psi46test at DESY

Daniel Pitzl, DESY

18.6.2014

1 Introduction

psi46test is C++ code to test CMS pixel readout chips (ROCs) from a PC via USB and a pixel test board. It was developed by Beat Meier at PSI since 2006 to test analog psi46 ROCs with analog ATB test boards. Functions for digital psi46dig ROCs were added since 2012 and support for digital DTB test boards started in 2013. The code was developed under Windows but as plain C++ code without a graphical user interface it was always running under Linux or on Macintosh as well. The PSI code is available from git (`git clone https://github.com/psi46/psi46test.git`).

The DESY development branched off in March 2014 when DTB FW/SW version 2.0 appeared. The command line interface is kept. Output from tests is still written in ASCII format to the log file, but in addition ROOT histograms are booked, filled, stored, and displayed in a static canvas. The code is extended with higher level tests for optimizing DAC settings. The state of the ROC (and the test board) is represented in software. dacParamter and trimParameter files can be written in the same format as used by psi46expert and pXar. Code for the wafer prober at PSI was removed. The code is always tied to a specific version of the DTB firmware and software (via the RPC remote procedure call mechanism), which can be inspected at `https://github.com/psi46/pixel-dtb-firmware).` This manual is supposed to document the tests available in the DESY version of psi46test.

1.1 Installation

Register at github (invent a user name and password).

Ask Claudia.Seitz@desy.de to add your user name to the repository.

While you wait, install the usb driver library: `libftd2xx.so` from ftdi.com

A ROOT installation is required (and `make`, and a C++ compiler)

```
git clone https://user@github.com/clseitz/Psi46testDesy.git
cd Psi46testDesy
make
```

1.2 Start

Connect a DTB via a USB cable to your computer.

```
cd Psi46test
bin/psi46test d.log
```

You should see something like

```
psi46test for DTB V2.2 (4.6.2014)
reading psi46test.ini...
```

```
logging to d.log
```

(if you get an USB error here that the port cannot be opened on your computer try `. initb.sh` on Linux)

```
USB opened DTB_WS6MP2
DTB DTB_WS6MP2 opened
```

(if your output stops here, the DTB firmware may be outdated (before 2.0). You need a psi46test version compatible with any FW from 1.06 to 1.26 and then upgrade dtb_v2.xy.flash to the current version)

```
--- DTB info-------------------------------------
Board id:  77
HW version:  DTB1.2
FW version:  2.2
SW version:  2.21
USB id:  DTB_WS6MP2
MAC address:  40D85511804D
Hostname:  pixelDTB077
Comment:
-------------------------------------------------
PC hash 333290928
DTB hash 333290928
RPC call hashes of PC and DTB match:  333290928
ROOT application...
+-cmd commands ----------------------+
| help list of commands |
| exit exit commander |
| quit exit commander |
+-----------------------------------+
gainFile:  /home/pitzl/psi/dtb/tst215/phroc-c405-trim30.dat
open ROOT window...
MyMainFrame...
open Canvas...
> exit
```

2 commands

Commands are defined in cmd.cpp. To add a command, put the code in a new block `CMD_PROC(cmdname){}` somewhere in the file and register it in the `cmd()` section (towards the end) with a help string: `CMD_REG( cmdname, "cmdname <argument> explain what it does" )`. No header files are involved.

Commands may either be entered interactivley at the prompt, or read from a file like `script/mycmd.roc` which gets called from the prompt by giving the file name without the .roc extension: `> mycmd` (the path `script/` is defined in `psi46test.ini`).

Commands check for their mandatory arguments and don't execute if they are missing or out of range. `help` prints a list of commands with their parameters.

The state of the ROC (DACs, trims, thresholds) is represented in software. A test (e.g. a DAC scan) is supposed to put back the original state, unless a DAC is changed on purpose.

Commands and measurements are written to the log file. This was implemented for offline parsing, processing, and plotting. It is still useful for reconstructing the conditions under which a particular test in a session was executed. Measurements are now also written as 1D and 2D histograms into a ROOT file `Test.root`, for direct plotting and offline processing.

