People on call

Don’t hesitate to call if worried about safety of yourself or the detector.
Jordan’s cell: 646 275 9949.

Table of contents

- People on call
- Table of contents
- Messages of the day
- Shifter responsibilities
- DCS (temperatures, dewpoints, voltages, currents)
- Watching for power supply trips
- The PC
- Terminal with tabs you need
- My module name shorthand
- Modules not connected or otherwise not in detconfig
- Modules to test with DTB
- Cabling map and script that produces the POS configs
- The slots in the uTCA crate and the fiber bundles
- Post-powercycle of the uTCA crate
- Switching to a new disk
- Power off a disk at CAEN
- Switch out the green power cables
- Getting the right pixel FEC cables
- Change the configuration to a new disk
- Turn on a disk
- Test CCU ↔ portcard communication
- Some POS reminders
- Initial configure with POS after detector power up
- Analyzing VcThrCalDel
- Running a calibration with POS
- What good FED phases look like
- TBM PLL scan
- PixelAlive for checking the state of the readout
- SCurve runs + analysis (current shifts)
- BumpBondTest runs + analysis

Messages of the day

- 8/24: instead of copying the data to cms-sidet001 to run the check scripts, just run them on the local computer. (The SCurve and BumpBondTest runs sections have been updated following this instruction.)

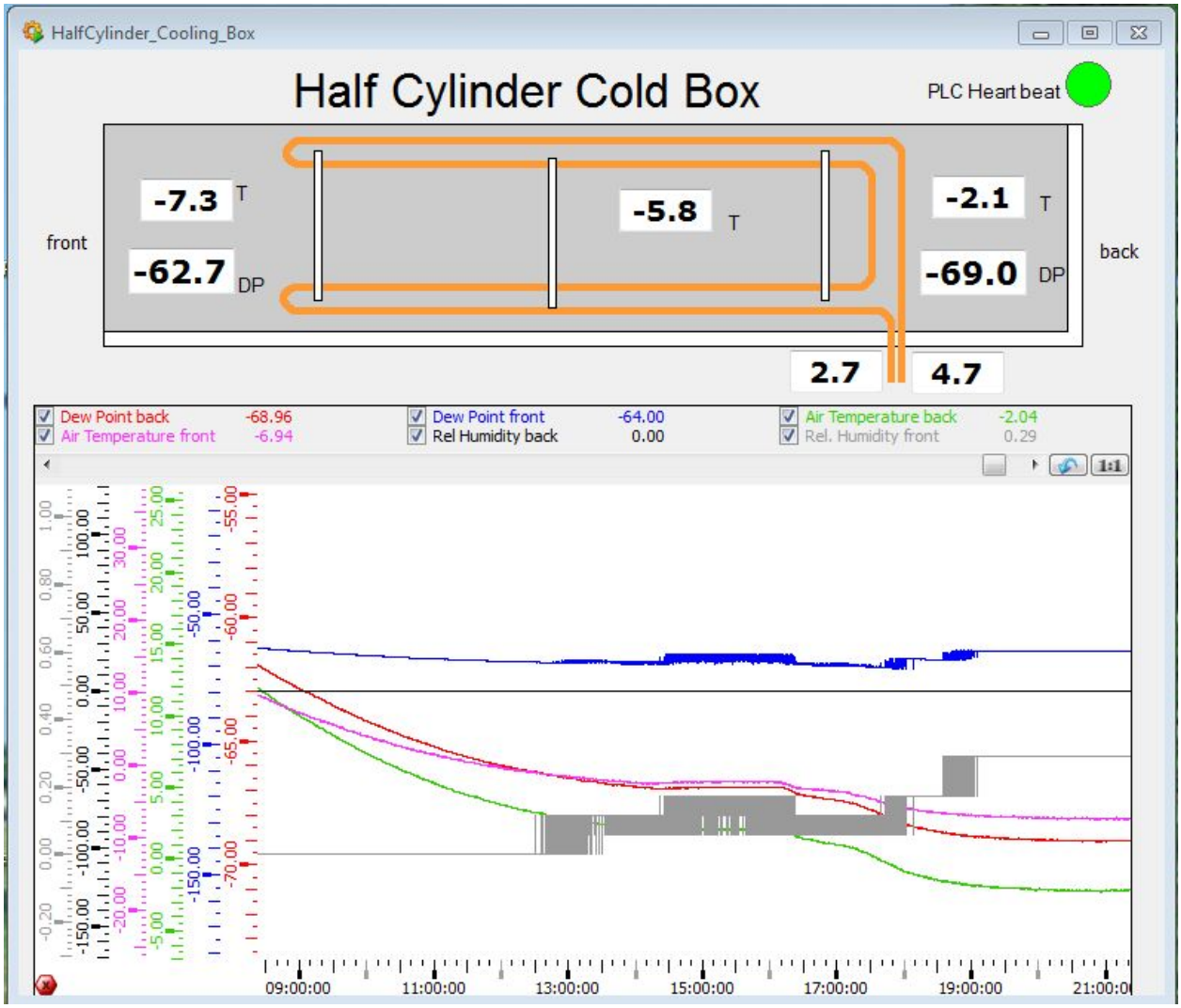
Shifter responsibilities

- Safety, especially for night shifts. Did you do the CO2 training?
- Monitor the relative humidity in the box: it should stay ~0%.
- Watch for power supply trips.
- Coming in, understand from the previous shifter the shift plan in terms of calibration runs needed, how you should monitor the data, etc., and read the messages of the day.
- Leaving, inform the next shifter of the activities and events of the shift, for which you should also have kept a good elog.

DCS (temperatures, dewpoints, voltages, currents)

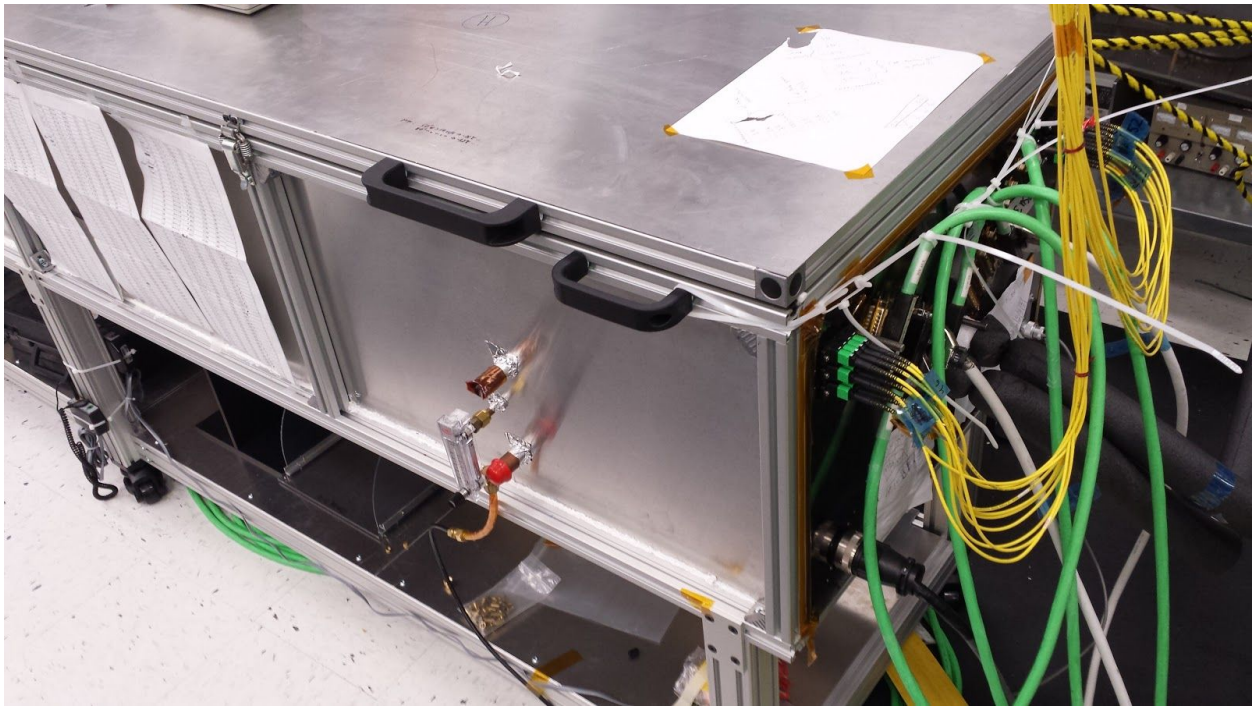
The DCS system running on the Windows PC near the CO2 system monitors temperatures in the coldbox and shuts off the detector power if one of the interlock conditions is met: too little margin between dewpoint and temperature, module or HC temperature too high. One thing we don't have a good automatic mechanism on is whether the dry nitrogen purge has failed, which is what keeps the humidity and dewpoint down. So one of the most important shifter responsibilities is to keep an eye on the humidity and dewpoint measurements shown on the DCS screen.

