Beat Kamland-Zen to it!
Find majorana neutrinos now!

Penn Daq

FOR DUMMIES®

2nd Edition

Debug electronics and test data flow with ease

A Reference for the Rest of Us!®

FREE eTips at dummies.com®

General Pez, CISSP
Information Security Consultant

Foreword by Stuart McClure, President/CTO, Foundstone, Inc.

The Official Penn DAQ Users Guide

Richie Bonventre

All questions should be sent to rbonv@hep.upenn.edu. Bugs or feature requests can be posted on github at

`http://www.github.com/pennsnoplus/penn_daq`

Contents

1 Installation and configuration **4**
- 1.1 Dependencies . . . 4
- 1.2 Installing debugDB . . . 4
- 1.3 Installing penn_daq . . . 4
- 1.4 Networking . . . 4
- 1.5 Configuration . . . 5

2 Running Penn DAQ **6**

3 List of commands **7**
- 3.1 DAQ commands . . . 7
- 3.2 XL3 commands . . . 7
- 3.3 FEC commands . . . 9
- 3.4 MTC commands . . . 9
- 3.5 Tests . . . 11

4 Using debug DB **15**
- 4.1 The Main Page . . . 15
- 4.2 Quick Links . . . 15
  - 4.2.1 Recent Tests . . . 16
  - 4.2.2 Documentation . . . 16
- 4.3 Test Pages . . . 16
- 4.4 Final Test . . . 16
- 4.5 CrateView . . . 17

5 XL3 setup **17**

6 SBC setup **17**

1 Installation and configuration

1.1 Dependencies

Penn_daq has been designed to be platform independent and to have as few dependencies as possible. At the moment, the only thing actually required is the curl library in order to communicate with the database. If you want to host the debug database locally, you will also need couchdb, and kansojs.

1.2 Installing debugDB

If you are only going to be using the database on snotpenn01, you can skip this section. First, clone the debugdb repository:

```
git clone git://github.com/mastbaum/debugdb.git
```

Then, from within the debugdb directory,

```
kanso push <database name>
kanso pushdata <database name> daq_docs.json
```

"debugdb" is recommended as the database name.

Check that the installation was successful by visiting the URL returned by kanso push.

1.3 Installing penn_daq

You can get the most up to date version of the code by cloning the penn_daq git repository. If you aren't going to be committing any changes to the code, you can get a read-only copy with the command

```
git clone git://github.com/pennsnoplus/penn_daq.git
```

If you are going to need commit privileges, you can either fork your own copy (see github documentation), or contact us to get your account added to the pennsnoplus organization. You then should create a local ssh key, and in your account settings for github, add the public key you've created. You can now check out a copy of the repository with the command

```
git clone git@github.com:pennsnoplus/penn_daq.git
```

Once you have your penn_daq directory cloned, move there and type "make". That should be it. If there are any error messages or if it fails to compile, please submit a bug report on github.

1.4 Networking

By default the XL3 expects to be on the 10.0.0.x network. It will try to connect to the DAQ at 10.0.0.31. Each XL3's ip address will be 10.0.0.(crate number 1-19) and it will connect to the DAQ on port 44600+(crate number 1-19). The SBC's address and port and the ports for viewers and controllers can all be changed in the penn_daq configuration file.

1.5 Configuration

The last step before running the code is to set up the configuration file. This file is located in the data directory and is called, somewhat redundantly, "config.cfg". That folder also has several other ".cfg" files with default settings for various places and occasions; you should copy over whichever one suits your need.

DB_ADDRESS If you are running a local database, this should be set to "localhost". For snotpenn01, set it to "snotpenn01.snolab.ca"

DB_PORT By default this is 5984. Only change this if you changed the port for your local database

DB_USER For a local database in admin party mode, leave this blank. For snotpenn01, set to "snoplus"

DB_PASSWORD For admin party, leave blank. For snotpenn01, set to default snoplus password

DB_BASE_NAME Leave as default value

DB_VIEW_DOC Leave as default value

XL3_PORT Unless you configure the ml403s differently, this should be set to 44601

NEED_TO_SWAP If you are not running on a big endian machine, this should be left at 1

SBC_PORT Can be anything besides (XL3_PORT - XL3_PORT+20) and should be different than CONT_PORT and VIEW_PORT

SBC_USER Leave as default value

SBC_SERVER Should be set to the ip address given to the SBC

MTC_XILINX_LOCATION Leave as default value

DEFAULT_SSHKEY If your sbc user has a password, you can create a local ssh key and give it to the sbc. This should point at the location of the ssh key relative to the penn daq directory. If the sbc has no password, you can leave this blank

CONT_PORT Can be anything besides (XL3_PORT - XL3_PORT+20) and should be different than SBC_PORT and VIEW_PORT

CONT_CMD_ACK Leave as default value

CONT_CMD_BSY Leave as default value

VIEW_PORT Can be anything besides (XL3_PORT - XL3_PORT+20) and should be different than SBC_PORT andCONT_PORT

CURRENT_LOCATION Can be set to "penn", "ag", or "ug" to set the location that will be recorded in the debug database

BUNDLE_PRINT Sets frequency it will print readout speed to screen. It will print out after this many packets.