2.1 DTB commands

These commands don't require the presence of a ROC. Use them to check the USB connection and the board.

rpcinfo prints the list of functions available in the DTB SW via RPC.

upgrade dtb_v2.21.flash load new firmware/software into the FPGA. Wait until the LEDs are off. Exit psi46test. Power cycle the DTB (unplugging the power cable) to make sure that the new executable is loaded from EPROM.

help prints the list of commands defined in cmd.cpp

info prints DTB info

version prints DTB hardware, firmware, and software version numbers

boardid prints the production serial number

welcome play the LED startup sequence

setled bits play with the four LEDs on the test board (bit pattern: 0 all off, 15 all on)

pon low voltage on

getva measure the analog supply voltage on the board

getvd measure the digital supply voltage on the board

va mV set the analog supply voltage (range 0 to 3000 mV, 1700 mV is fine)

vd mV set the digital supply voltage (range 0 to 3000 mV, 2500 mV is fine)

poff low voltage off. Do this before exiting from psi46test.

quit (or exit) closes the DTB connection, the log and ROOT files and ends psi46test.

If you want to keep them, `rename` d.log and/or Test.root before you start again, otherwise they get overwritten.

2.2 starting up with a ROC

Connect a ROC (or module) via SCSI cable to the DTB. You may also want to connect (negative of about -150 V) bias voltage to the red-ringed lemo connector.

Start again: `bin/psi46test c405.log`

s405 execute start-up commands for a given chip (`script/s405.roc`)

getid measure digital current, should be around 22-28 mA per ROC

getia measure analog current, should be around 25 mA per ROC. If it is around 5 mA the ROC is not properly programmed. Inspect the settings in the start script. The problem is either here, or the ROC is dead.

deser160 2D scan of clock phase and 160 MHz deserializer phase (for single ROCs), searching for the proper ROC header 7FA (hex). Sets the new values, if successful.

DAC	name	comment	DAC	name	comment
1	Vdig		17	VoffsetRO	
2	Vana		19	Vcomp_ADC	
3	Vsf		20	VIref_ADC	
4	Vcomp		22	VIColOr	
7	VwllPr		25	Vcal	
9	VwllSh		26	CalDel	
10	VhldDel		253	CtrlReg	
11	Vtrim		254	WBC	
12	VthrComp		255	RBreg	
13	VIBias_Bus				

Table 1: DAC paramters for psi46digV2.1

The following commands elicit a response from the ROC only if it is properly set up (Vana, WBC, CalDel, VthrComp are the most critical DACs). See below for more algorithmic procedures.

fire col row [nTrig] pulse one pixel. Columns are 0..51, rows are 0..79

arm col row enable column, un-mask one pixel and prepare it for calibrate pulses

arm col:col row:row enable range of columns, un-mask ranges of pixels and prepare for calibrate

single single calibrate event display (one cycle of reset-calibrate-trigger-token or whatever is programmed in the pattern generator)

cole col enable one column

cole col:col enable range of columns e.g. cole 0:51 for all)

pixe col row unmask one pixel

pixe col:col row:row unmask range of pixels (e.g. pixe 0:51 0:79 for the entire ROC)

cal col row activate column and row for calibrate

cal col:col row:row activate ranges of columns and rows for calibrate (cols and rows independently

cald clear calibrate from all pixels

mask all pixels

cold col disable column

pixd col row disable pixel

2.3 DAC parameters

Table 1 shows the DAC parameters for psi46digV2.1.

2.4 setting DAC parameters

optia target set Vana to get the desired target analog current [mA], .e.g. optia 25

show current DAC settings (presumably, from book-keeping in psi46test; reading back DACs from the ROC is not possible)

dac number value set a DAC value (number is the DAC address). Some DACs have shortcuts:

vana value set Vana [0:255] (check analog current: 1 mA / 6 DAC units)

vthr value set global threshold VtrhComp [0:255]

vcal value set test pulse amplitude [0:255]

ctl set control register (0 = small Vcal, 4 = large Vcal, 1 = ROC off)

caldel col row measure pixel efficiency vs `CalDel`, set `CalDel`

caldelroc scan CalDel for the entire ROC (perfect pixels respond to all triggers, alive pixels have at least 50% response), sets CalDel in the plateau region

thrmap guess measure pixel threshold map for current settings (faster if guess is close to truth)

vthrcompi

vthrcomp target [guess] set VthrComp such that the minimum pixel threshold is at target Vcal units

trim target set Vtrim and trim bits such that all pixel thresholds are as close as possible to target Vcal units

effmap nTrig measure efficiency map (PixelAlive) with n triggers per pixel

trimbits adjust trim bits to recover maximum efficiency

wtrim chip write current trim bits to `trimParameters_chip.dat`

phmap ntrig measure pixel pulse height map

tune set gain and offset such that the pulse heights of all pixels are in 80% of the ADC range, for large and small Vcal, with 10% margins against overflows and underflows

wdac chip write current DAC settings to `dacParameters_chip.dat`

2.5 DAC scans

For diagnostic purposes: scan a dac (or two) for one pixel or the entire ROC, and measure efficiency, pulse height, or threshold. The DAC is not changed.

