: When the command r<CR> is received by the RIC40 unit, the temperature at which the low calibration point was established will be returned in a text string terminated by <CR><LF>. The low calibration point temperature format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Determine the low temperature calibration point:
send: r<CR>
returned: 10.0 <CR><LF> (the RIC40 was calibrated at 10.0C)
returned: -10.0 <CR><LF> (the RIC40 has the default Low Calibration Temp)

Command: **T**
Function: **Return the Measured Temperature at the High Calibration Point**
Description: When the command T<CR> is received by the RIC40 unit, the measured temperature when the RIC40 was at the high calibration point temperature will be returned in a text string terminated by <CR><LF>. The measured temperature format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Determine the measured temperature at the high calibration point of 75.0C:
send: T<CR>
returned: 73.2 <CR><LF> (the plate was measured to be 73.2C when the RIC40 was set to 75.0C)

Command: **T(measured_temp)**
Function: **Set the Measured Temperature at the High Calibration Point**
Description: When the command T(measured_temp)<CR> is received by the RIC40 unit, measured_temp will be stored and used with the High Calibration Point temperature to calculate calibration offsets. The measured_temp format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

**IMPORTANT NOTE: Both the RIC40 temperature and the calibration measurement temperature must be steady at the High Calibration Point for at least 10 minutes before the measured temperature is entered. Additional settling time may be required depending on the thermal conduction of the material at the calibration measurement point. For example, if a liquid sample is the measurement point for the calibration, more time may be required to reach a stable measurement temperature. If the unit or the measured value are not steady at the High Calibration Point temperature when the measured value is entered, significant calibration error may result.

Example – The RIC40 High Calibration Point is 75.0C, the measured temp when the unit is steady at 75.0C is 73.2C. Set the measured High Calibration Point temperature:

send: T73.2<CR>
returned: ok <CR><LF>

Example – Verify that the Measured Temperature at the High Cal Point is 73.2
send: T<CR>
returned: 73.2<CR><LF>

Command: **t**
Function: **Return the Measured Temperature at the Low Calibration Point**
Description: When the command t<CR> is received by the RIC40 unit, the measured temperature when the RIC40 was at the low calibration point temperature will be returned in a text string terminated by <CR><LF>. The measured temperature format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – Determine the measured temperature at the low calibration point of 10.0C:
send: t<CR>
returned: 11.3 <CR><LF> (the plate was measured to be 11.3C when the RIC40 was set to 10.0C)

Command: **t(measured_temp)**
Function: **Set the Measured Temperature at the Low Calibration Point**
Description: When the command t(measured_temp)<CR> is received by the RIC40 unit, measured_temp will be stored and used with the Low Calibration Point temperature to calculate calibration offsets. The measured_temp format length is variable depending on the number of digits in the calibration temperature and whether the value is negative. The value will always include a decimal point and one character representing the fractional tenth value. If the command is not received in the proper syntax, e<CR><LF> will be returned.

**IMPORTANT NOTE: Both the RIC40 temperature and the calibration measurement temperature must be steady at the Low Calibration Point for at least 10 minutes before the measured temperature is entered. Additional settling time may be required depending on the thermal conduction of the material at the calibration measurement point. For example, if a liquid sample is the measurement point for the calibration, more time may be required to reach a stable measurement temperature. If the unit or the measured value are not steady at the Low Calibration Point temperature when the measured value is entered, significant calibration error may result.

Example – The RIC40 Low Calibration Point is 10.0C, the measured temp when the unit is steady at 10.0C is 11.3C. Set the measured Low Calibration Point temperature:

send: t11.3<CR>
returned: ok <CR><LF>

Example – Verify that the Measured Temperature at the Low Cal Point is 11.1
send: t<CR>
returned: 11.3<CR><LF>

Command: **H**
Function: **Reset the High Temperature Calibration Points to Default**
Description: When the command H<CR> is received by the RIC40 unit, the High Temperature Calibration Point and the Measured Temperature at the High Calibration Point will be set to the default value of 100.0C. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – set the High Calibration Point to Default (100.0C)
send: H<CR>
returned: ok <CR><LF>

Command: **h**
Function: **Reset the Low Temperature Calibration Points to Default**
Description: When the command h<CR> is received by the RIC40 unit, the Low Temperature Calibration Point and the Measured Temperature at the Low Calibration Point will be set to the default value of -10.0C. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – set the Low Calibration Point to Default (-10.0C)
send: h<CR>
returned: ok <CR><LF>

Macro Commands

Command: **m**
Function: **Macro to return all 4 calibration values**
Description: When the command m<CR> is received by the RIC40 unit, the Low Temperature Calibration Point, the Measured Temperature at the Low Calibration Point, the High Temperature Calibration Point, and the Measured Temperature at the High Calibration Point will be returned in that order in a string delimited by commas and terminated by <CR><LF>. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – return all cal points from the previous examples
send: m<CR>
returned: 10.0,11.3,75.0,73.2 <CR><LF>

Command: **M**
Function: **Macro to return all Status, SP, PT, Timer**
Description: When the command M<CR> is received by the RIC40 unit, a string containing the Status String, Set Point Temperature, Plate Temperature, and Current Timer Value will be returned delimited by commas and terminated by <CR><LF>. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – return Status, SP, PT, Timer
send: M<CR>
returned: StbLH,-10.0,-10.0,00:04:13<CR><LF> (temp is steady, timer not running, unit not broadcasting, low temp cal done, high temp cal done, set point is -10.0, plate temp is -10.0, the timer value is 4 mins 13 sec)

