Richard Ziegahn

June-July 2018

ricky.ziegahn@gmail.com

Cryostat Temperature Control

General Overview

This program is designed to allow the monitoring and updating of the settings on LAKESHORE 330, LAKESHORE 340, and LAKESHORE 336 temperature controllers. The three temperature controllers listed do not have all the same commands or response times, this program is designed to work with all of them. The 340 and the 336 are very similar commands and functionality, the model 330 differs. The 330 has no alarm capabilities and a single, the 340 and the 336 have 2. The heaters also all have different power ranges. The rest of the differences are in the commands that are sent or queried to/from the temperature controller.

This temperature controller can do the following things:

- Query the model
- Query and change the target temperature(s)
- Query the heater output(s)
- Query and change the heater range(s)
- Query and set alarms (340 and 336 only)
- Query the temperature(s)

When the program starts up, it will query the current state of the machine (target temperature(s), heater range(s), alarm setup), and create and input file. The input file is named according to the model (tempinputXXX.txt, where XXX is the model number). Every 10 seconds, the program will read the text file, and compare the text in the file to what the program has last read from the temperature controller.

To make the next part make sense, it will be necessary to explain the variable nomenclature. If a variable begins with an ‘i’, it means it has been inputted and may or may not have been received by the temperature controller. If a variable begins with an ‘r’, it means that it has been read from the machine and represents the current state. Read variables and inputted variables will now be referred to as ‘r’ and ‘i’ in explanations.

If a change is made to the text file (r != i), it will write it the differences to the temperature controller (nothing is written if there are no differences; if a change is made to a single parameter, only that one will be written because r != i for that variable only). For each parameter, after writing, it will query the device to compare the read value (r) to that in the text file (i) (note that beforehand, it will check that the inputted value is in the correct format in the textfile). If it is not the same, it indicates that the command failed, the program will try and update it one more time; if it fails again, it will wait until the next cycle to try again and display and error. If it is the same, on the next check it will not be picked up as a difference because the read variable is now the same as the inputted variable (r == i). The read variables are the values that are displayed in the python window, so the value in the text file will not display until the temperature controller has received and returned it. The program will also keep trying until the temperature controller has received successfully (r == i).

Command overview

Query Model:

330: *IDN?

340: *IDN?

336: *IDN?

Set Temperature Setpoint:

330: INPUT: SETP[set point]

340: INPUT: SETP <loop>,<value>

336: INPUT: SETP <output>,<value>

Query Temperature Setpoint:

330: INPUT: SETP?

RETURN: <value>

340: INPUT SETP? <loop>

RETURN: <value>

336: INPUT SETP? <output>

RETURN: <value>

Query Heater Percentage:

330: INPUT: HEAT?

RETURN: <heater value> #increments of 5%

340: INPUT: HTR?

RETURN: <heater value>

336: INPUT:HTR? <output>

RETURN: <heater value>

Set Heater Range:

330: INPUT: RANGE[heater range]

0 is off, 1 is low, 2 is medium, 3 is high

340: INPUT: RANGE <range>

0 is off; 1 is 2.5mW; 2 is 25mW; 3 is 250mW; 4 is 2.5W; 5 is 25W

336: INPUT: RANGE <output>,<range>

Output 1: 0 is off; 1 is 0.1W; 2 is 10W; 3 is 100W

Output 2: 0 is off; 1 is 0.5W; 2 is 5W; 3 is 50W

Query Heater Range:

330: INPUT: RANG?

RETURN: <range>

340: INPUT: RANGE?

RETURN: <range>

336: INPUT: RANGE? <output>

RETURN: <range>

Set Alarm:

330: N/A

340: INPUT: ALARM <input>, <off/on>, <source>, <high value>, <low value>, <latch enable>, <relay>

336: INPUT: ALARM <input>,<off/on>,<high value>,<low value>,<deadband>,<latch enable>,<audible>,<visible>

Query Alarm:

330: N/A

340: INPUT: ALARM? <input>

RETURN: <off/on>, <source>, <high value>, <low value>, <latch enable>, <relay enable>

336: INPUT: ALARM? <input>

RETURN: <off/on>,<high value>,<low value>,<deadband>,<latch enable>,<audible>,<visible>