effdac col row dac count trigger responses (efficiency) for one pixel vs a DAC

phdac col row dac pulse height (ADC) vs DAC for one pixel

calsdac col row dac sensor calibrate pulse height (ADC) vs DAC at CtrlReg 4 (high range Vcal) for one pixel

thrdac col row dac threshold (in small Vcal units) vs DAC for one pixel

dacdac col row dacx dacy 2D DAC-DAC scan for one pixel, pulse height and efficiency

dacscanroc dac [nTrig] maps of pulse height and efficiency vs dac for all pixels (dac 25 at ctl 0 gives S-curves, dac 25 at ctl 4 gives gain calibration, dac 26 gives CalDel)

gaindac calibrated pulse height vs Vcal for all pixels, checks the gain calibration

2.6 maps

effmap nTrig pixel efficiency map (PixelAlive)

thrmap guess measure pixel threshold map for current settings (faster if guess is close to truth)

phmap nTrig pulse height map (vary vcal and ctl to explore full range)

2.7 sensor calibrate and bump bond test

The ROC test pulse may be directed towards a pad on the surface of each pixel, inducing a (small) charge into the sensor across the air gap capacitance, which can be detected if the bump bond connection is good.
The tests internally select `ctl 4` to get the large test pulse range (and set it back to previous).

cals col row active one pixel for sensor calibrate (requires `cole col` and `pixe col row` to see the response with `single`)

calsdac col row dac sensor calibrate pulse height (ADC) vs DAC at CtrlReg 4 (high range Vcal) for one pixel

calsmap nTrig sensor calibrate pulse height map

bbtest nTrig sensor calibrate pulse height map with bump bond statistics

dacscanroc dac -nTrig sensor calibrate (selected by negative nTrig) pulse height and efficiency vs dac for all pixels (`dac` 12 at `ctl` 4 gives bump bond test)

2.8 data taking

The pattern generator on the DTB can be operated in a loop, repeating its programmed cycle (typically reset-cal-trigger-token = rctk) at an adjustable rate. The rate is determined by the clock frequency (typically 40 MHz) and the sum of the delays in the pattern generator sequence (at least WBC) plus a programmable delay: R = f / N, e.g.

pgloop 1000 gives a rate of 40 kHz.

A DAQ process is started on the DTB such that the FPGA writes the deserialized raw data into memory. Up to 50 M words (100 MB) can be stored. The pattern generator loop and the DAQ are stopped every few ms and the memory is readout via USB. This introdcues some dead time but allows for almost concurrent decoding and display of the data.
The result are random trigger hit maps and pulse height distributions, which can be used with sources (X-ray, Sr, Ru), with fixed test pulse patterns (arm), or just with noise (at lowest thresholds, enable all pixels).

3 algorithms

Details about algorithms that set DAC parameters.

3.1 analog current

DAC `Vana` controls the analog current that supplies pre-ampliefer and shaper of each pixel. The current is measured on the test board (`getia`). There is an offset current of about 5 mA per ROC for `Vana` = 0. At full range (`Vana` 255) the current is about 45 mA per ROC, with an approximately linear dependence and a slope of 1 mA / 6 DAC units. The design operating analog current is 24 mA / ROC. Command `optia target` takes the desired current [mA] as an argument and tries to adjust `Vana` accordingly. It usually succeeds in a few iterations.

3.2 timing

Coarse (unit: BC = clock cycles = 25 ns): `WBC = tct - 7` for psi46digV2.1, WBC = tct - 6 for earlier digital ROCs, WBC = tct - 5 for analog ROCs, where tct ist the time between calibrate and trigger in the pattern generator sequence (e.g. WBC 99 for tct 106).
Fine: CalDel shifts the timing of the test pulse, unit: 1 DAC = 0.4 ns, dynamic range 0..255 = 100 ns = 4 BC.

