

PSI43 Testing at Fermilab

Hardware

Refer to <http://www.physics.rutgers.edu/~bartz/cms/> for more information

- Linux workstation
 - CPU: 1.7 GHz Celeron
 - Memory: 512 MB
 - System bus: 400 MHz
 - Disk: 37.27 GB
 - Partitioned to boot system with Windows and Linux
 - 10/100 Mb/s Ethernet
 - CD-ROM
 - 3.5" floppy
 - Graphics card: Needed a graphics card for which a driver was available in Red Hat Linux 7.1. Could not use the ATI Radeon card that came with the system. Replaced it with a Cirrus Logic 5434 PCI card (very old one)

- HDI adapter board with VHDI and one or more PSI43 chips attached to the VHDI. The VHDI is wire-bonded to the HDI
- FPGA based TBM board
- FEC board (very basic, temporary)
- LEMO and 34-wire cables
- A 2 channel digital scope with 500 MHz analog bandwidth: Tektronix TDS 3054B
- 5V/3A power supply: Tektronix PS2521G (programmable)
- TTL pulse generator (for clock): LeCroy 9210 (40 MHz)
- TTL pulse generator (for triggering): 1340 Pulse generator (General Radio) (100 KHz)
- A 20 MHz ADC card with 12 bit resolution (PCIDAQ 2012) plugged into a PCI slot. Card has 5 input ports, 4 for A/D signals and 1 for CLK.

Data from the data channel (of TBM) was connected to ADC channel 0 (J1), and the clock channel (of TBM) was connected to ADC channel 4 (J5).

No special hardware settings were required. Used default settings.

Software

Refer to <http://www.physics.rutgers.edu/~bartz/cms/> for more information

- Red Hat Linux 7.1 (obtained in CD from John Doroshenko in Rutgers)
- Kernel version is 2.4.2-2
- Glibc version 2.2.2
- Obtained Cosmo software (from Rutgers). Use the latest
- Obtained Cyber Research's PCI DAQ 2000 ADC card driver from John and installed it as instructed

```
cd /root/drivers
./dask_inst.pl
```

The driver for the ADC has to be reloaded whenever the system is rebooted.

Executing the command `/sbin/lsmmod` displays the modules **p9812** and **adl_mem_mgr**, indicating that the driver has been installed.

Follow John's directions in the Rutgers web-site to install the Cosmo software.

Connecting the System

- Follow Ed Bartz's instructions in the Rutgers web-site closely.
- TBM connections:
 - Connect Data (out) to channel 1 of the scope
 - Connect 34-pin flat ribbon cable to the FEC board
 - Connect the wire leads to the power supply.
 - Set the Hub address (5 bits) to 0
 - Set the TBM address (bit 6) to 0
 - Set the Loop Token (bit 8) to **0 (on) when there is no readout chip (ROC) connected. Set this to 1 (off) when there is a chip.**
- FEC connections
 - Connect the 34-pin flat ribbon cable to the TBM
 - Connect to the parallel port of the PC
 - Connect port on FEC labeled CLK to the pulser supplying the clock signal. Pulse width was 25 nsec, period was 50 nsec (20 MHz)
 - Connect port on FEC labeled SCAL to the pulser supplying trigger signal level 0 to 3.5 V.