Event Notification Commands

Command: **b(mm:ss)**
Function: **Broadcast the Plate Temperature**
Description: When the command b(mm:ss)<CR> is received by the RIC40 unit, the unit will return the plate temperature every time interval defined by mm:ss. To disable the Plate Temperature, set the time interval to 00:00. Sending b<CR> will return the current settings. If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – return the plate temperature every 5 seconds
send: b00:05<CR>
returned: ok <CR><LF>

Example – stop broadcasting the plate temperature
send: b00:00<CR>
returned: ok <CR><LF>

Example – return the current Broadcast Plate Temperature settings
send: b<CR>
returned: 00:00 <CR><LF>

Command: **B(sz)**
Function: **Broadcast Temperature or Timer Events**
Description: When the command B(ab)<CR> is received by the RIC40 unit, the unit will enable or disable Temperature or Timer Events depending on the case of the variables s and z as shown in the table below:

Return string "TEMP_STEADY<CR><LF>" when plate temperature is steady
S = enable
s = disable

Return string "TIMER=0<CR><LF>" when timer reaches zero
Z = enable
z = disable

Sending B<CR> will return the current settings.
If the command is not received in the proper syntax, e<CR><LF> will be returned.

Example – set to broadcast when the temperature is steady at the set point but do not broadcast when the timer has reached zero.
send: BSz<CR>
returned: ok <CR><LF>

Example – return the current Broadcast Event settings
send: B<CR>
returned: Sz <CR><LF>

Utility Commands

Command: **S**
Function: **Return Status String**
Description: When the command S<CR> is received by the RIC40 unit, the unit will return a string of 5 characters in the order stblh<CR><LF> that indicate the current status of the unit as described in the table below:

s = temperature is not steady
S = temperature is steady
t = timer is not running
T = timer is running
b = unit is not broadcasting
B = unit is broadcasting
l = low temp cal has not been done (default values stored)
L = low temp cal has been done
h = high temp cal has not been done (default values stored)
H = high temp cal has been done

Example – return the Status String
send: S<CR>
returned: StbLH <CR><LF> (temp is steady, timer not running, unit not broadcasting, low temp cal done, high temp cal done)

Serial Command Quick Reference Table

			Example	
	Command	**Function**	**sent**	**returned**
ID	**v**	return model and version	v <CR>	RIC40 v1.0 <CR LF>
	V	return serial number	V <CR>	12345678 <CR LF>
	>(abc123...)	set user string (10 chars max)	>UNIT 10 <CR>	ok <CR LF>
	>	return user string	> <CR>	UNIT 10 <CR LF>
Temperature and Timer	**s**	return set point temperature	s <CR>	-10.0 <CR LF> off <CR LF> if idle mode
	n(xxx.x)	set new set point temperature (x.x min)	n25.0 <CR>	ok <CR LF>
	p	return plate temperature	p <CR>	-10.0 <CR LF>
	i	set idle mode (plate is off)	i <CR>	ok <CR LF>
	a	return current timer value	a <CR>	00:04:13 <CR LF>
	a(hh:mm:ss)	set timer value (24:59:59 max)	a00:05:00 <CR>	ok <CR LF>
	au	start timer--count up	au <CR>	ok <CR LF>
	ad	start timer--count down	ad <CR>	ok <CR LF>
	ap	pause/stop timer	ap <CR>	ok <CR LF>
	ac	clear timer (00:00:00)	ac <CR>	ok <CR LF>
	M	macro to return status,sp, pt, timer	M <CR>	StbLH,-10.0,-10.0,00:04:13 <CR LF>
Calibration	**R**	return HIGH cal point temperature	R <CR>	75.0 <CR LF>
	r	return LOW cal point temperature	r <CR>	10.0 <CR LF>
	T(xxx.x)	set measured temperature at HIGH cal point (x.x min)	T73.2 <CR>	ok <CR LF>
	T	return measured RTD temp at HIGH cal point	T <CR>	73.2 <CR LF>
	t(xxx.x)	set measured temperature at LOW cal point (x.x min)	t11.3 <CR>	ok <CR LF>
	t	return measured RTD temp at LOW cal point	t <CR>	11.3 <CR LF>
	H	reset HIGH temp cal points to default (100.0C)	H <CR>	ok <CR LF> (R and T now 100.0)
	h	reset LOW temp cal points to default (-10.0C)	h <CR>	ok <CR LF> (r and t now -10.0)
	m	macro to return all 4 cal values (r,t,R,T<CR>)	m <CR>	10.0,11.3,75.0,73.2 <CR LF>
Event Notification	**b**(mm:ss)	Broadcast Plate Temperature returns Plate Temp every mm:ss, 99:59 max	b00:10 <CR>	plate temp <CR LF> returned every 10 seconds
	B(sz)	Broadcast Temperature or Timer Events Return "TEMP_STEADY" when temp is steady S = enable s = disable Return "TIMER=0" when timer reaches zero Z = enable z = disable	BSz <CR>	TEMP_STEADY <CR LF> returned at event TIMER=0 <CR LF> not returned at event
tility	**S**	return status (stblh<CR>) s = temperature is not steady S = temperature is steady t = timer is not running T = timer is running b = unit is not broadcasting B = unit is broadcasting l = low temp cal not done (default values stored) L = low temp cal has been done h = high temp cal not done (default values stored) H = high temp cal has been done	S <CR>	StbLH <CR LF> --temp is steady --timer not running --unit not broadcasting --unit has been calibrated at low temp --unit has been calibrated at high temp

<CR> is return char (for example: “enter” keyboard press for HyperTerminal, “\r” for C pgms, ASCII hex char “0D”)