3.3 threshold trimming

Symbolic equation: pixelThreshold = globalThreshold - `Vtrim (15 - trim bits)`, where the pixel threshold is determined from a Vcal scan with several triggers per point, searching for the point with 50% response. In low Vcal range (CtrlReg 0), a Vcal threshold of 30 DAC units corresponds to about 1500 electrons signal. `VthrComp` and `Vtrim` are global DACs, affecting the entire ROC, while the four `trim bits` can be set for each pixel individually. As the equation shows, the trimming can only lower the threshold from the value determined by `VthrComp`. The `trim bits` act inverted: 15 means no effect, while 0 gives the maximum threshold reduction as allowed by `Vtrim`. Due to transistor variations from pixel to pixel the untrimmed threshold distribution (Vtrim 0, trim bits 15) is rather broad, with an RMS of typically 6 Vcal DAC units and a non-Gaussian distribution that reflects geographical variations across the ROC. The goal of the trimming procedure is to sharpen the threshold distribution to about 1 Vcal DAC unit (50 e) and a mean value a low as possible, but staying clear by at least $6\,\sigma$ from the noise level. The dynamic range of the threshold DACs is rather large (except for digV2 ROCs at nominal analog current), so that thresholds from 1 ke to 10 ke can be reached, in 50 e steps. The trimming procedure thus starts with selecting a threshold target (e.g. 30 Vcal DAC units), and adjusting the DACs and bits to reach that. A second constraint can be derived from the threshold equation: a smaller value of Vtrim leads to a closer spacing of the trim bit steps and a sharper threshold distribution.

The trimming procedure starts by measuring the untrimmed threshold distribution and identifying one pixel with the highest and one with the lowest threshold (dead pixels are flagged and ignored).

3.3.1 Global threshold

`VthrComp` acts inversely: a smaller DAC setting gives a harder globalThreshold (higher in Vcal DAC units). VthrComp is determined from the lowest pixel in the untrimmed distribution, setting its threshold to the target value (and pulling all other pixels along): command `vthrcomp` target.

Changing the threshold may influence the timing of the comparator, so CalDel should be checked and adjusted (command `caldelroc`).

3.3.2 Vtrim

The pixel with the highest threshold in the untrimmed distribution is used to set Vtrim, since it needs the largest correction. Its trim bits are set to 0 for maximum effect and Vtrim is increased until the target threshold is reached (command `trim target`)

3.3.3 trim bits

The trim bits are set in five iterations in the same trim command. First, all trim bits are set to 7 (half way) and a threshold map is taken. Many pixels may already be in the noise and don't respond; this is recognized and their trim bits are increased again in subsequent steps. For the others, the measured threshold is compared to the target, and the trim bits are adjusted in steps of 4, 2, 1, and 1 units, with the appropriate sign. A final threshold map should be taken for documentation (command `thrmap guess`, where `target` is a good guess).

3.3.4 efficiency check

The trimming procedure requires only 50% response for a valid threshold measurement. Some pixels apparently end up too close to the noise and require some further trim bit adjustment. An efficiency map with e.g. 100 triggers per pixel is taken and all pixels below 100% are inspected. Their trim bits are increased in steps of one until 100% response or end of range at 15 is reached (command `trimbits`).

The trim bits can be written to an ASCII file (`trimParameters_chip.dat`) with the comand `wtrim chip`.

3.4 pulse height tuning

Adjust gain and offset such that all pixel pulse heights fit into the ADC range, for large and small Vcal.

4 offline processing

The ROOT and log files from some tests are used for further processing and analysis.

4.1 gain calibration

Pulse height gain and offset varies from pixel to pixel. For best position resolution (using charge information) the variation should be calibrated out (can we quantify this? test beam analysis with the raw pulse height!). The calibrated pulse height distributions allow monitoring of the threshold and of the sensor charge collection efficiency in beam data.

The gain calibration starts from scans of pulse height vs test pulse amplitude, in low and high range:

ctl 0 small Vcal

dacscanroc 25 nTrig measure pulse height vs Vcal for each pixel, nTrig = 10 takes about 90 s, filling a 2D histogram `PH_DAC25_CR0_map`

ctl 4 large Vcal

dacscanroc 25 nTrig filling `TH2D PH_DAC25_CR4_map`

`mv Test.root phroc-c405-Ia25-trim30.root`

The gain calibration varies with several dacs (Vdig, Vana, Vsf, Vrg, Vtrim, VthrComp, PHOffset, PHscale) and with temperature.