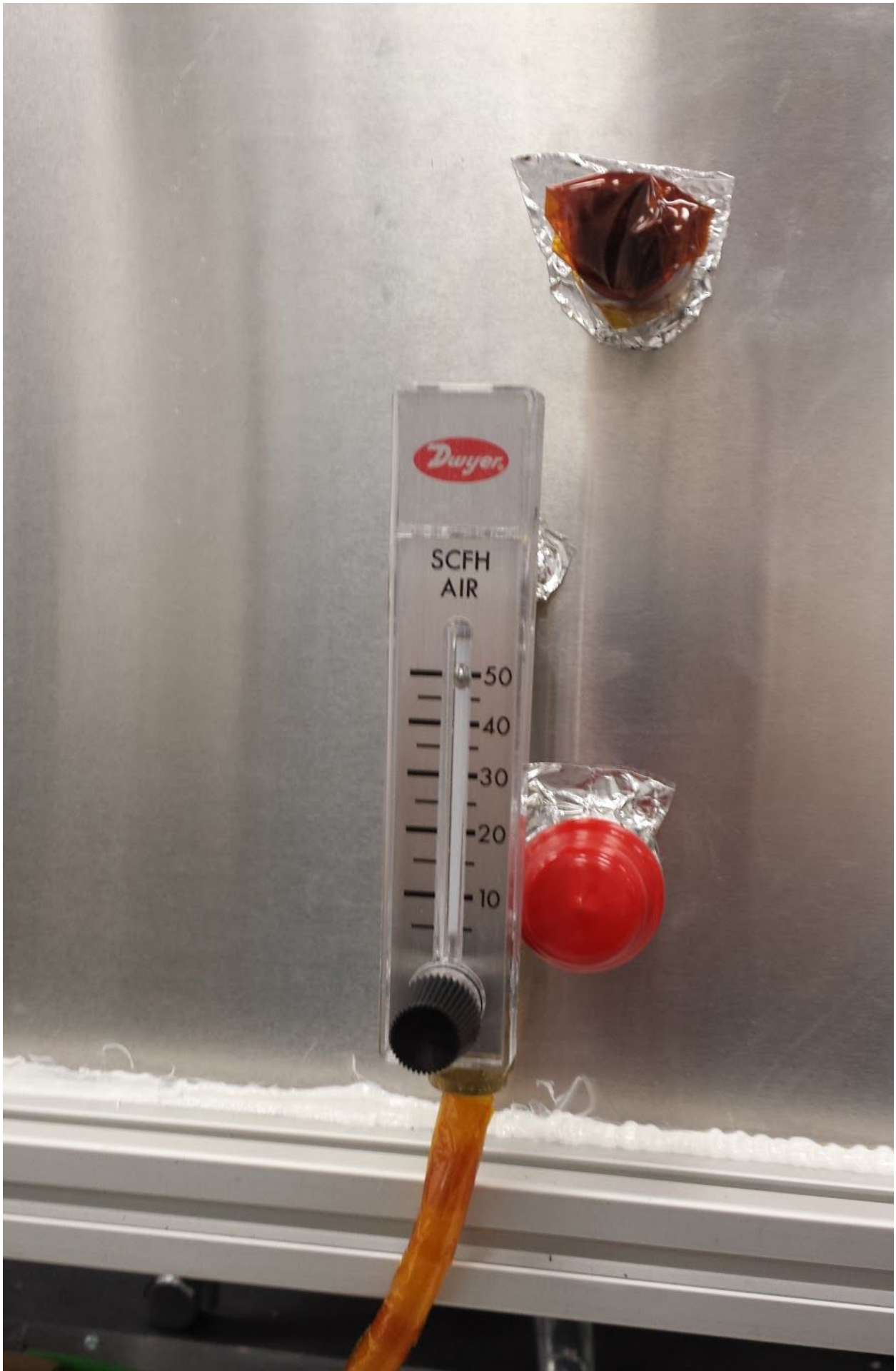
- If the screen is blank, jiggle the mouse to wake it up (don't click the mouse or push a keyboard button).
- Look for this window:



The boxes labeled “DP” are the dewpoints at the front and back of the box, calculated from the air temperature and relative humidity. (Dewpoint = the temperature and humidity at which moisture condenses out of the air on surfaces as dew.) The relative humidities (black and grey plotted curves and text above the plot) should stay very near 0%.

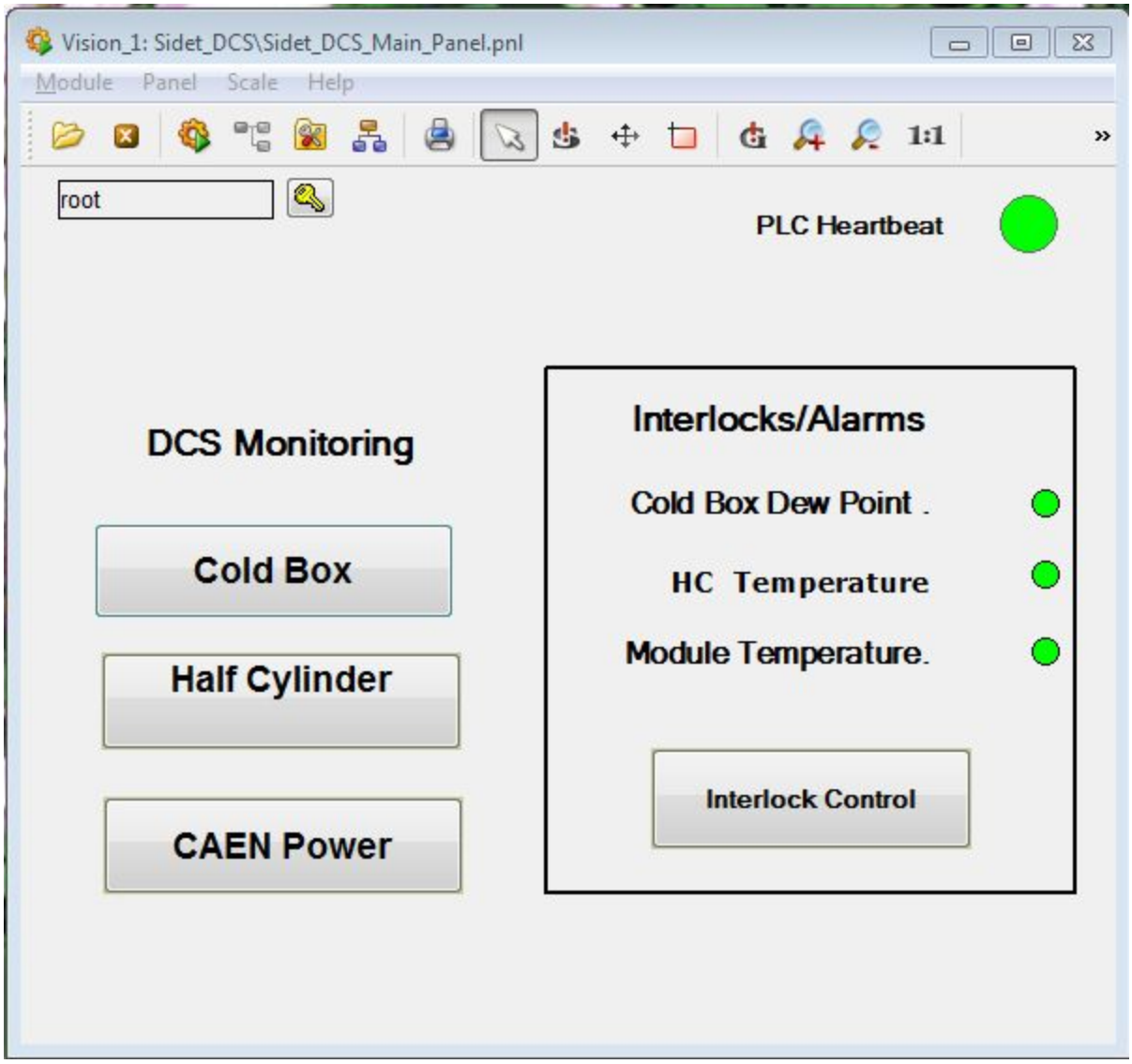
- If the humidity and dewpoint are rising, check the nitrogen purge inlet and gauge, which is at the end of the coldbox nearest the PC:



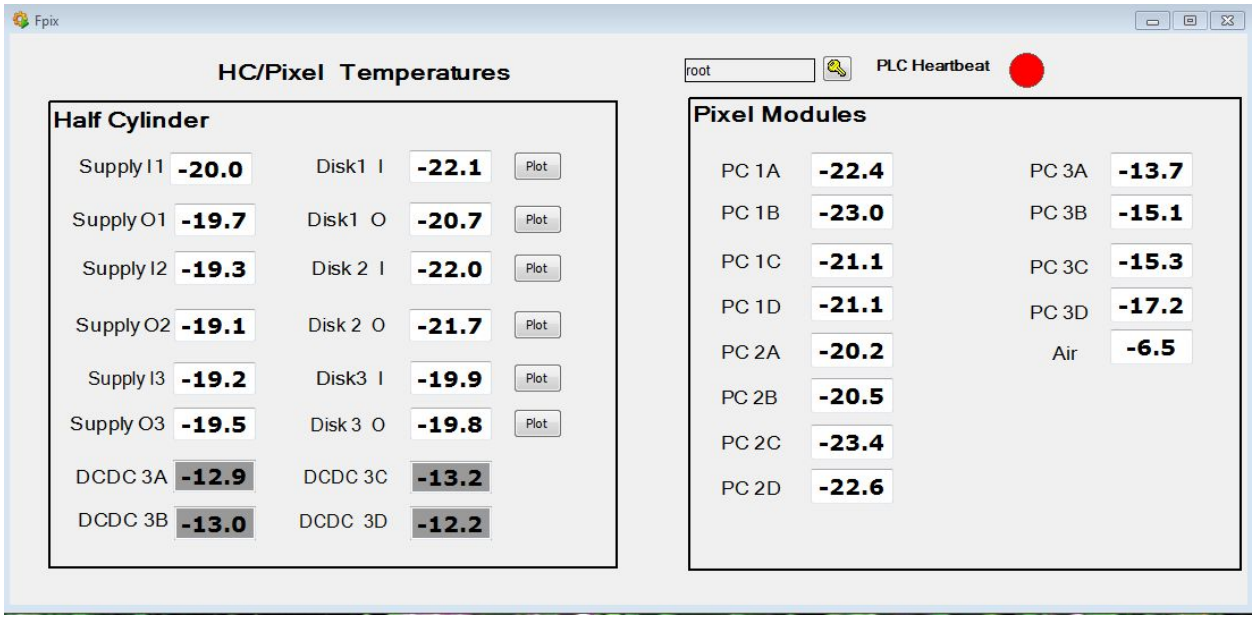

Dwyer
SCFH
AIR
50
40
30
20
10

The little ball should register 50 SCFH of flow.