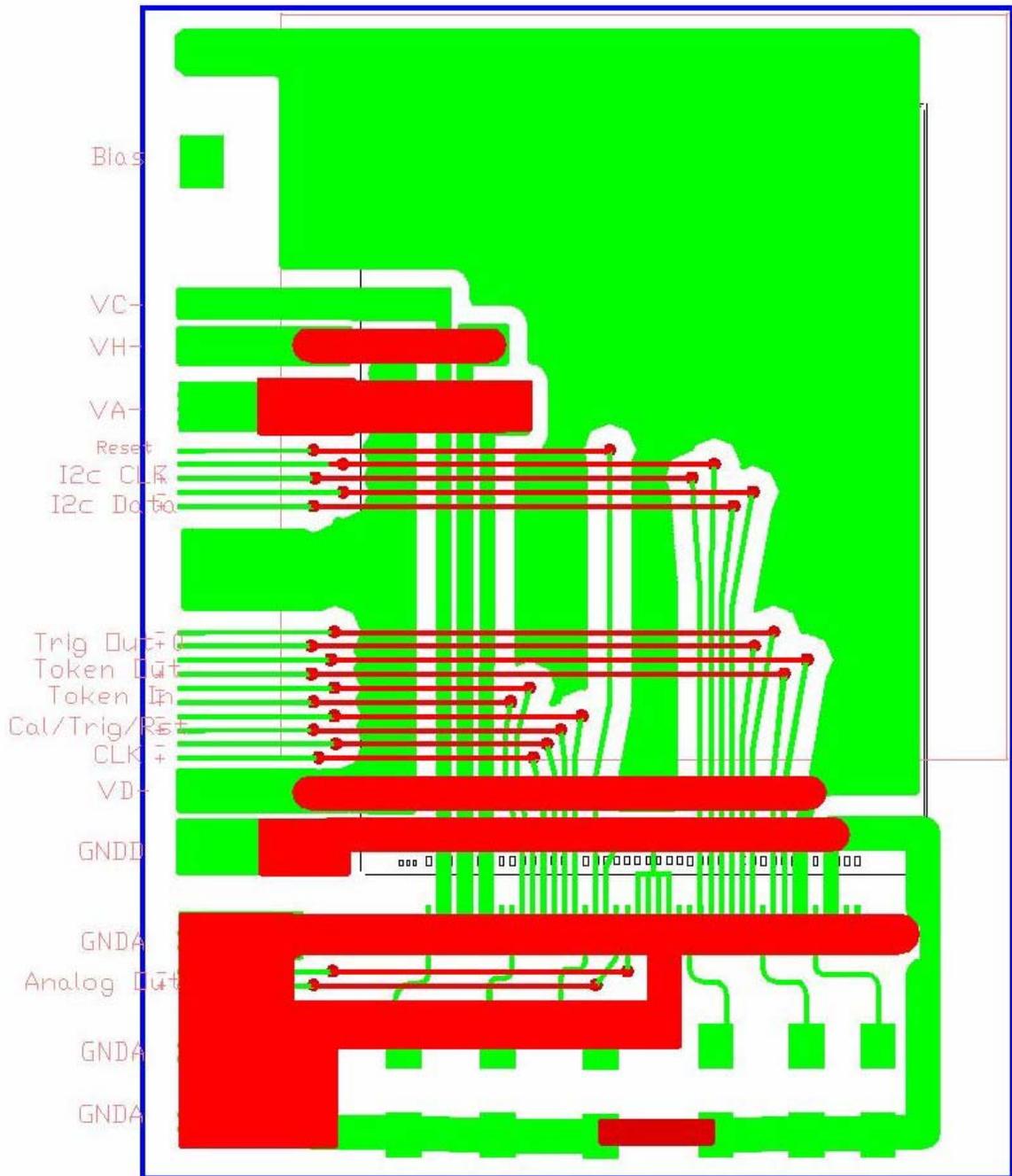


Figure 2 *VHDI layout*

- Define the horizontal axis to be the X-axis and the vertical the Y-axis
- Take the lower left hand corner to be the origin (0,0)

- The chip (as shown in the pin out diagram) is connected to the traces along the X-axis
- The VHDI traces along the Y-axis are connected to the HDI

Chip to VHDI Connections (along the X-axis direction)

The chip is placed on the VHDI with the topmost pad (as shown in the above layout) connected to the VHDI trace farthest out along the X-axis

In the following tabulation, start from the outermost (farthest to the right) trace along the X-axis of the VHDI layout and the top pad of the CMS pixel chip layout. All chip pads designated Cap are connected to capacitors on the VHDI.

Coordinate reference: pad 1 is the bottom pad Cap(C) in the figure showing the pin layout of the chip. The corresponding trace is that closest to (0,0) along the X-axis.

