ARA Smart Power System
User Manual

December 11, 2017

1 Introduction

A photo of the DAQ box component of the power distribution system, with parts labeled, is given in figure 1.

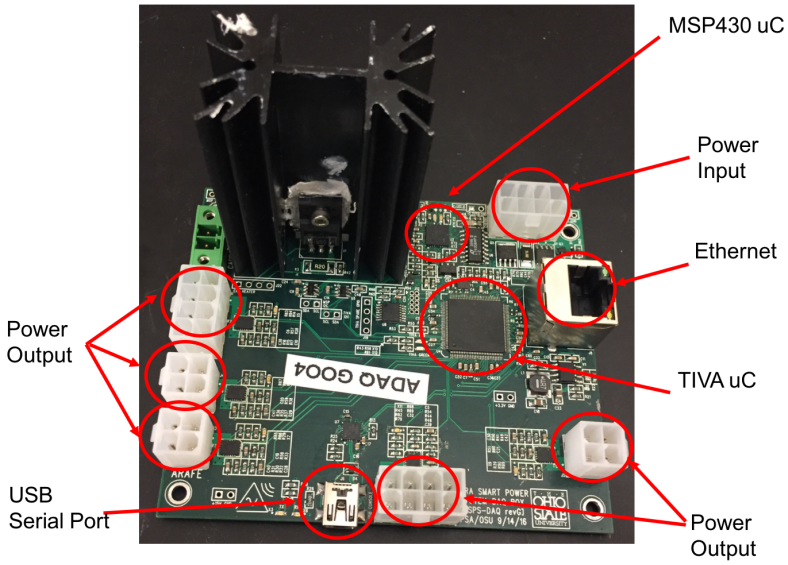

Figure 1: A picture of the ADAQ (ADAGG004) with major components listed.

2 Power Box

3 DAQ Box Web Interface

3.1 Main control page

The main ASPS-DAQ microcontroller presents a web page that allows monitoring/control of all outputs in the DAQ box, with the exception of the fiber transceiver, for obvious reasons. The fiber transceiver *can* be disabled through the USB serial command interface.

3.2 Ethernet Serial Servers

The main ASPS-DAQ microcontroller presents serial servers on several ports.

- Port 23: ASPS-DAQ Heater debug/reprogramming port
- Port 24: ASPS-Power serial port
- Port 25: SBC serial port
- Port 26: ASPS-DAQ Heater command/monitoring interface

To access any of these ports simply telnet to them. For example: `telnet 128.xxx.xxx.xxx 24`

Note, however, that many Telnet clients automatically echo normally, and so therefore some care is needed to avoid any hassles. In addition, the Console Redirection in the SBC's BIOS uses VT100-style function keys, so some configuration is needed.

At the current time, the most appropriate Telnet client for ASPS-DAQ is PuTTY. In this case, the following options need to be set:

- Under "Terminal," select "Force off" for Local echo.
- Under "Keyboard," select VT100+ for the Function keys and keypad setting.

With these settings, the complete boot process for the SBC can be viewed on port 25, and remote BIOS access is possible.

3.2.1 Resetting serial servers

Only 1 user can be connected to a serial server at a time. The status of the serial servers can be seen under the "serial.html" page (e.g. http://ip.address.here/serial.html), and a serial server which is connected to an unknown client (or a client which failed to close the connection somehow) can be forcibly disconnected.

3.3 ASPS-DAQ Heater

The heater section of ASPS-DAQ serves as a temperature watchdog. At power on, it holds off activating the remainder of the system until the temperature has reached a specified target (typically $-40\,^{\circ}\mathrm{C}$) via the use of an onboard adjustable heater. It is the only section of the ASPS-DAQ board which is required to operate to $-55\,^{\circ}\mathrm{C}$.

Communication with the ASPS-DAQ heater is done via a serial connection through the main ASPS-DAQ microcontroller, which presents it on a specified TCP port. It should be noted it is *not* possible to communicate with the ASPS-DAQ heater until the main power has been activated. Therefore, any changes to the "autonomous behavior" parameters should be done with *extreme* caution.

All communication is done via JSON packets.

There is *also* a secondary serial port interface to the main ASPS-DAQ microcontroller, used for debugging and bootstrap reprogramming. This is also presented on a specified TCP port. Entering the device into bootstrap mode is done via the Web interface at http://ip.address.here/bsl.html.

3.3.1 LED behavior

At power-on, the red LED by the heater will blink 2 times, indicating power-on cycle behavior. This is a useful thing to note if seen at any other time, because at power-on, the heater initially *disables* the main +15 V rail, which shuts down the station completely, before running through the decision tree as to whether or not to power on. (A normal reset does not cause this behavior - it is only caused by an initial power-on).

At any other time, the red LED indicates that the *PID controller is working to find the proper current.* The PID controller is extremely fast, which means that the red LED will only be on briefly once the current starts to ramp up if the heater is on.

The green LED indicates that the target temperature has been reached and everything is OK.