- If the purge is not working or the humidity is rising, call for help.
- If you do not see the DCS window above, try maximizing things from the toolbar, or if the window has been closed, you can reopen it by clicking on “Cold Box” in this window:



- The other important thing to monitor here are the temperatures, as shown by this window:

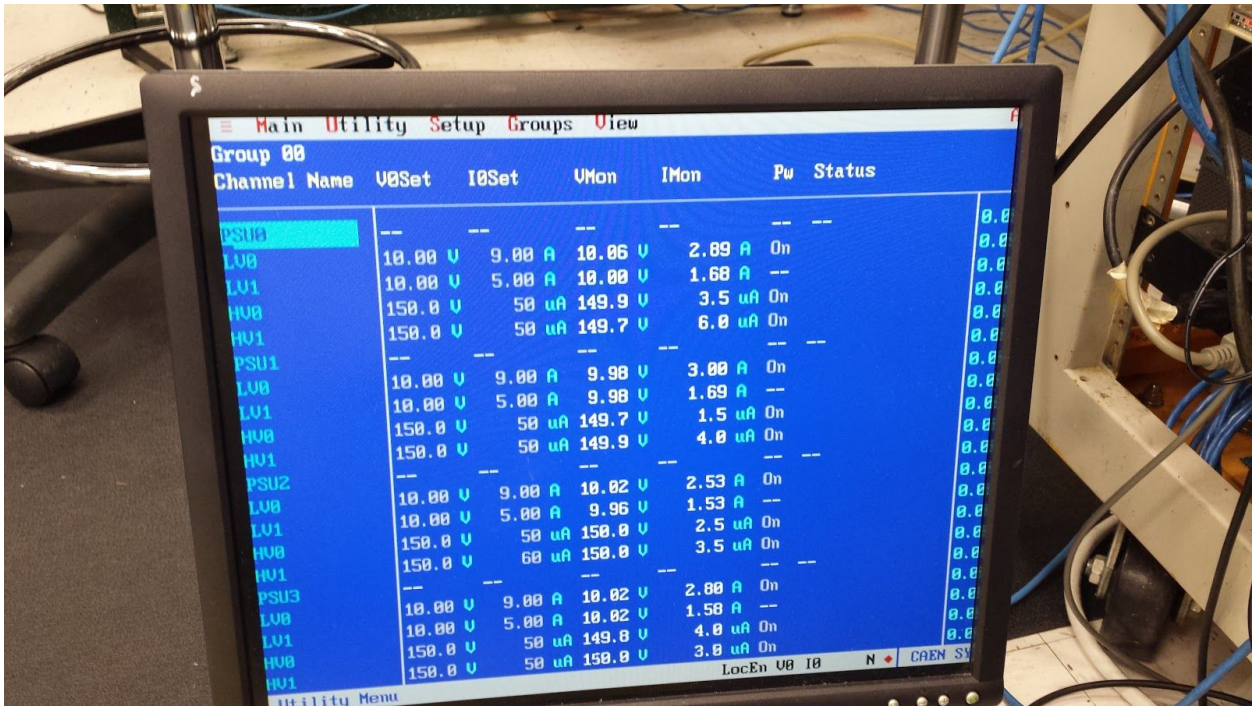


(It can be re-opened by clicking on “Half Cylinder” from the main panel, as above.)

- The temperatures of the supply and disks should be about -20degC, as should the modules you are not powering. In the example above, the modules on disk 3 are powered, which is why they are about 5-6 degrees warmer.

Watching for power supply trips

Every once in awhile, study the CAEN monitor:



If there has been a trip, where the current supplied (“IMon”) exceeded the limit (“I0Set”), or other power supply problem, the channel will be off and there will be a red message in the “Status” column. Data that is taken with power off will be considered bad, so it is best to try to fix power supply problems as they come up--call someone on the on-call list at the top of this document.

The PC

The hostname is dellfromdoug, but it is currently not available to be ssh’d into. The username is fnaltest. The password is on a sticky note on the monitor.

Terminal with tabs you need

There should already be a full screen terminal with ~11 tabs open. If not, running `jmtterms.sh` on the desktop will relaunch them, setting up the environments needed.

My module name shorthand

I am lazy, so below you might see shorthand like this:

- BmI,3,4,1,2 means BmI_D3_BLD4_PNL1_RNG2.
- 3,4,1,2 means D3_BLD4_PNL1_RNG2. The HC must be clear from context.
- 4,1,2 means BLD4_PNL1_RNG2. The HC and disk must be clear from context.

Modules not connected or otherwise not in detconfig

BpO = HC4 (current):
1,3,1,2 (bad cable routing) and 3,8,2,1 (bad HV current draw).
All 14 modules behind D2_PRT1.

BmI = HC1:
Five modules on each disk are not connected, two on each of portcards B and D, and one on portcard C. They are all on the top of the portcard stack. For portcard B, they are on ports 2 & 6, C just port 6, and D ports 2 & 4. In shorthand notation, the module names are 4,2,1 5,2,1 6,2,1 10,2,1 11,2,1 on each disk.

Modules to test with DTB

Any in the list of not-connected modules above, and
BpO:

- No extra yet

BmI:

- 3,3,1,2: has issue on roc13 that I think is just a poor calibration of rocdelay
- 3,4,1,2: has timing issue

Cabling map and script that produces the POS configs

The master map is curated by Hannsjörg. My map is currently only different in that it is sorted by disk / portcard / portcard connection, and only includes BmI for now.

I use `~/TriDAS/pixel/jmt/write_other_hc_configs.py` to do advanced queries. The "cable map" terminal tab usually has an interactive python shell open, if not you start it with

```
python -i $JMT/write_other_hc_configs.py disk
```

where `disk` should be `3` for disk 3, etc. Then you can do things like:

```
>>> pprint(countem([(m.fec, m.mfec, m.mfecchannel) for m in d.modules if
(m.bld,m.pnl,m.rng) in
[(10,1,2),(14,1,2),(16,1,2),(3,1,1),(3,1,2),(7,2,2),(9,2,1)]]))
{(9, 1, 1): 4, (9, 1, 2): 2,
 (9, 2, 1): 2, (9, 2, 2): 1,
 (10, 1, 1): 2, (10, 1, 2): 6,
 (10, 2, 1): 1, (10, 2, 2): 3}
```

`$JMT/write_other_hc_configs.py` starts from configs that Doug sends me from module testing. He produces dac, tbm, trim, and mask files, and the above script produces most of the rest of the configs, particularly nametranslation, detconfig, portcard, and portcardmap. (The rest--fedcard, fedconfig, fecconfig, tkfecconfig, ttcciconfig, ltcconfig, amc13, globaldelay25, calib--are static and shouldn't need to be changed.

The slots in the uTCA crate and the fiber bundles

The slots are:

- 1: fed1299
- 2: fed1300
- 3: fed1294
- 4: fed1295
- 5: fed1296
- 6: fed1297
- MCH1 (bottom): NAT MCH
- MCH2 (top): amc13
- 7: fed1298
- 8: empty
- 9: trkfec (tracker FEC)
- 10: pxfec10 (pixel FEC)
- 11: pxfec09 (pixel FEC)

- 12: empty

Some comments:

- The tracker FEC has one mezzanine. Only the bottom 4 transceivers are filled with 8 fibers. (The color to purpose mapping is shown on the paper taped to the top of the shelf, or below.) This 12-way LC ↔ MPO connects to the fiber bundle hanging off the rack labeled on blue tape as "CCU".

Table 1: DOH to Ribbon channel mapping.

DOH	DOH channel	Ribbon channel	Fibre colour	Function
A	PD0	1	blue	CK (from FEC to DOH A)
A	PD1	2	orange	DA (from FEC to DOH A)
B	PD0	3	green	CK (from FEC to DOH B)
B	PD1	4	brown	DA (from FEC to DOH B)
		5	gray/slate	unused (dark)
		6	white	unused (dark)
		7	red	unused (dark)
		8	black	unused (dark)
B	LD1	9	yellow	DA (from DOH B to FEC)
B	LD0	10	violet	CK (from DOH B to FEC)
A	LD1	11	pink/rose	DA (from DOH A to FEC)
A	LD0	12	aqua	CK (from DOH A to FEC)

SFP left					SFP right	Ring pxFEC	Ring tk FEC
H (Clock)	Rx			Tx	G (Data)	Ring 2 B	Link 4
	Tx			Rx			
F (Clock)	Rx			Tx	E (Data)	Ring 2 A	Link 3
	Tx			Rx			
D (Clock)	Rx			Tx	C (Data)	Ring 1 B	Link 2
	Tx			Rx			
B (Clock)	Rx			Tx	A (Data)	Ring 1 A	Link 1
	Tx		PINK	Rx			

- pxfec10 has one mezzanine, fully plugged in with fibers, while pxfec09 has two mezzanines. The bottom is fully occupied with fibers, the top is empty. We don't yet have enough MPO ↔ 12-way fiber bundles for the second mezzanine on pxfec09. So, at most two disks' DOH can be plugged in at once. See Getting the right pixel FEC cables for instructions.

- Do not be confused by the fed numbering versus the roman numerals on top of the fed fc7 extraction handles. The roman numerals correspond to the labels on the jumpers, however: e.g. "VI bot" goes to the bottom receiver on fed1297.
- The fed fiber mapping is as in Hannsjoerg's/my cable map. (In my map, ignore the column "FED used"-- this was the mapping when I didn't yet have all 7 feds.) You don't have to move the bundles, for disks 1-3 they are already plugged in correctly. If you take them out, the 2 digit number on blue tape corresponds to the last two digits of the “POH SN” column in the cable maps.

Post-powercycle of the uTCA crate

After the uTCA crate is power cycled, the fed and fec cards come up running the fc7 default firmware (“golden image”), and will have to have their specialized firmwares loaded. We also have a post-powercycle script that (for now) selects the backplane clock and resets the TTC decoders on the pixel fec and feds.