• GNDA	2 thin traces (#1 and #2)	pad 1, 2
• GNDD	2 thin traces (#3 and #4)	pad 3, 4
• Cap (DAC)	1 thin trace (#5)	pad 5
• VD-	2 thin traces (#6, #7)	pad 6, 7
• Token_in -	1 thin trace (#8)	pad 8
• Token_in +	1 thin trace (#9)	pad 9
• Cap (Dig)	1 thin trace (#10)	pad 10
• Trig_out -	1 thin trace (#11)	pad 11
• Trig_out +	1 thin trace (#12)	pad 12
• I ² C_data -	1 thin trace (#13)	pad 13
• I ² C_data +	1 thin trace (#14)	pad 14
• I ² C_clk -	1 thin trace (#15)	pad 15
• I ² C_clk +	1 thin trace (#16)	pad 16
• Cap (LVDS)	1 thin trace (#17)	pad 17
• V_iref	1 thin trace (#18)	pad 18
• I ² C_a ₀	1 thin trace (#19)	pad 19
• I ² C_a ₁	1 thin trace (#20)	pad 20
• I ² C_a ₂	1 thin trace (#21)	pad 21
• I ² C_a ₃	1 thin trace (#22)	pad 22
• A _{out} -	1 thin trace (#23)	pad 23
• A _{out} +	1 thin trace (#24)	pad 24
• GNDD	1 thin trace (#25)	pad 25
• Reset	1 thin trace (#26)	pad 26
• Cap(H)	1 thin trace (#27)	pad 27
• Cal_trig_res -	1 thin trace (#28)	pad 28
• Cal_trig_res +	1 thin trace (#29)	pad 29
• Clk -	1 thin trace (#30)	pad 30
• Clk +	1 thin trace (#31)	pad 31

- Token_in - 1 thin trace (#32) pad 32
- Token_in + 1 thin trace (#33) pad 33
- Cap(A) 1 thin trace (#34) pad 34
- V_{vref} 1 thin trace (#35) pad 35
connected to 1.6 Meg ohm resistor
with a 15 pF capacitor in parallel
- VA - 2 thin traces (#36,#37) pad 36, 37
- VH - 2 thin traces (#38,#39) pad 38, 39
- VC - 2 thin traces (#40,#41) pad 40, 41
- Cap(C) 1 thin trace (#42) pad 42 (bottom pad in chip)

VHDI to HDI Connections (Y-axis of the VHDI trace map)

Start from low y to high y in the VHDI layout

- GNDA width 60 trace #1
- Analog_out + width 5 (thin) trace #2
- Analog_out - width 5 trace #3
- GNDA width 20 trace #4
- GNDD width 20 trace #5
- VD - width 20 trace #6
- Clk + width 5 trace #7
- Clk - width 5 trace #8
- Cal_trig_rst + width 5 trace #9
- Cal_trig_rst - width 5 trace #10
- Token_in + width 5 trace #11
- Token_in - width 5 trace #12
- Token_out + width 5 trace #13
- Token_out - width 5 trace #14
- Trig_out + width 5 trace #15
- Trig_out - width 5 trace #16
- VD - width 50 trace #17
- I²C_data + width 5 trace #18
- I²C_data - width 5 trace #19
- I²C_clk + width 5 trace #20
- I²C_clk - width 5 trace #21
- Reset width 5 trace #22
- V_A - width 20 trace #23
- V_H - width 20 trace #24
- V_C - width 20 trace #25
- Bias width 25 trace #26

The address lines $I^2C_a_0 \rightarrow I^2C_a_3$

- On the VHDI, the traces are all connected to ground
- In the negative powered CMOS ground is high (1) and float is low (0)
- To set a 1x1 chip to address 6, wire-bond the address lines $I^2C_a_1$ and $I^2C_a_2$ to the VHDI, i.e. to ground.

Bias

- The trace labeled bias on the HDI/VHDI interface is meant for sensor high voltage. Remember, all the signals and voltages to/from the pixel/sensor system go through the HDI/VHDI.

Chips in a 1x5 plaquette

- In a 1x5 module (plaquette), all I^2C address pads have to be wire-bonded to the VHDI for all 5 chips.
- When configured in this mode in a 1x5 module, the first chip in the module has an I^2C address 0 by default, the second is 1, the third is 2, the fourth is 3, and the fifth is 4

Column and Row Address Levels

There are overall 5 analog output levels defined in the PSI43 chip

- UBLK (ultra black) -400 mV (differential)
- Level 0 -100 mV (differential)
- Level 1/BLK (black) 0 mV (differential)
- Level 2 +100 mV (differential)
- Level 3 +200 mV (differential)

The row and column addresses are analog encoded digital signals with 5 signal levels. Numerical values are computed using base5 arithmetic.