3.3.2 Monitoring

The ASPS-DAQ heater puts out a constant stream of monitoring data. Each JSON key contains a specific group of data.

PID data ("pid" key) This key contains an array of (in order) the setpoint, input, and output of the heater current PID loop. These values are proportional to the current flowing through the heater. The scale is roughly 1 mA (e.g. setpoint $500 \simeq 500$ mA).

Temperature data ("temps" key) This key contains an array containing only the local (microcontroller) temperature at the current moment. Remote temperature will be added in the future.

Voltage data ("volts" key) This key contains an array containing the input voltage, and the +15 V voltage. Note that these two should be close to identical if the system is on.

3.3.3 Current control

The setpoint for the heater (which returns to 0 once the system is above temperature for the wait period) can be modified via a "current" key- that is, a JSON packet with a key of "current" and a value equal to the desired setpoint.

3.3.4 Heater parameters

The "autonomous behavior" parameters can be queried and altered with the "heaterparams" key. Querying is done by sending a JSON packet with a "heaterparams" key and an empty value (or an array with less than 2 entries). The heater parameters are

Index	Description	Units	Default
0	Minimum turn-on temperature	0.01 deg C	-4000
1	Heater current when below temp	mA	500
2	Maximum wait time below temp	sec	7200
3	Wait time after target temp reached	0.5 sec	600
4	P-term in PID	128	
5	I-term in PID	64	
6	D-term in PID	0	

Altering these values is done by sending a JSON packet with a key of "heaterparams", and an array containing the index to alter (0-6), followed by the new value. Note that changing the P, I, or D terms could result in very bad behavior, and these changes are preserved over power cycles!

4 DAQ Box USB Interface

In addition to the web interface, the ASPS-DAQ also includes a USB interface, and can be accessed through the mini-USB port as labeled in figure 1. It supports 38400/8/N/1 (which is a high baud rate for ARA boards).

One can connect to it with an serial port client, such as TeraTerm, PuTTY, etc. On a Linux machine, a common way to so would be to use screen, e.g. `sudo screen /dev/ttyUSB0 38400`.

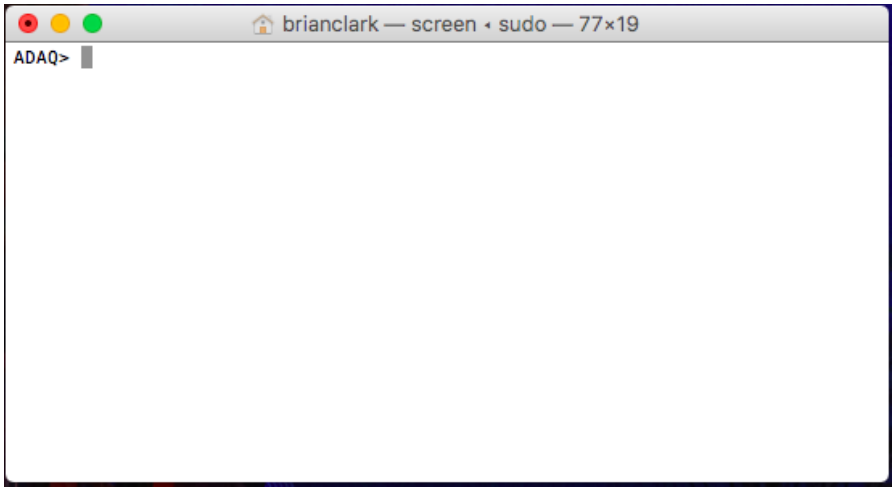

Figure 2: A picture of the ADAQ USB interface in a terminal.

After logging in, you will be presented with a user interface as in figure 2, with the prompt `ADAQ>`. At this point you can enter a variety of commands and the ASPS-DAQ will respond with information.

The commands available are, in alphabetical order:

- `bridge`
  - This connects the main serial console to one of the four serial servers hosted by the ASPS-DAQ.
  - Execute by running "`bridge x`" where x = 0 for heater monitor, 1 for ASPS-power, 2 for SBC, 3 for heater command interface.
  - Example: to connect to the ASPS power serial interface, type `bridge 1`.
  - To exit, enter +++ (three "plus" symbols). Interface will close and return you to the `ADAQ>` prompt. No return character (like `\n` etc.) needed.
- `bsl`
  - This will reboot either the heater microcontroller or the ASPS-PWR microcontroller in bootstrap-loader (BSL) mode.
  - Execute by running "`bsl x y`" where x = 0 or 1 for do not/do put the heater into BSL mode, and y= 0 or 1 for do not/ do put the ASPS-PWR uC into BSL mode.
  - Example: to rebot the ASPS-PWR microcontroller , but not the heater uC microcontroller, into BSL mode, type `bsl 0 1`.
- `control`
  - This will turn off the various outputs to the board.
  - Execute by running "`control x y`" where x = 0, 1, 2, 3, or 4 to select the output to control, and y = 0 or 1 for "off" and "on" respectively.
    - 0 corresponds to AUX2 (typically unused)
    - 1 corresponds to the DSPM
    - 2 corresponds to AUX1 (typically the IceCalDuo)
    - 3 corresponds to ARAFE
    - 4 corresponds to the Fiber Switch
  - Example: to reboot the ASPS-PWR microcontroller , but not the heater microcontroller , into BSL mode, type `bsl 0 1`.
- `getmac`
  - This will report the mac address of the TIVA microcontroller on the ASPS-DAQ.
  - Execute by running "`getmac`" (no arguments).
- `loop`
  - This launches the TIVA microcontroller into an artificial infinite loop. It is a debugging tool used to test whether or not the watchdog is able to catch a runaway process.