- Wait until all fc7s have booted to their default golden image firmware. The top led will be solid green, the second from top led will flash blue at about 1 Hz, the third will be off, the 4th (group at bottom) will be solid blue, and the 5th and 6th will be off.
- Get the amc13 in a good state, distributing the clock to the backplane. In the “amc13” terminal tab, issue

  ```
  ./run.sh
  ```

  You should see a status table like JMTBAD. We won’t know if the clock is OK until the last step below.
- In the “fwloader” terminal tab, issue

  ```
  ./powerup.sh tkfec
  ```

  to tell the boards to load all their firmwares, in order: tracker fec (slot 9), pxfec09 (slot 11), pxfec10 (slot 10), fed1294 (3), fed1295 (4), fed1296 (5) fed1297 (6), fed1298. There will be two or three “ERROR - Timeout” messages among “WARNING” messages while the board is rebooting. If you see more and the firmware load has failed, the board reverts to golden image.
- When a board is loading firmware, you will see the top green led flash for a few seconds, then the leds 3-5 will change as appropriate for the board type. (Led 1 and 2 will be solid green and flashing blue as for golden image, while led 6 will be off ~always JMTBAD.)

firmware	**led 3**	**led 4**	**led 5**
golden	off	blue	off
tracker fec	pink	green	pink

pixel fec	start: red flashes pink with good programming	start: green after script below: green if clock OK, orange if not	start: off after running programming: red
fed	green	start: green after script below: green if clock OK, orange if not	start: rapidly flashing red after script below: toggles with triggers

- In the “utca” tab, issue

  ```
  . post_powercycle.sh
  ```

  This tells the pixel fecs and feds to take their clocks from the backplane and resets their TTC decoder blocks. Led 4 should stay green; if orange, something is wrong with the clock. Led 5 on the pixel fec is off (will become red after the first programming happens), while led 5 will stop flashing rapidly and stay in the state it was (on or off). It will toggle red with triggers while running. Yes, it’s annoying that the feds’ leds don’t all wind up in the same state.
- Now everything should be happy; try testing CCU ↔ portcard communication, running VcThrCalDel to check the pixel fec timings, and doing a TBM PLL scan + PixelAlive. If things aren’t happy, you can try power cycling the detector...

Switching to a new disk

See other sections for details on each bullet.

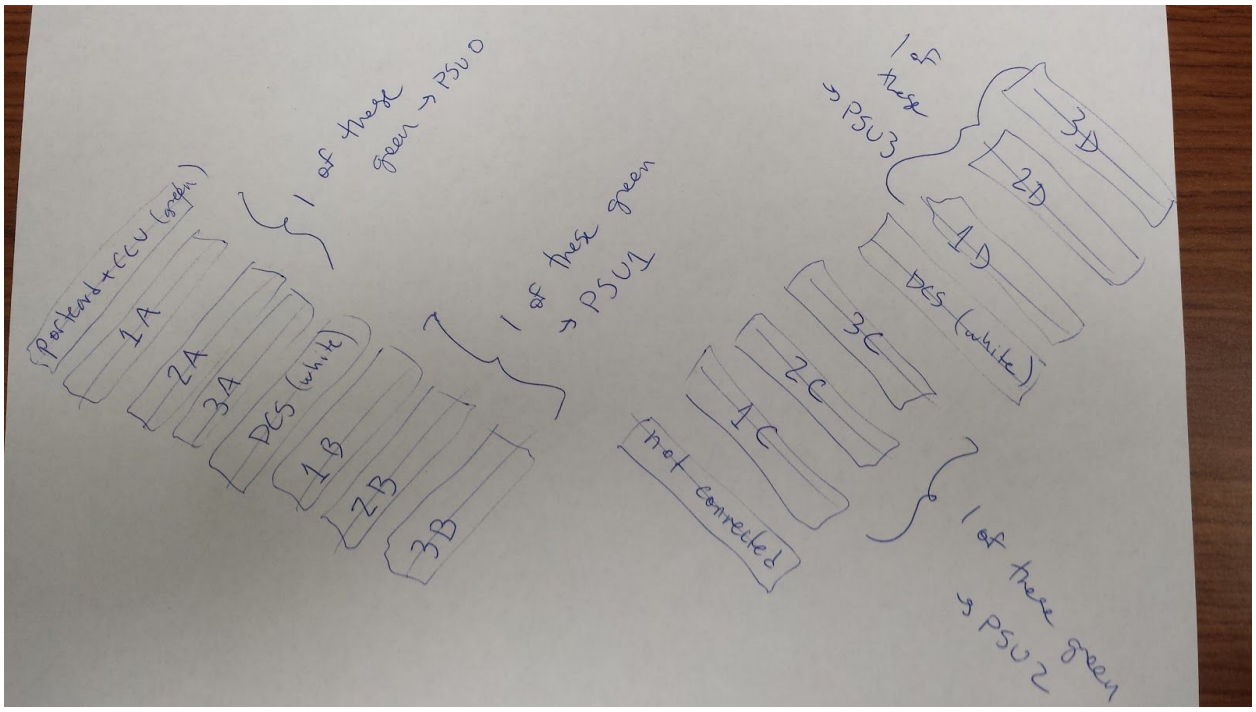
- Power off the current disk at CAEN if necessary
- Switch out the green power cables at the end of the coldbox
- If switching to disk 1 or 3, check that the right pixel FEC fiber bundles are connected.
- Change the configuration to the new disk
- Turn on the disk you switched to
- Configure with POS and check that currents are OK
- Run VcThrCalDel to be sure pixel fec channel timings are OK
- Run PixelAlive until the temperatures for the disks' modules ~stabilize
- Run TBM PLL scan--do even if everything looks "perfect" in the PixelAlive
- Run PixelAlive once more to be sure readouts still perfect--loop back through TBM PLL scan once if not (did you remember to insert the new tbm config key?)
- Now you can run whatever you want.

Power off a disk at CAEN

- Turn off HV channels first. They are not ganged so you have to do two per PSU0-3. The ramp down goes at 10 V/s, so 10 s at the 100 V set point.
- When HV is fully ramped down, turn off PSU0-3 LV channels. Each pair is ganged.
- Turn off that disk's portcards (CH0X = disk X+1) and CCU (CH03).
- Scroll fully to the right and disable that portcard channel. (Not absolutely necessary but I like to do it so I don't accidentally turn on the wrong portcard after switching disks.)

Switch out the green power cables

- Be very careful: if you pull too hard and/or slip, you may jerk on the fiber bundles.
- The three for which the cables come from the top are ziptied for strain relief, you should not have to cut the ziptie to move the cable.
- See the diagram below. (If the small image is hard to read, that piece of paper should be floating around the test stand.) You only have to move 4 cables, leave the other three (1 green, 2 white) where they are.
- Before reconnecting the cables, blow out the water using the nitrogen line, being very careful not to starve the purge that goes into the coldbox. **Do not do this if you have not been trained in person.**
- Be sure the power cables are fully seated.
- PSU0 hooks to one of the A portcards, PSU1 to one of the B, and so on.



Getting the right pixel FEC cables

- All the MPO ↔ 12-way LC bundles that go from the dangling bundles to the pixel fecs are labeled except for those that connect to disk 2's DOHs.
- The MPO end that dangles from the rack is labeled according to the disk + pair of portcards: "PC X-1/2 X AB" for portcards A and B (PRT1 and PRT2 in POS language) for disk X and "PC X-3/4 X CD" for portcards C and D (PRT3 + PRT4). Disk 1 labels are written on blue tape, disks 2 and 3 on orange.
- Disk 2 is always plugged into the upper half of the lower mezzanines, i.e. mfec 2 channels 1 and 2 on fecs 9 and 10.
- Disk 1 and disk 3 bundles have to be swapped out as needed: the "AB" one goes to the MPO <-> 12-way LC marked "PFEC0" with cello tape, "CD" to "PFEC1". If you don't, the DOH do not see the signal from the pxfec on SDA, and will be stuck resetting themselves → you can't talk to the modules, you can't talk to the portcard devices via I2C.

Change the configuration to a new disk

You swap out the nametranslation, detconfig, and portcardmap objects. In the “cable map” terminal tab, run

```
./insert_as.sh disk_number Default
```

where disk_number is 1 for disk 1, etc. This automatically updates all the PixelAlive, TBMDelayWithScores, BumpBondTest, etc., aliases to use the right set of configs.

Turn on a disk

- Turn on portcard first (e.g. CH01 for disk 2): first you have to enable the channel at the right most column. Then you power on. The current should be ~2.7 A at first power.
- Then turn on CCU = CH03. Its current should be ~1 A, and the portcard current should fall slightly (~0.05 A, comparable to the resolution).
- Test the CCU ↔ portcard communication (see separate entry).
- Turn on the PSU0-3 low voltage lines. They are ganged. For HC1, before configuring with POS, the currents should be approximately 2.7-2.8 A on LV0 = digital and 0.35-0.4 A on LV1 = analog power for PSU1-3, and 3 A dig and 0.4 A ana on PSU0. The difference is because 5 modules on portcards B, C, D are not connected.
- Follow Initial configure with POS after detector power up. After all modules are configured, the currents should be about 2.6 A dig and 1.5 A ana for PSU1-3 and 2.9 A dig and 1.7 A ana on PSU0.

- Once the LV has come on fully, turn on the HV for each channel. They are not ganged. The set point is 100 V, and it ramps up over 10 seconds. After fully on, the currents should stabilize around 5 uA (0.5-7 uA is fine).