MAX_PENDING_CONS Leave as default value

2 Running Penn DAQ

For this section we will assume that you already have your SBC and XL3 both set up and running. If you need help getting these set up, please check the relevant sections of the documentation (note that they don't need to be started up in any particular order, you can run penn_daq first and start the XL3's after).

Once everything else is running, you are ready to start up penn_daq. First, run (penn_daq directory)/bin/penn_daq. If everything is set up correctly, it should tell you that it connected to the database with return code 200. If you get a 404 or 403 or any other error code, then check that the database is running and that you have the configuration all set up correctly. Next, if the XL3 is running and you are connected to the xl3 network with the correct IP, you should see the e3 connect to penn_daq. Now that you are connected, you should start a control terminal to start issuing commands to the XL3. The control terminal can be started by running (penn_daq directory)/bin/tut. You can also issues commands by telnetting to the daq on the port specified in the configuration document. You should see a notice once you connect on the main penn_daq window that a controller was connected. Finally, to connect to the sbc, from tut enter the command "sbc_control". If you have the address of the SBC correct, you should see it remotely start up OrcaReadout and connect.

Penn DAQ is now a multi threaded program. When you issue it a command, it checks which of its connections are available, and any connections that will be used by that command are locked until it finishes. If you try running a second test on the same connection, the tut window will flash "connection busy" to let you know. Otherwise, you can have penn_daq running multiple things at once, so for example you can initialize the mtc and the fecs at the same time.

Results of all SNO+ electronics testing through penn_daq are automatically logged to a CouchDB database. debugdb is a CouchDB application written with the kanso framework to provide a user interface to the test data.

3 List of commands

3.1 DAQ commands

help Gives a list of all the commands recognized by penn_daq. Can also list the various register address maps.

print_connected Prints out a list of the connected devices. Note that the DAQ does not know when an XL3 disconnects, so it will continue to show the XL3 connected unless you manually quit it from the DAQ

start_logging Logs all the output to a file

stop_logging Stops logging output to a file. By default logging is off.

set_location Tells the DAQ whether it is being run at Penn, or above ground or underground at site. Can also be set in configuration file

sbc_control Starts or stops the readout code on the SBC and connects it to the daq

- **-r** (default) Kill OrcaReadout and then restart it before connecting
- **-c** Start OrcaReadout and connect to it
- **-k** Kill OrcaReadout

clear_screen Clears the main terminal of output

reset_speed Zeroes the average speed calculated for pedestal runs

run_macro Run a list of commands from a file in the macro folder

- **-f (filename)** Location of the file relative to the macro folder

stop_macro Stops a running macro from issuing any more commands

3.2 XL3 commands

xr Read from an xl3 register specified by number

- **-r (int)** Register number
- **-c (int)** Crate number

xw Write to an xl3 register specified by number

- **-r (int)** Register number
- **-c (int)** Crate number
- **-d (hex)** Data

xl3_rw Read/write to an xl3/fec register by address

- **-c (int)** Crate number
- **-a (hex)** Address
- **-d (hex)** Data

xl3_queue_rw Add an xl3/fec register read/write to the xl3 command queue

- **-c (int)** Crate number
- **-a (hex)** Address
- **-d (hex)** Data

debugging_on Turn debugging output to serial port on for an xl3

- **-c (int)** Crate number

debugging_off Turn debugging output to serial port off for an xl3

- **-c (int)** Crate number

change_mode Turn xl3 readout on or off

- **-c (int)** Crate number
- **-n** Normal mode - readout is on
- **-i** Init mode - readout is off
- **-s (hex)** Data available slot mask for readout