It was found that the gain curve (PH vs Vcal) of digital ROCs is well described by a Weibull distribution function:

$$f = p_4 - p_3 \exp(-t^{p_2}),\ t = p_0 + x/p_1$$

where x is the test pulse amplitude (Vcal DAC) and f the measured pulse height [ADC]. $p_4$is the asymptotic pulse height (in the saturation region), $p_3$is the dynamic range from zero to saturation and the rest are shape parameters which can be given an interpretation by looking at the derivatives:

$$f' = p_3 p_2 t^{p_2-1} \exp(-t^{p_2})/p_1,\ f'' = -p_3 p_2 \exp(-t^{p_2}) \left((p_2-1)t^{p_2-2} - p_2 t^{p_2-1}\right)/p_1^2$$

f has an inflection point where f' has a maximum and where f''has a zero, namely at $t_{inf} = ((p_2-1)/p_2)^{1/p_2}$. The maximum gain is then $f'(t_{inf})$. It turns out that $p_0$is always very close to one (like 0.9998). It is thus tempting to reduce the number of parameters by setting $p_0$to one. Furthermore, $p_1$turns out the rather large (10^5in Vcal DAC units), inviting one more approximation:

$$t^{p_2} \approx (1 + x/p_1)^{p_2} = \exp\left(p_2 \ln(1 + x/p_1)\right) \approx \exp\left(p_2 x/p_1\right)$$

leading to a double exponential

$$f \approx p_4 - p_3 \exp\left(-\exp(p_2 x/p_1)\right)$$

which is known as the Gompertz function. It describes the transition towards saturation quite well but has slope zero at $x = 0$ while the gain curve rises almost linearly from threshold. We use the Weibull fit (the tanh fit used for analog ROCs does not give a good discription of the turn-over towards saturation; it is too sharp).

The fit is done simultaneously to the low Vcal (x_0) and high Vcal(x_4) range measurements using one more parameter for rescaling Vcal: $x_0 = x_4/p_5$, where p_5 is around 7.

The fit is numerically problematic as not only do the parameter values range over several orders of magnitude (which could be cured by rescaling) but also their precisions. This is reflected in a huge condition number (10^{16}) for the Hessian matrix (which upon convergence is the inverse of the covariance matrix of the fit parameters). Convergence depends crucially on the start values for the parameters. However, we want to perform 10^8fits automatically but successfully. Migrad (from Minuit) may converge for 95% to 99% of the pixels which is not good enough. A modern quadratic approximation optimization algorithm (BObyQA) or gradient based algorithms (the derivatives of f with respect the parameters p_i are analytic) like L-BFGS do not fare better. The best performance was found with the good old Nelder-Mead simplex algorithm, which reaches 100% convergence and finds the same χ^2minimum as the other algorithms in the cases where they all converge.

In data analysis, and also for gain monitoring in psi46test, the inverse function is needed to translate a measured pulse height a from ADC counts into Vcal DAC units:

$$f^{-1} = p_1 \left(\left(- \ln \left((p_4 - a)/p_3 \right) \right)^{1/p_2} - p_0 \right)$$

which is nicely analytic but reveals one problem: the argument of the logarithm must be positive, thus the measured pulse height a must never fluctuate above the asymptotic value p_4. A protection is put in place, leading to an artificial peak at large pulse heights (but reflecting the loss of pulse height sensitivity in the saturation region).

4.2 S-curves

S-curve is descriptive pixel slang for threshold curves as obtained from counting the number of pixel responses to a given number of triggers N as a function of a dac. Each response count n_i is drawn from a binomial distribution with unknown success probability. The fit involves a model for the success probability as a function of the dac. When the width of the threshold is governed by noise, a Gaussian error distribution ranging from 0 to 100% is well justified. In the presence of non-Gaussian tails one may try a Student's t distribution. A general threshold can often be parametrized by a Fermi function (which is equivalent to a tanh function). In all cases, the quoted threshold is defined as the dac value where 50% efficiency is reached. In this way, the threshold can also be determined without fitting, just by scanning the data curve, as is done in the FPGA. The width of the S-curve can be determined from the 10% to 90% range, which is $2.56\,\sigma$ for the Gaussian error distribution.

4.3 bump bond test

The ROC has a switch (`cals`) on each pixel which allows to send the test pulse to a pad on the top metal layer. When a sensor is present it forms a (small) air gap capacitance and some charge gets induced. When the bump bond is functional the pixel circuit amplifies the signal and if the threshold is sufficiently low it can be detected and read out. Two tests are available:

bbtest nTrig fast map of all pixels' responses to `cals` pulses, done with the largest pulse (`CtrlReg` 4, `Vcal` 255) but with fixed threshold settings.

dacscanroc 12 -nTrig scan global threshold `VthrComp` while pulsing through the sensor (selected with negative nTrig). Should be done with large pulses (`CtrlReg` 4, `Vcal` 255).