Test CCU ↔ portcard communication

- In CCU terminal tab, do

  `. init.sh ch`

  where `ch` is 10 for disk 1, 11 for disk 2, and 12 for disk 3. You should see something like

```
$ . init.sh 12
semaphore already exists, created by PID 4129
utca mode
VME FEC will be used with the file connections.xml
FecAccess::FecAccess 6
12-08-16 08:14:29.193011 [7f59dab63840] WARNING - Address overlaps observed - report
file written at
"/tmp/fnaltest/uhal/OverlapReport-home-fnaltest-TriDAS-FecSoftwareV3_0-generic-config-a
ddress_mfec.xml.txt"
FecAccess created



JMT JMT JMT NO RESET


FEC 0x0 Ring 0x0 CCU 0x7e found
FEC 0x0 Ring 0x0 CCU 0x7d found
FEC 0x0 Ring 0x0 CCU 0x7c found
FEC 0x0 Ring 0x0 CCU 0x7b found

CCU Address set to 0x7b

Read I2C Device (CCU 0x7b, channel 0x12, i2c address 0x40):
---->  Value: 0x0

PIA Channel Address set to 0x31
CCU 7b PIA ch 31: get DDR: 0
CCU 7b PIA ch 31: get Data: c0
PIA Channel Address set to 0x32
CCU 7b PIA ch 32: get DDR: 0
CCU 7b PIA ch 32: get Data: 0
PIA Channel Address set to 0x33
CCU 7b PIA ch 33: get DDR: 0
CCU 7b PIA ch 33: get Data: cf
```

```
CCU Address set to 0x7c

Read I2C Device (CCU 0x7c, channel 0x12, i2c address 0x40):
---->  Value: 0x0

PIA Channel Address set to 0x31
CCU 7c PIA ch 31: get DDR: 0
CCU 7c PIA ch 31: get Data: c0
PIA Channel Address set to 0x32
CCU 7c PIA ch 32: get DDR: 0
CCU 7c PIA ch 32: get Data: 0
PIA Channel Address set to 0x33
CCU 7c PIA ch 33: get DDR: 0
CCU 7c PIA ch 33: get Data: cf
CCU Address set to 0x7e

Read I2C Device (CCU 0x7e, channel 0x12, i2c address 0x40):
---->  Value: 0x0

PIA Channel Address set to 0x31
CCU 7e PIA ch 31: get DDR: 0
CCU 7e PIA ch 31: get Data: c0
PIA Channel Address set to 0x32
CCU 7e PIA ch 32: get DDR: 0
CCU 7e PIA ch 32: get Data: 0
PIA Channel Address set to 0x33
CCU 7e PIA ch 33: get DDR: 0
CCU 7e PIA ch 33: get Data: cf
CCU Address set to 0x7d

Read I2C Device (CCU 0x7d, channel 0x12, i2c address 0x40):
---->  Value: 0x0

PIA Channel Address set to 0x31
CCU 7d PIA ch 31: get DDR: 0
CCU 7d PIA ch 31: get Data: c0
PIA Channel Address set to 0x32
CCU 7d PIA ch 32: get DDR: 0
CCU 7d PIA ch 32: get Data: 0
PIA Channel Address set to 0x33
CCU 7d PIA ch 33: get DDR: 0
CCU 7d PIA ch 33: get Data: cf
```

- If instead you see a lot of messages with `Exception caught when doing PIA access` like

```
Exception caught when doing PIA access: FecExceptionHandler:
```

```
    What: device is already connected by another thread
    File: /home/fnaltest/TriDAS/FecSoftwareV3_0//generic/src/common/FecAccess.cc line:
3105
    Method: keyType FecAccess::addPiaAccess(keyType, enumAccessModeType)
    When: Thu Aug 11 17:45:59 2016
    Error code: 810
    Error from the FEC high level layers
        FEC 0x0 ring 0x0 CCU 0x7e channel 0x33
```

you have to powercycle the CCU: turn off, wait a second, turn on. Make sure the CCU reaches full voltage / current (1 A) before trying init.sh again. Keep powercycling until you see something like the above. (The values for PIA Data will not be the same, and I2C read values will not be the same if you have already programmed the portcard with POS.)

- If you see the PIA lines, but not the I2C one, instead seeing another exception like

```
------------ Exception ----------
FecExceptionHandler:
    What: Fails on sending a frame, ring lost
    File: /home/fnaltest/TriDAS/FecSoftwareV3_0//generic/src/common/FecRingDevice.cc
line: 705
    Method: virtual void FecRingDevice::writeFrame(tscType8*, bool)
    When: Thu Aug 11 17:45:59 2016
    Error code: 210
    Error from the FEC high level layers
        FEC 0x0 ring 0x0
        FEC status register 0 = 0x490
---------------------------------
```

that means you cannot do the I2C read from the delay25 chip, probably because the pixel FEC is not hooked up (you forgot to switch FEC fiber bundles, see the section above).

- The meanings of the PIA DDR and Data bits are explained in [this google doc](). In short, you should see:
    - DDR = 0 for every channel
    - Ch 31 Data = c0
    - Ch 32/33 Data according to which disk you are operating in this table

Disk powered	Ch 32 Data	Ch 33 Data
1	f	90
2	f0	a0

3	0	cf

Some POS reminders

- Install-dependent environment variables are set by the `local.sh + setenv.sh` scripts. If you want a new terminal in which to do (most) anything POS-related, you need to source setenv.sh:

  ```
  cd ~/TriDAS
  . setenv.sh   # this sources local.sh too
  ```

- The environment variable `$BUILD_HOME` is `/home/fnaltest/TriDAS` on dellfromdoug.
- `$JMT` is `$BUILD_HOME/pixel/jmt`.
- Runs live in `$POS_OUTPUT_DIRS = $BUILD_HOME/pixel/PixelRun/Runs`, e.g. `$BUILD_HOME/pixel/PixelRun/Runs/Run_1000/Run_1402`.
- Log files go in `$POS_LOG_DIRS = $BUILD_HOME/pixel/PixelRun/Logs`.
- The configuration files are in `$PIXELCONFIGURATIONBASE` = `$BUILD_HOME/Config`. In particular:
  - The map of config aliases to global keys (e.g. PixelAlive) is in `$PIXELCONFIGURATIONBASE/aliases.txt`, along with the list of version aliases for subkeys (calib, tbm, amc13, etc.)
  - The list of which sub keys comprise global keys is in `$PIXELCONFIGURATIONBASE/configurations.txt`.
- Given a run number, e.g. 1401, you can find out which e.g. tbm key was used by following this example:

  ```
  $ cat $POS_OUTPUT_DIRS/Run_1000/Run_1401/PixelConfigurationKey.txt
  Pixel Run Alias / Run Type = BumpBondTest1
  Pixel Global Configuration Key = 5799
  ```

  You then find that key in `configurations.txt`:

  ```
  key 5799
  detconfig   110
  portcardmap   66
  nametranslation   67
  fedcard   100
  dac   106
  lowvoltagemap   100
  maxvsf   100
  mask   14
  trim   100
  ```

```
tbm   128
dcdc   100
portcard   116
fedconfig   100
fecconfig   100
tkfecconfig   0
ttcciconfig   0
ltcconfig   0
amc13   5
globaldelay25   0
calib   101
```

The sub keys can be in any order. Then the files for e.g. tbm are in `$PIXELCONFIGURATIONBASE/tbm/128/`.

Initial configure with POS after detector power up

Follow the instructions in Running a calibration with POS through clicking Configure, using VcThrCalDel as the calibration. I use this to be sure the timing of the system is “right” after first startup--see Analyzing VcThrCalDel. You can run it whether some fibers have bad phases or not (the corresponding plots will show up empty or with decoding errors), but at least all the modules will be configured. You should also check that most / all phases are good (What good FED phases look like). They may not be after a power cycle, in which case you need to run the TBM PLL scan.

Analyzing VcThrCalDel

- In the “runs” terminal tab, the alias `vccd` will cd to the last run directory (alias clr), extract the plots from the root file, and dump into a pdf that is then launched with evince.
- The current settings are marked with blue-grey lines, while the calibration algorithm extracts a new setting in white. The current settings should be fine--well within the red of the tornado--and should not be adjusted unless necessary. One module that is an exception is BmI,3,4,1,2, which seems to drift by as much as ½ BX. (Why???)

Running a calibration with POS

- In the “pos” terminal tab, Ctrl-C to kill the old XDAQ session. We don’t have to do this, but I like to have the log files separate for separate runs.
- Restart XDAQ with `r`. This is an alias you can see the definition of with `alias r`.
- Go to the main XDAQ page in Firefox http://dellfromdoug:1973/
- Click PixelSupervisor (it has no icon).
- Initialize. (After doing this, do not Initialize again. You can’t from the GUI, but if you use the expert bookmark “init”, you will crash the POS.)

- Select the calibration you want from the radio menu and click Configure.
- Watch the terminal for errors, in particular from the tracker fec communication to the portcard (exceptions will spew), and be sure all the printed phases from the feds are OK.
- Click Start, and make a note of the run number, calibration type, and any other relevant notes about the state of the system in your elog. (The run number is displayed on the PixelSupervisor page.)
- Watch that it doesn't crash. During the run, the red lights on the FED should toggle with every trigger, and every 1000 events the data is printed to the screen.

What good FED phases look like

The string of 1s may be shorter, or longer--pixel alives will tell you if you have decoding problems.

```
 9   0  23  23    00000000000000001111100000000000  0 31 16 10  0  2 23
11   0  23  23    00000000000000001111110000000000  0 31 15  9  0  2 23
12   0  18  18    00000000000000000000000000111110  0 31  6  0  0  2 18
21   0  14  23    11110000000000000000000000000001  1  0  0 27  1  1 14
23   0  18  18    00000000000000000000000000111100  0 31  6  1  0  2 18
24   0  11  11    00000111000000000000000000000000  0 31 27 23  0  2 11
```

TBM PLL scan

- Follow Running a calibration with POS, choosing TBMDelayWithScores from the list.
- A description of the calibration algorithm is in this separate google doc.
- When done, the root file and new TBM*.dat are written out in the run directory. The calibration code isn't yet smart enough to insert this as a new tbm key. Find out what the current tbm key is (see Some POS reminders), say 128, and what key you want the new settings to be (usually the first unused one), say 129. Then you can do

  ```
  cd $POS_OUTPUT_DIRS/Run_1000/Run_1406
  $JMT/insert_tbms.sh 128 129
  ```

  to merge the new settings into the old set, and the new Default alias will be set to new_key_number.
- Take a PixelAlive to confirm the new settings are good / better than before.