sm_reset Reset the vhdl state machine on an xl3

- **-c (int)** Crate number

read_local_voltage Read the voltages on an xl3

- **-c (int)** Crate number
- **-v (int)** Voltage select FIXME

hv_readback Read out HV voltage and current

- **-c (int)** Crate number
- **-a** Show supply A
- **-b** Show supply B

crate_init Initialize the xl3 and fec and load fec xilinx

- **-c (int)** Crate number

-s (hex) Slot mask

-x Load normal xilinx code

-X Load caldac xilinx code

-v Reset HV setpoints

-b Use most recent cbal values

-d Use most recent zdisc values

-t Use most recent ttot values

-a Use cbal,zdisc,ttot values

-w Use crate/slot specific ECAL/ECA values

3.3 FEC commands

fr Read from a fec register specified by number

-r (int) Register number

-c (int) Crate number

fw Write to a fec register specified by number

-r (int) Register number

-c (int) Crate number

-d (hex) Data

read_bundle Read a single bundle from a specified fec

-c (int) Crate number

-s (int) Slot number

-q Quiet output

3.4 MTC commands

mr Read from a mtc register specified by number

-r (int) Register number

mw Write to a mtc register specified by number

-r (int) Register number

-d (hex) Data

mtc_read Read from a mtc register specified by address

-a (hex) Address

mtc_write Write to a mtc register specified by address

-a (hex) Address

-d (hex) Data

mtc_init Initialize the mtcd and mtca

-x Load xilinx

set_mtca_thresholds Load mtca threshold dacs

-(00-13) (float) Load dac (00-13) to x volts

set_gt_mask Set which triggers are masked in

-m (hex) Mask of triggers

set_gt_crate_mask Set which crates are masked in to the global trigger

-m (hex) Crate mask

set_ped_crate_mask Set which crates are masked in to get pedestals

-m (hex) Crate mask

enable_pulser

disable_pulser

enable_pedestal

disable_pedestal

set_pulser_freq Change the frequency of the pulser

-f (float) Frequency in Hz. Set to 0 to enable software triggers

send_softgt Send one software global trigger

multi_softgt Send multiple software global triggers

-n (int) How many to send

3.5 Tests

run_pedestals Fully set up a pedestal run. Sets up mtcd, turns on pedestals and pulser, sets crate masks, then enables pedestals on crates and switches them to readout mode.

-c (hex) Crate mask

-s (hex) Slot mask for all crates

-(00-19) Slot mask for crate (00-19)

-p (hex) Channel mask for all crates, all slots

-f (float) Pulser frequency in Hz. Set to 0 to enable software triggers

-t (int) Global trigger delay

-w (int) Pedestal width

run_pedestals_mtc Sets up the mtcd to run pedestals. Turns on pedestals and pulsers, but does not set crate masks.

-f (float) Pulser frequency in Hz. Set to 0 to enable software triggers

-t (int) Global trigger delay

-w (int) Pedestal width

run_pedestals_crate Sets up crates to run pedestals. Enables pedestals on channels and switches xl3s to readout mode

-c (hex) Crate mask

-s (hex) Slot mask for all crates

-(00-19) Slot mask for crate (00-19)

-p (hex) Channel mask for all crates, all slots

end_pedestals Fully stops a pedestal run. Turns off pedestals and pulser and disables pedestals on fecs, then turns off readout on the xl3s.

-c (hex) Crate mask

end_pedestals_mtc Stops a pedestal run on the mtcd. Turns off pedestal and pulser

end_pedestals_crate Stops a pedestal run on crates. Disables pedestals on fecs, then turns off readout on the xl3s.

-c (hex) Crate mask

trigger_scan Scans mtcas across threshold and nhit

-t (int) Trigger to mask in on mtcd

-c (hex) Crate mask to mask in to gt

-s (hex) Slot mask for all crates

-(00-19) Slot mask for crates (00-19)

-m (int) Minimum threshold in dac counts to scan down to

-d (int) Which dac to program thresholds on mtca. By default will be the same as the trigger set by -t.

-n (int) Total nhit to scan up to

-q Enable quick mode (only scans every 10th nhit)