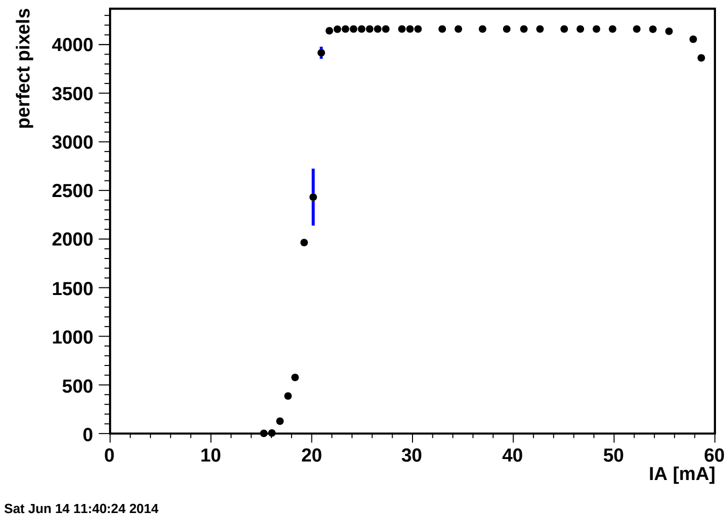


Figure 1: perfectly efficient pixels vs analog current. 22 mA is required to reach the plateau. At large current the effective threshold is too high

5 plots

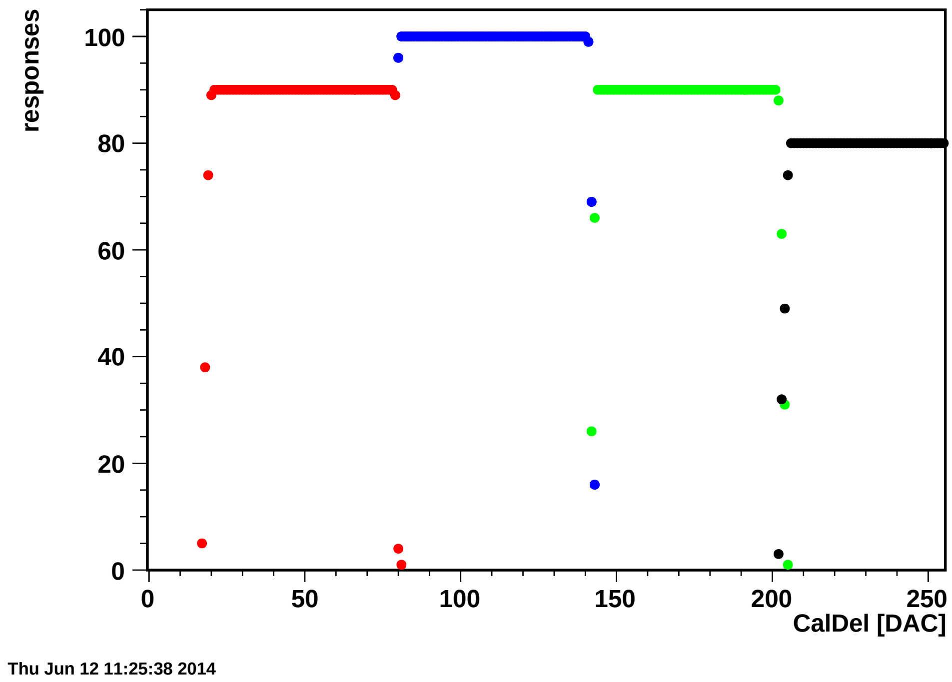


Figure 2: Responses vs `CalDel` for one pixel. tct 106. `WBC` 100 (red, 90 triggers), `WBC` 99 (blue, 100 triggers, working point), `WBC` 98 (green, 90 triggers), `WBC` 97 (black, 80 triggers).