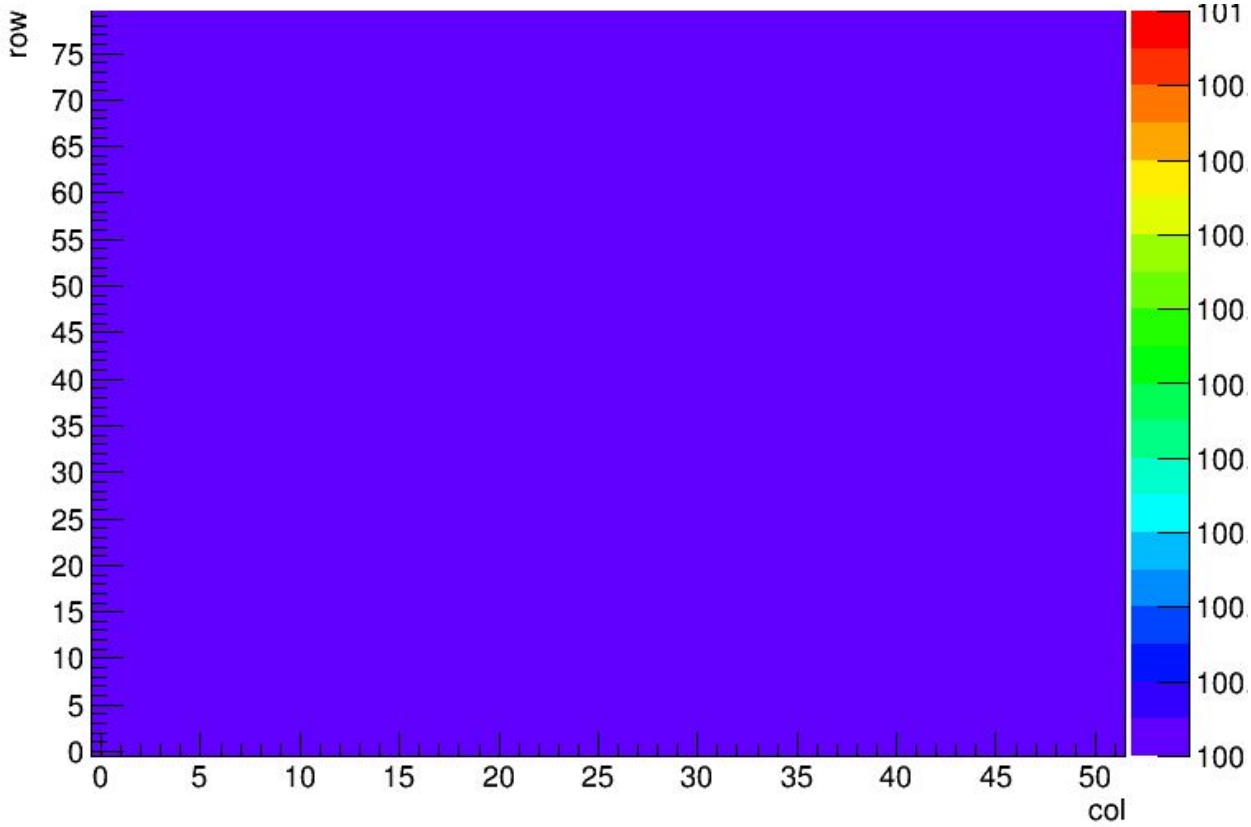
PixelAlive for checking the state of the readout

- Follow Running a calibration with POS, choosing PixelAlive from the list.

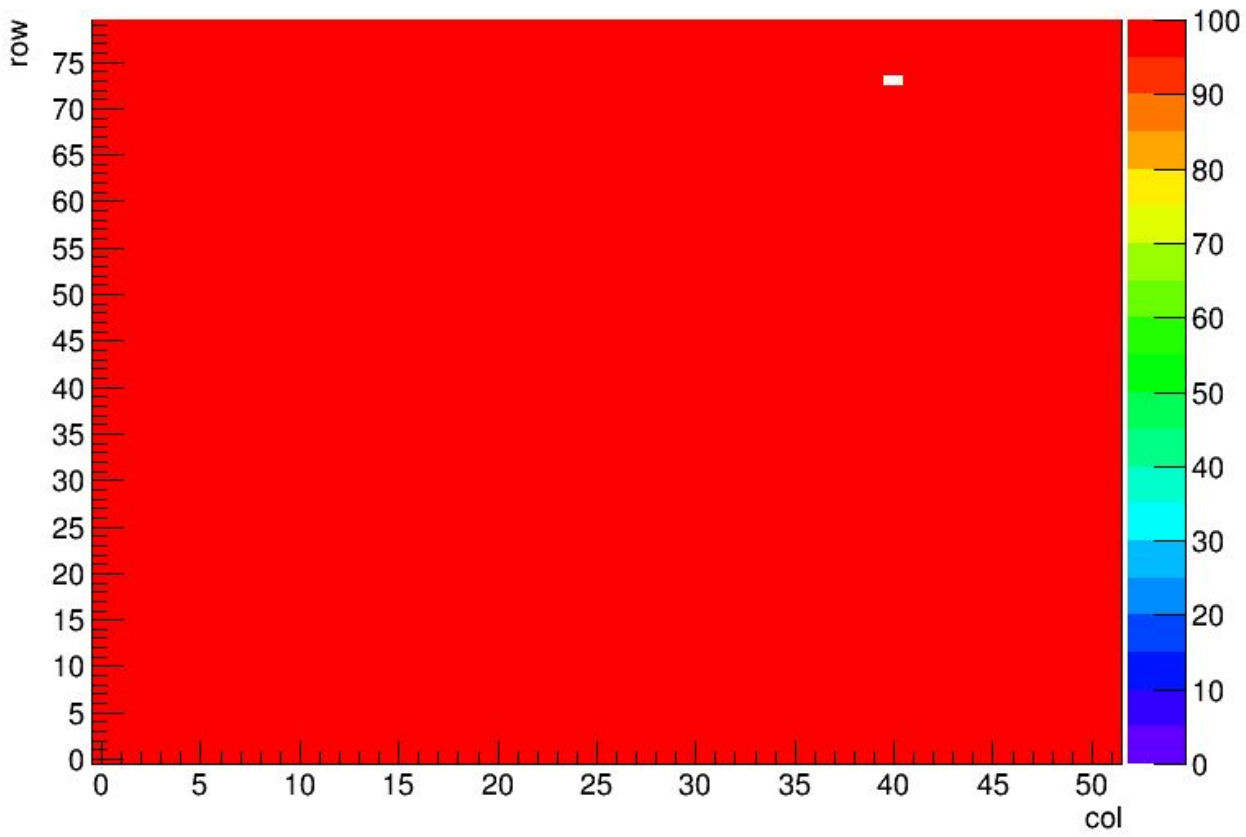
- In the “runs” terminal tab, issue `pxal`, an alias that cds to the last run directory, runs the pixelalive analysis script to create the root file from the dmp files, extracts the plots from the root file, and displays the results in a pdf.

Currently there is some issue with the readout such that the pixelalives are not what we would normally call perfect. There is no issue with the modules seen yet, so we redefine perfect. Examples are below, extracted from a reference run for disk 3, with full results on dellfromdoug in `~fnaltest/run1410.pdf`. Other reference runs are 1231 for disk 1 and 1221 for disk 2.

- Truly perfect readouts and rocs will show up purple:

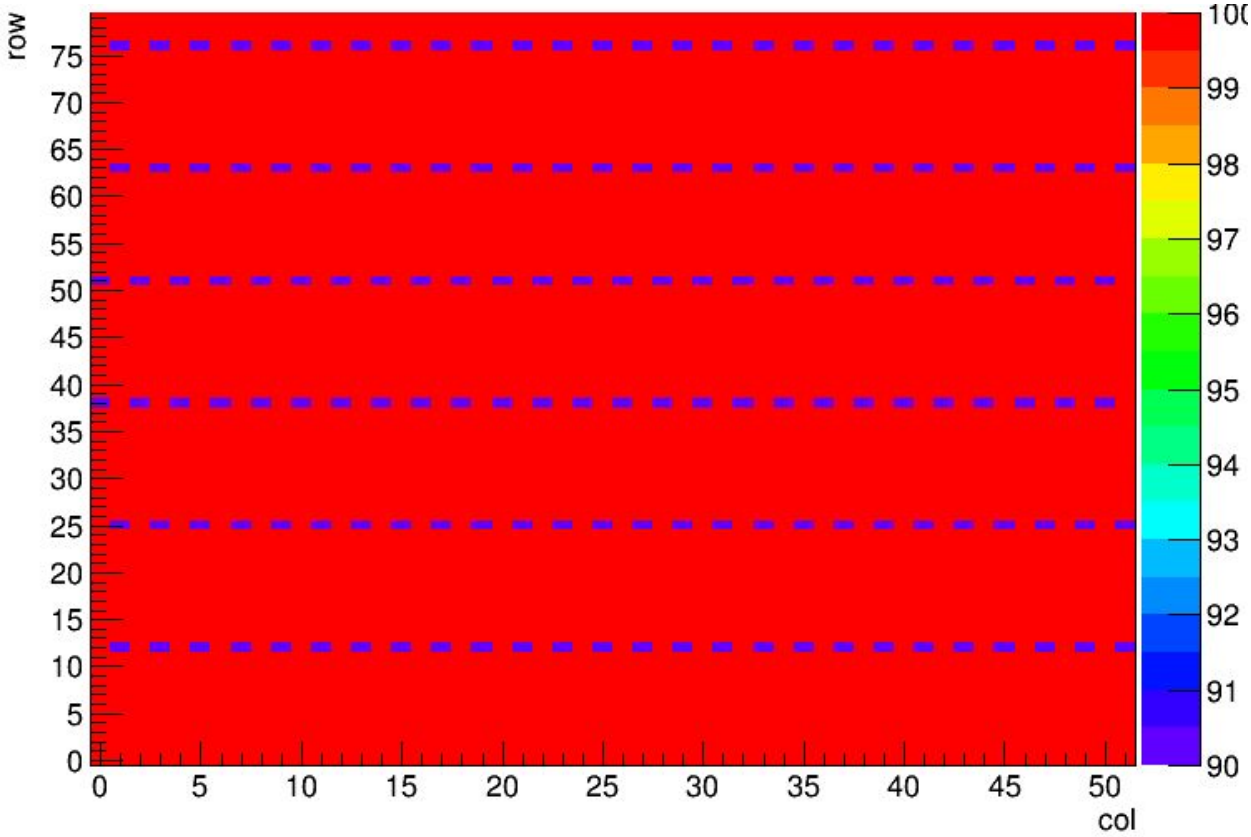


- Rocs with dead/inefficient pixels but otherwise perfect readouts will show up read with white or other-colored holes:



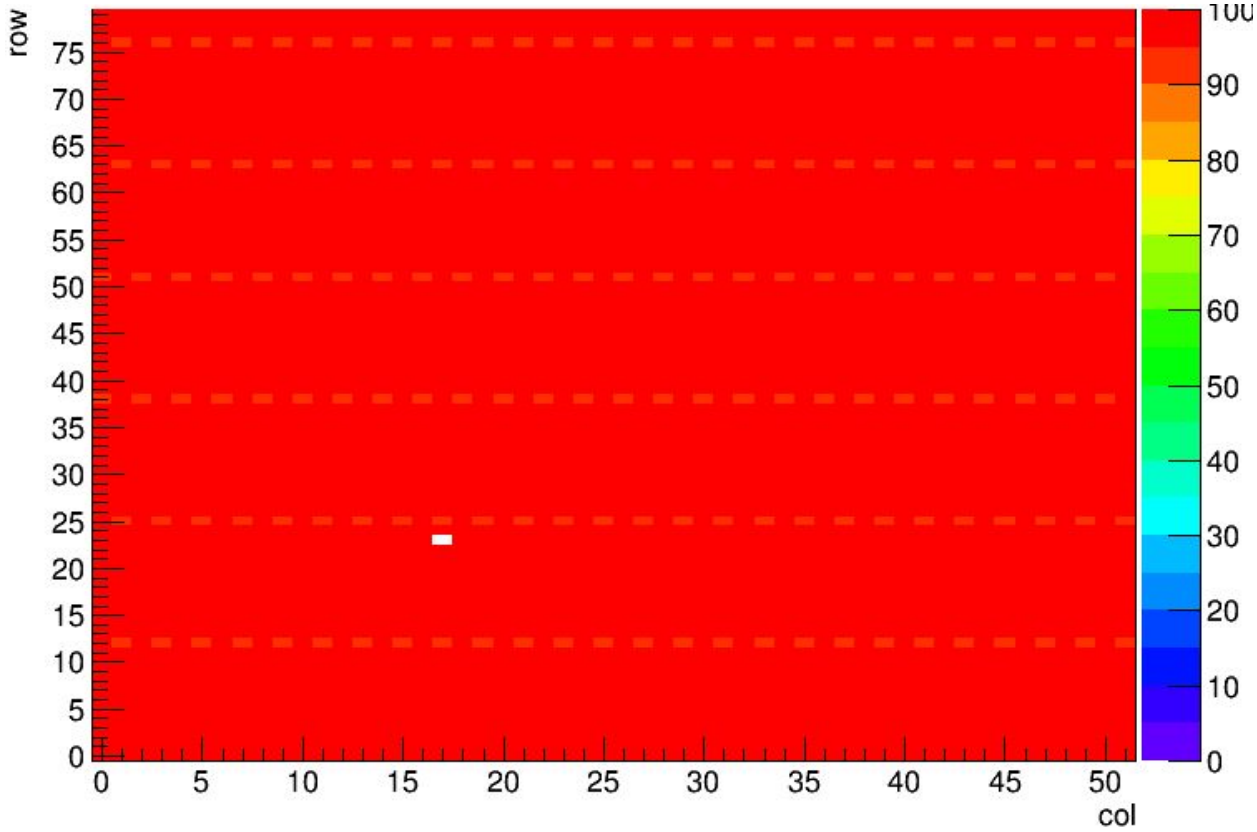
- A stripey pattern can happen at regular intervals up the chip. It is some feature of the readout. It goes away with different TBM phase settings, and when it appears on one roc it appears on the rest of the rocs for the same tbm core: rocs0-7 are core A, 8-15 are core B. (The stripe moves around with different number of triggers per point in a predictable way, and the inefficiency deepens with fewer triggers per point, so it is some weird deadtime somewhere. But hard to track right now since it can also just go away on

its own, run to run. Watch the elog for developments.)



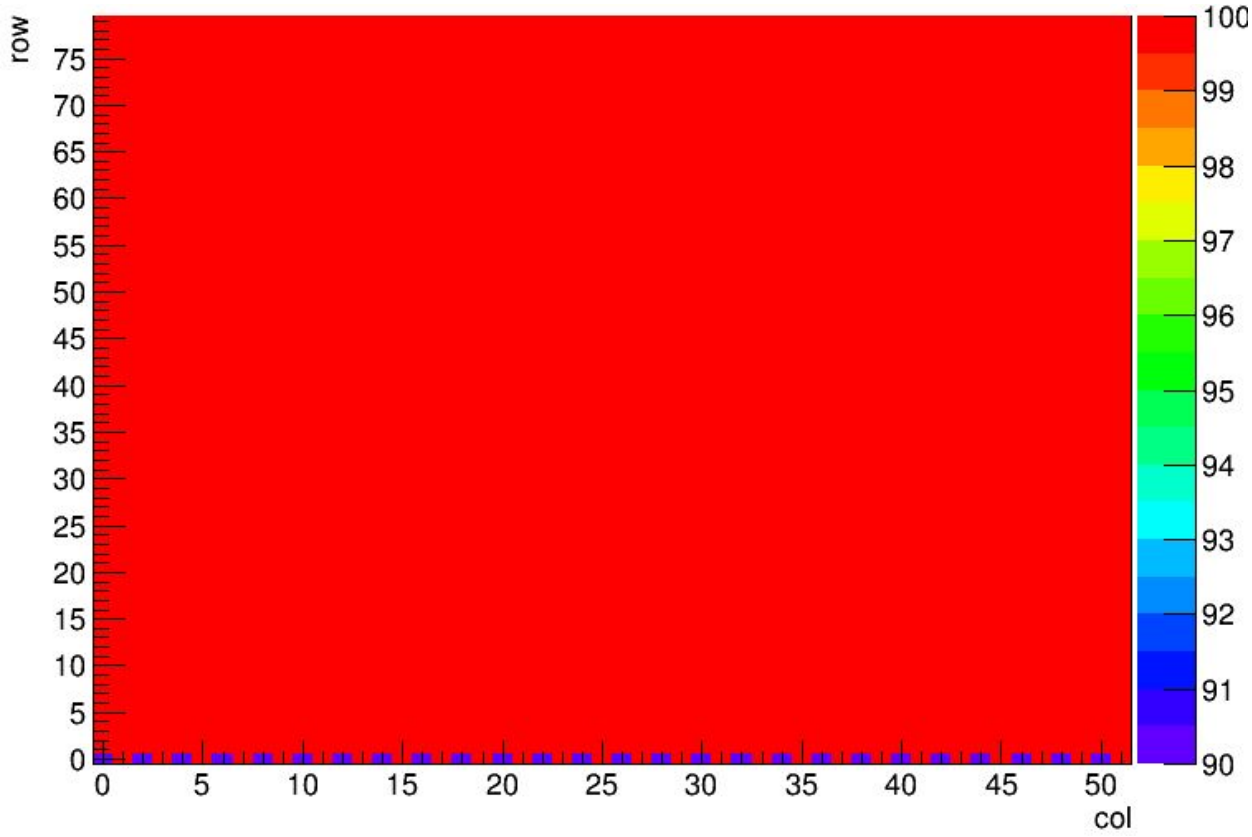
- If there is a dead/inefficient pixel, the color range might change such that the stripey pattern only appears faint, and depending on your screen type, you might have to move

your head around to see it.



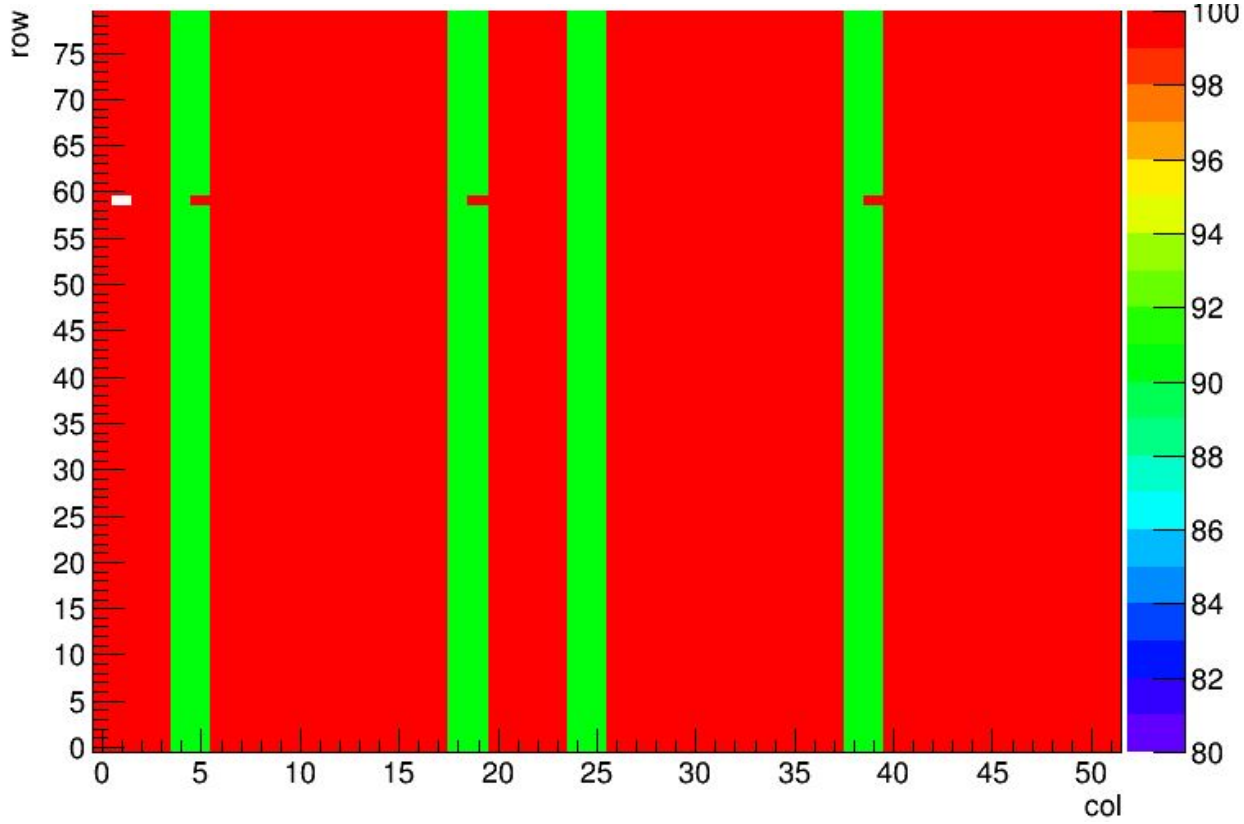
- Another stripe can happen when the first trigger is lost. Again, this changes with TBM phase settings, happens for all the rocs in a TBM core, and goes away on its own