  - Execute by running "`loop`" (no arguments).
  - Should be followed by a watchdog reset signal.

- `identify`
  - Prints the ASPS-DAQ board id to the screen.
  - Execute by running "`identify`" (no arguments).
  - Name of the board should print to screen, e.g. `ADAQF003`. "F" designates the revision of the board (revF) and 003 designates the individual serial number within that revision series. There are only two active revision series of the ASPS-DAQ, revF and revG.

- `ip`
  - Requests the ASPS-DAQ to print its IP address to the console.
  - Execute by running "`ip`" (no arguments).
  - If the board has a mac address, and it is connected to the internet by a local DHCP server, it will report its IP address.

- `reboot`
  - Reboots the ASPS-DAQ TIVA microcontroller.
  - Execute by running "`reboot`" (no arguments).

- `savemac`
  - **WARNING WARNING WARNING, READ ME BEFORE USING THIS COMMAND: This command can (basically) only be run once in the lifetime of the TIVA microcontroller!! Do not use unless you know what you're doing!**
  - Stores the mac address entered with `setmac` to permanent memory.
  - Execute by running "`savemac`" (no arguments).

- `setid`
  - Sets the id of the ASPS-DAQ microcontroller. This is the id that is reported by the command `identify`.
  - Execute by running "`setid ADAQ`α`###`", where α is the board revision letter (e.g. A, B, etc.) and ### is the three-digit serial number of the board within that revision line. It will only take eight alpha-numeric characters as an argument.
  - Example: to set a microcontrollers id to ADAQG006 type `setid ADAQG006`.

- `setmac`
  - Sets the mac address of the ASPS-DAQ microcontroller. This is the mac address that is reported by the command `getmac` and saved by the command `savemac`.

- Execute by running "`setmac ## ## ## ## ## ##`", ## are the bytes of the mac address, which are typically visualized as ##:##:##:##:##:##.
- Example: to set a microcontrollers mac address to FC:C2:3D:0D:A8:7C type `setmac FC C2 3D 0D A8 7C`.
- This command must be followed by `savemac` to take effect permanently. HOWEVER, YOU SHOULD READ THE SECTION ON `savemac` BEFORE PROCEEDING!!!

- `status`
  - Reports various housekeeping information on the board (e.g., temperatures and the like), print out like figure 3.
  - Execute by running "`status`" (no arguments).
  - The various components are:
    * The currents being consumed by the four outputs (same labeling as for command `control`, so see there for what each means.
    * What channels are currently on (meaning they are being allowed to supply current).
    * What channels are currently in fault. This means they were either turned off with the `control` command, or they tripped their electronic fuses by drawing too much power.
    * The temperature of three on board sensors. 0 is the temperature reported by the thermometer internal to the TIVA uC, 1 is the temperature of the die of one of the onboard MIC280s, and should be taken as a proxy for the board temperature, and 3 is the temperature of the thermistor taped to the wall of the DAQ box. In this example in figure 3, that thermistor is disconnected and therefore reporting the error value of 127.

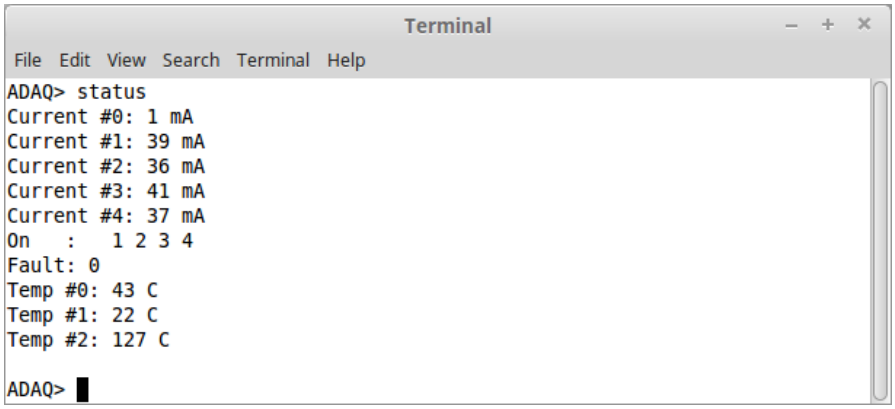

Figure 3: A picture of the ADAQ USB interface in a terminal.