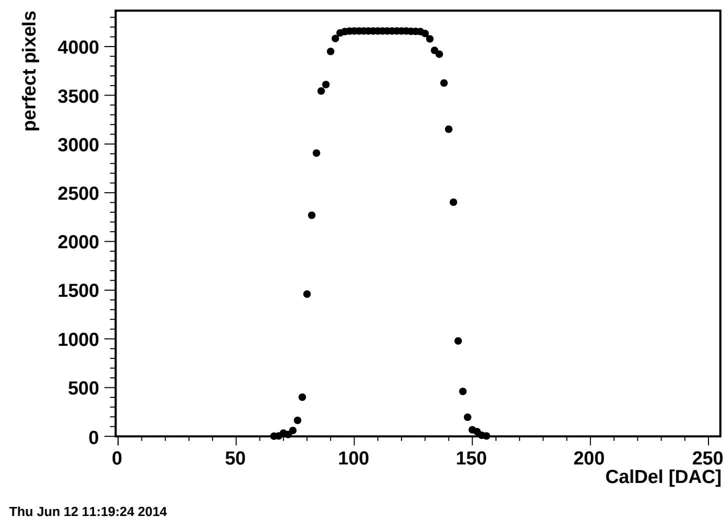


Figure 3: `CalDel` scan for an entire ROC counting the number of fully responding pixels at each point at large `Vcal`. The working point is set on the plateau towards the left edge, to allow for timewalk at smallest pulse heights.

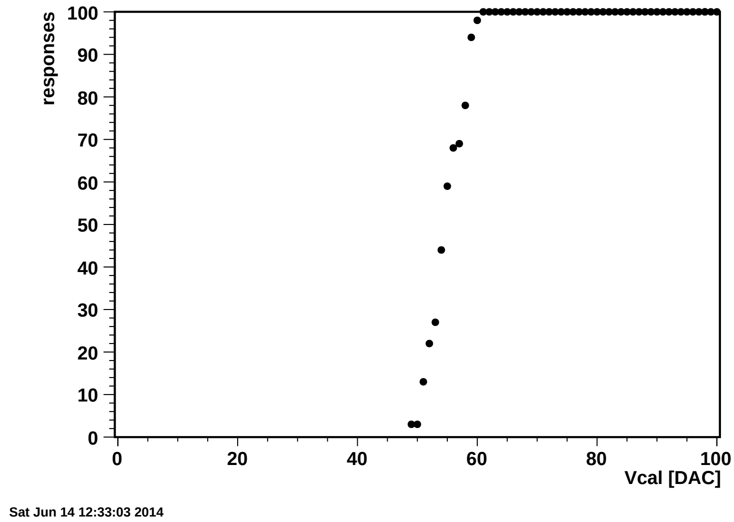
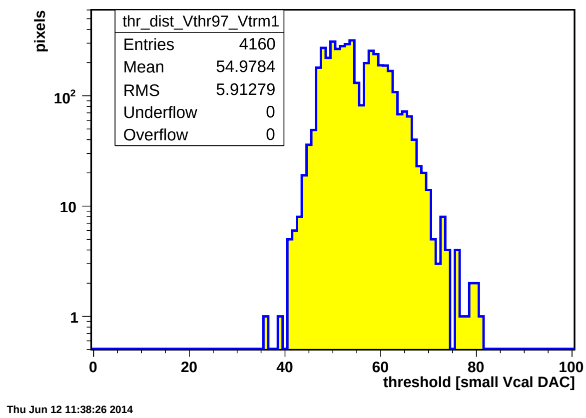


Figure 4: Single pixel S-curve: counting responses to 100 triggers as a function of the test pulse amplitude (in low range). The threshold is defined as the Vcal value where 50% efficiency is reached. The width of the curve is taken as a measure of the noise.