run-to-run.



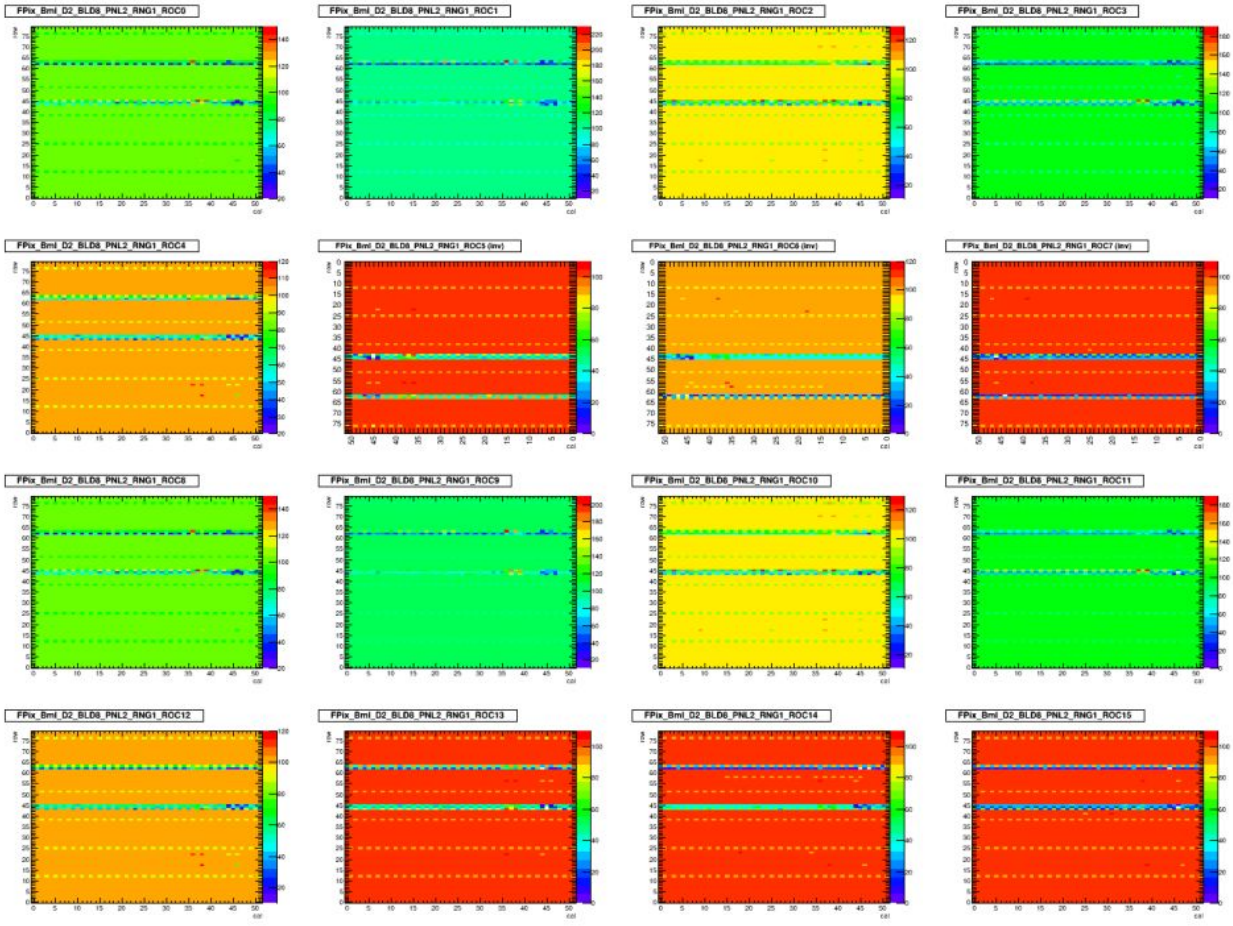
- BmI,3,3,1,2 ROC13 ends up with some inefficent double columns. I have seen this go away, and I have seen it on other modules' roc13, so I think this may be just decoding errors and may be fixed with a roc delay scan. This has not yet been done for lack of

time.



- When there are problems with phases, a group of rocs (4), a tbm core, or the whole module will look spazzy. Then you should run a TBM PLL scan and recheck the

PixelAlive, at least.



SCurve runs + analysis (current shifts)

These are the vcal s-curves used to test the quality of the trimming and the noise (related to the width of the turn-on) compared to pre-installation module testing. Each chunk takes 26 minutes, and there are 80 chunks per disk.

I think we've got the shifts planned such that you should not need to do anything special other than go from one chunk to the next, checking the data along the way.

I'll handle setting up the detector after cooling + power cycles, and switching from one disk to the next, but if I can't, you can try the appropriate instructions in the other sections.

After each chunk, run the analysis script and check the plots to be sure you are taking good data.

E.g. after SCurveX finishes as run Y:

- Every 5 chunks, take a PixelAlive to check the state of the readout.

- Start SCurve$((X+1)) (very similar to any other calibration run):
    - In the “pos” terminal tab, Ctrl-C to kill the old XDAQ session
    - restart XDAQ with `r` (alias described in POS sections)
    - go to the XDAQ page in Firefox http://dellfromdoug:1973/
    - click PixelSupervisor
    - Initialize
    - Select “SCurve$((X+1))” in the radio menu and click Configure
    - Watch the terminal for errors, and be sure all the printed phases are OK
    - Click Start
    - Watch that it doesn’t crash. During the run, the red lights on the FED should toggle at ~5 Hz (9 Hz triggers), and every 1000 events the data is printed to the screen.

- In the “runs” tab, run the analysis script on the data from SCurveX:

  ```
  cd TriDAS/pixel/PixelRun/Runs/Run_1000
  ./doit_scurve.sh Run_Y
  ```

  Note: If for some reason you are running on an incomplete detector configuration, open:

  ```
  ~/TriDAS/pixel/PixelAnalysisTools/test/configuration/SCurveAnalysis.xml
  ```

  Change field <Feds Analyze=”nnnn-nnnn…-nnnn> (currently line 25) to a dash separated list of FEDs actually used to take the data. There should be one SCurve_n.err and one SCurve_n.dat file (where n is the FED number) for each FED used, in folder Run_Y.

  Once the script is complete, a PDF will appear on-screen. Check that there are the appropriate number of entries in the plots. There should be (55 modules for disk 1,3 or 42 modules for disk 2) * (16 rocs / module) in each roc plot, and (52 pixels / roc) * (number of rocs) in the two “all Pixels” plots. Also check that the distributions look OK: thresholds should be 35-36 vcal units, with noise <~1 vcal unit.

BumpBondTest runs + analysis

Each chunk takes 26 minutes. After every chunk, run the analysis script and check the plots to be sure you are taking good data.

E.g. after BumpBondTestX finishes as run Y:

- Start BumpBondTest$((X+1)) (very similar to any other calibration run):

    - In the “pos” terminal tab, Ctrl-C to kill the old XDAQ session
    - restart XDAQ with `r` (alias described in POS sections)
    - go to the XDAQ page in Firefox http://dellfromdoug:1973/
    - click PixelSupervisor
    - Initialize
    - Select “BumpBondTest$((X+1))” in the radio menu and click Configure
    - Watch the terminal for errors, and be sure all the printed phases are OK
    - Click Start
    - Watch that it doesn’t crash. During the run, the red lights on the FED should toggle at ~5 Hz (9 Hz triggers), and every 1000 events the data is printed to the screen.

- In the “runs” terminal tab, run the analysis script on the data from BumpBondTestX:

  ```
  cd TriDAS/pixel/PixelRun/Runs/Run_1000
  ./doit_bb3.sh Run_Y
  ```

  Note: If for some reason you are running on an incomplete detector configuration, open:

  ```
  ~/TriDAS/pixel/PixelAnalysisTools/test/configuration/SCurveAnalysis.xml
  ```

  Change field <Feds Analyze=”nnnn-nnnn…-nnnn> (currently line 25) to a dash separated list of FEDs actually used to take the data. There should be one SCurve_n.err and one SCurve_n.dat file (where n is the FED number) for each FED used, in folder Run_Y.

  Once the script is complete, the PDF will appear on-screen. Flip through the pages to see if things look OK: are there the right number of pages (about number of rocs / 10), are there the right number of entries in the plot (2 rows * 52 columns = 104 total, 52 in each of top and bottom), do the distributions look not-screwed-up, etc.