-f (filename) File to print results to

fec_test Does read/write test of xl3/fec/cmos registers

-c (int) Crate number

-s (hex) Slot mask

-d Write results to database

mem_test Does a read/write test of fec fifo and tests all address bits

-c (int) Crate number

-s (int) Slot number

-d Write results to database

vmon Reads voltages on fec

-c (int) Crate number

-s (hex) Slot mask

-d Write results to database

board_id Reads out ids of motherboard, daughterboards, and hv card

-c (int) Crate number

-s (hex) Slot mask

-d Write results to database

ped_run Run pedestals and check bundles

-c (int) Crate number

-s (hex) Slot mask

-p (hex) Channel mask

-n (int) Number of pedestals per cell

-l (int) Lower limit of charge before channel is flagged

-u (int) Upper limit of charge before channel is flagged

-f (float) Pulser frequency in Hz. Set to 0 to use software global triggers

-t (int) Global trigger delay

-w (int) Pedestal width

-b Mark as a balanced run in database

-d Write results to database

zdisc Zero discriminators

-c (int) Crate number

-s (hex) Slot mask

-r (float) Target rate

-o (int) Offset

-d Write results to database

crate_cbal Balance short and long integration charges

-c (int) Crate number

-s (hex) Slot mask

-p (hex) Channel mask

-d Write results to database

cgt_test Test global trigger counters and rollover

-c (int) Crate number

-s (hex) Slot mask

-p (hex) Channel mask

-d Write results to database

cmos_m_gtvalid Set delays to get correct gt valid window

-c (int) Crate number

-s (hex) Slot mask

-p (hex) Channel mask

-g (float) Global trigger cutoff

-n Do not adjust twiddle bits

-d Write results to database

cald_test Test the fec adcs using the calibration dac

- **-c (int)** Crate number
- **-s (hex)** Slot mask
- **-l (int)** Lowest dac count to start at
- **-u (int)** Highest dac count to scan to
- **-n (int)** Number of points to check between upper and lower limit
- **-s (int)** Number of times to sample each point
- **-d** Write results to database

get_ttot Measure the trigger delay time FIXME

- **-c (int)** Crate number
- **-s (hex)** Slot mask
- **-t (int)** Target time
- **-d** Write results to database

set_ttot Change rmp and vsi dacs to get ttot right

- **-c (int)** Crate number
- **-s (hex)** Slot mask
- **-t (int)** Target time
- **-d** Write results to database

fifo_test Check that the fifo fills correctly and behaves as expected on overflow

- **-c (int)** Crate number
- **-s (hex)** Slot mask
- **-d** Write results to database

disc_check Checks that the discriminator is working and stable

- **-c (int)** Crate number
- **-s (hex)** Slot mask
- **-n (int)** Number of pedestals to fire
- **-d** Write results to database

mb_stability_test Checks that triggers are stable and none are skipped or corrupt

- **-c (int)** Crate number

-s (hex) Slot mask

-n (int) Number of pedestals to fire

-d Write results to database

chinj_scan Injects charge and does pedestal runs

-c (int) Crate number

-s (hex) Slot mask

-p (hex) Channel mask

-w (int) Pedestal width

-t (int) Global trigger delay

-n (int) Number of pedestals to fire per cell

-l (int) Chinj dac lowest dac count to start at

-u (int) Chinj dac highest dac count to stop at

-a (int) Which adc to inject charge to (0 = qhs, 1= qhl, 2 = qlx, 3 = tac)

-f (float) Pulser frequency in Hz

-e Enable pedestals

-d Write results to database

final_test Run a full suite of tests

-c (int) Crate number

-s (hex) Slot mask

4 Using debug DB

4.1 The Main Page

The main page features three panes:
— Quick Links — Recent Tests — Documentation —

4.2 Quick Links

- CrateView: See below
- All FECs: List of all FECs that have been tested, sorted by FEC ID
- All Daughterboards: List of all DBs that have been tests, sorted by DB ID
- All Crates: List of all crates that have been tests, sorted by crate ID
- All Tests: List of all test types that have been run, sorted by test name. Useful for, e.g., seeing all the fec_tests.

4.2.1 Recent Tests

A quick summary of the most recent tests. Click "view all" to see the entire history sorted by date.

4.2.2 Documentation

Links to some DAQ-related documentation

4.3 Test Pages

Each type of test has a page designed to make results readable quickly and easily. All test pages feature:

- The test name (e.g. "Final Test")
- Whether the test passed or failed
- The date the test was run
- The configuration in which the test was run:
- Location (Penn, surface, or underground)
- Crate ID
- Slot
- FEC ID
- Daughterboard IDs
- Additional information varies from test to test.

4.4 Final Test

Final tests may have any number of tests associated with them. The final test summary page includes a form for comments on the FEC and daughterboards, dark matter measurements, DC offsets, etc., and checkboxes indicating whether the board has been refurbished or cleaned. To save changes, click "Save" at the top of the form.

Under "Tests," the tests associated with this final test are listed along with their pass/fail status. Click on any test name for that test's details.

4.5 CrateView

CrateView provides a quick view of the most recent test in each slot of each crate. For example, to see the most recent tests in crate 12, click "12" on the left side of the table. Click a test name for test details (or, as usual, a FEC ID or DB ID for all tests run on that FEC/DB).

Below the most recent test data for each slot in CrateView is a link titled "History." This will show all tests run in this crate/slot.

5 XL3 setup

6 SBC setup