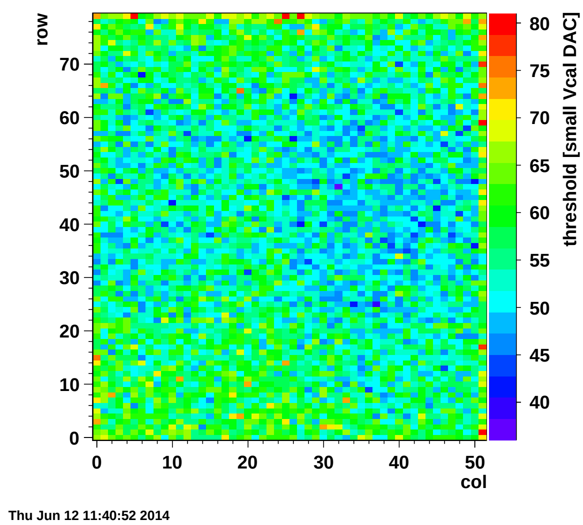


Figure 5: threshold distribution and map, untrimmed digV2.1 chip

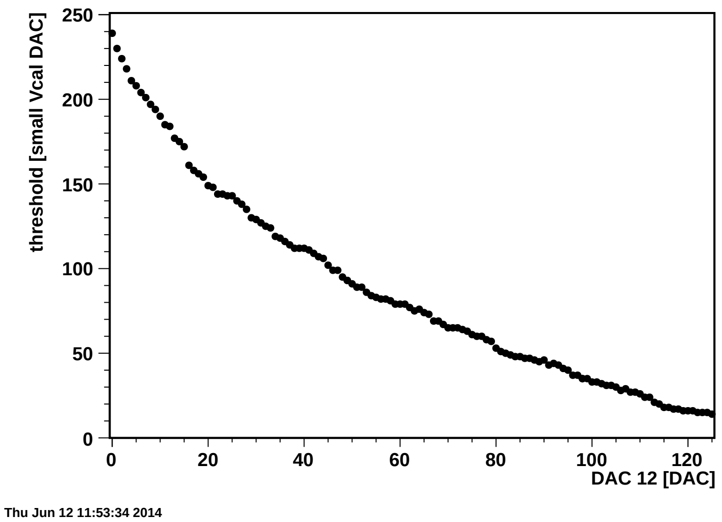


Figure 6: pixel threshold vs global threshold

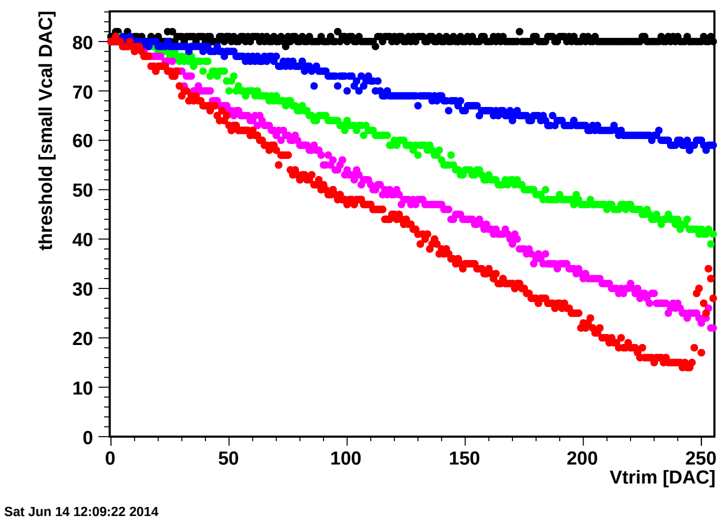


Figure 7: Pixel threshold vs `Vtrim` for different `trim bits`: top (black): 15, 2nd (blue): 11, mid (green): 7, 4th (magenta): 3, bottom (red): 0 (at large `Vtrim` and and trim bits 0 the threshold approaches the noise level and the measurement becomes unreliable). The trim bit spacing is closer at smaller `Vtrim`, potentially leading to a sharper threshold distribution.

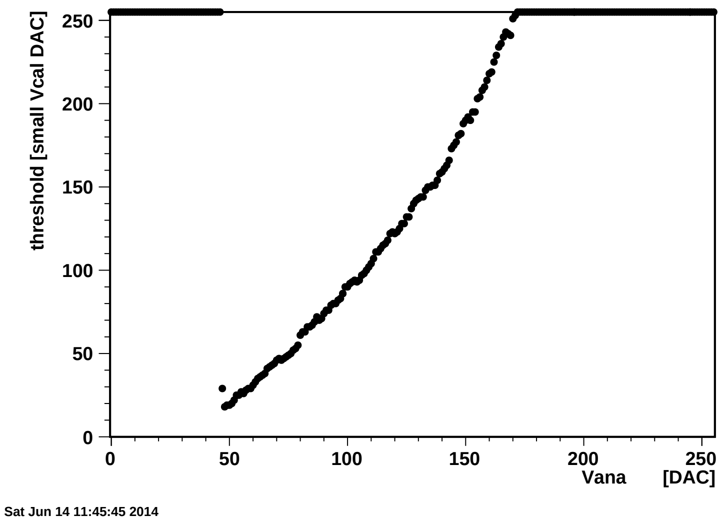


Figure 8: Pixel threshold vs `Vana`. Threshold 255 means overflow (current too low or threshold too high). Changing `Vana` changes the working point of preamplifier and shaper and the baseline at the input to the comparator, thus changing the pixel threshold with fixed comparator settings (`VthrComp`, `Vtrim`). The order of setting the DACs matters: don't change `Vana` after trimming (or re-trim, or at least take a threshold map).