CHEM Control Box USB Interface details

**Prerequisites:**

You will need the MBED virtual serial driver:

<http://mbed.org/handbook/Windows-serial-configuration>

This allows one to establish a connection to the CHEM CONTROL BOX via a windows serial. Once the driver is installed, a serial port will show up in the windows devices manager as an “MBED COM PORT”.

Settings:

Once the driver is installed, find the COM port in the windows device manager. You should be able to connect to this COM port with a terminal emulator such as PUTTY. The baud rate should be set to 115200, 8 data bits, 1 stop bit, no parity.

The CHEM CONTROL BOX emulates a simple command line interface over the COM port. This allows one to issue commands via a operator typing in commands or via some other automated means (A Labview program).

You can test the interface by connecting to the CHEM control box via PUTTY. Press enter a few times and you should see a “>” character indicating that a command can be accepted. Commands can be typed in and then executed via pressing enter (ASCII Carriage return & Line Feed). The terminal will echo back everything you type. (You don’t need PUTTY to locally echo typed characters).

Typing “help” and then enter will return a list of commands. Some commands will have arguments to control the command. They are typed after the command (with a space in between). See the command list for details on which commands have arguments. If there is a response from the command (for example, a command to get temperature), it will always come back with carriage returns and line feeds before and after the data. This can be used for a program (such as Labview) to automatically parse the data coming back.

**Commands:**

**Command Name:** *help*

**Arguments:** None

**Description:** Lists the available commands.

**Command Name:** *EH*

**Arguments:**  (Integer) Argument should be between 0 and 7

**Description:** Enables a heater channel. Argument should be between 0 and 7. Outputs will update when a FlushDigitalIO command is issued. This command queues up the operation. The FlushDigitalIO command executes the actual IO operation.

**Example:** “EH 3”

**Command Name:** *DH*

**Arguments:**  (Integer)Argument should be between 0 and 7

**Description:** Disables a heater channel. Argument should be between 0 and 7. . Argument should be between 0 and 7. Outputs will update when a FlushDigitalIO command is issued. This command queues up the operation. The FlushDigitalIO command executes the actual IO operation.

**Example:** “DH 3”

**Command Name:** ESV

**Arguments:** (Integer) Argument should be between 0 and 11

**Description:** Enables a solenoid channel. Argument should be between 0 and 11. Outputs will update when a FlushDigitalIO command is issued. This command queues up the operation. The FlushDigitalIO command executes the actual IO operation.

**Example:** “ESV 11”

**Command Name:** DSV

**Arguments:** (Integer) Argument should be between 0 and 11

**Description:** Disables a solenoid channel. Argument should be between 0 and 11. Outputs will update when a FlushDigitalIO command is issued. This command queues up the operation. The FlushDigitalIO command executes the actual IO operation.

**Example:** “DSV 11”

**Command Name:** DAHAS

**Arguments:**  none

**Description:** Disables all heaters and solenoids. Command is immediately executed.

**Example:** “DAHAS”

**Command Name:** EMDO

**Arguments:** (Integer) Argument should be between 0 and 3

**Description:** Enables one of the misc. digital output channels. Output will update when a FlushDigitalIO command is issued

**Example:** “EMDO 0”

**Command Name:** DMDO

**Arguments:**  (Integer) Argument should be between 0 and 3

**Description:** Enables one of the misc. digital output channels. Output will update when a FlushDigitalIO command is issued

**Example:** “DMDO 0”

**Command Name:** FDIO

**Arguments:**  none

**Description:** Updates the all of the data for the digital IO channels.

**Example:** “FDIO”

**Command Name:** FON

**Arguments:** None

**Description:** Turns on the fans

**Example:** “FON”

**Command Name:** FOFF

**Arguments:** None

**Description:** Turns off the fans

**Example:** “FOFF”

**Command Name:** BUZZ

**Arguments:** (float) 0.0 to 5.0

**Description:** Enables the buzzer for a time period.

**Example:** “BUZZ 0.25”

**Command Name:** T

**Arguments:** (Integer) Argument should be between 0 and 11

**Description:** Reads a thermocouple. Returns a string in the form of “TEMP:CHANNEL:VALUE”. Example “TEMP:0:25.25” is channel 0 25.25 degrees C

**Example:** “T 0”

**Command Name:** MFCI

**Arguments:** (Integer) Argument should be between 0 and 6

**Description:** Reads an MFC Analog Input Channel. Returns a string in the form of “MFCI:CHANNEL:VALUE”. Example “MFCI:3:1.234” is channel 0 at 1.234v

**Example:** “MFCI 2”

**Command Name:** MFCO

**Arguments:** (Integer) Argument should be between 0 and 6 and the analog output value (float)

**Description:** Writes to an MFC Analog output Channel.

**Example:** “MFCO 2 1.234”

**Command Name:** AOUT

**Arguments:** (Integer) Argument should be between 0 and 3 and the analog output value (float)

**Description:** Writes to a MISC analog output

**Example:** “AOUT 2 1.234”

**Command Name:** 4TO20

**Arguments:** (Integer) Argument should be between 0 and 1

**Description:** Reads a 4 to 20mA Input Channel. Returns a string in the form of “4TO20:CHANNEL:VALUE”. Example “4TO20:1:0.014” is channel 1 at 0.014a

**Example:** “4TO20 1”

**Command Name:** AIN

**Arguments:** (Integer) Argument should be between 0 and 3

**Description:** Reads a MISC analog input channel. Returns a string in the form of “AIN:CHANNEL:VALUE”. Example “AIN:1:2.144” is channel 1 at 2.144volts

**Example:** “AIN 1”

**Command Name:** MFCON

**Arguments:** None

**Description:** Turns on power to the MFC channels

**Example:** “MFCON”

**Command Name:** MFCOFF

**Arguments:** None

**Description:** Turns off power to the MFC channels

**Example:** “MFCOFF”