Columns

- The number of double columns is 26
- Column address has two base5 digits
- A column address representation will be of type C_1C_0
- C_0 is the least significant digit and can take values 0 through 4
- C_1 is the most significant digit and can take values $0x5^1$ through $4x5^1$
- These two digits can address only 25 columns, 0 through 24.
- An additional signal level has been created in the most significant “digit” (Level 5) to indicate the last column (column 25, numbering scheme: 0 to 25)

Rows

- There are 106 rows within a double column
- A row address is allocated 3 base5 digits.
- A row address representation will be of type $A_2A_1A_0$
- A_0 is the least significant digit and can take values 0 through 4.
- A_1 is the next least significant digit and can take values $0x5^1$ through $4x5^1$
- A_2 is the most significant digit and can take values from $0x5^2$ through $4x5^2$

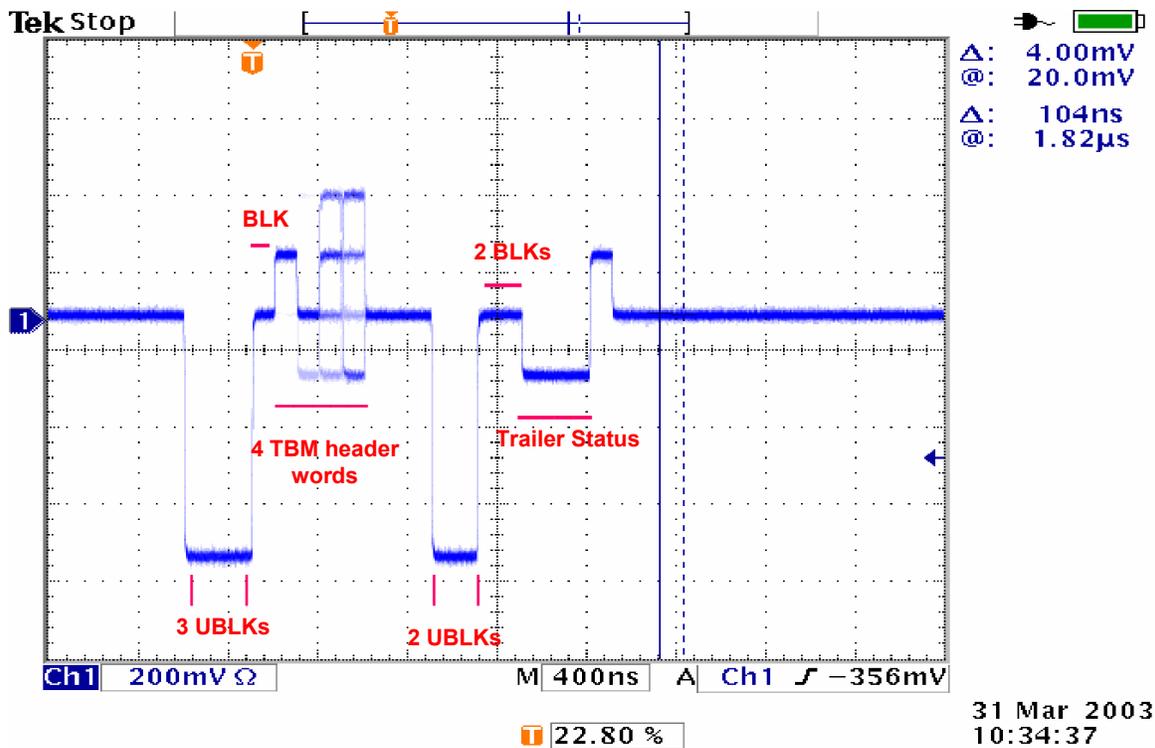


Figure 3 TBM Loop Around Mode

The above figure is a trace of the TBM running without a chip. It contains a header and a trailer.

TBM Header

- The header consists of
 - 3 UBLKs (3 CLKs)
 - 1 BLK (1 CLK)
 - 4 TBM header words (4 CLKs)
- The TBM header has 4 levels - 0 (-100 mV), 1 (0 mV), 2 (+100 mV), 3 (+200 mV)
- Hence each TBM header word can encapsulate 2 binary bits (4 levels)
- The 4 header words can contain 8 bits worth of data
- These 4 header words will normally contain an 8-bit event number

TBM Trailer

- The Trailer consists of
 - The trailer marker consisting of
 1. 2 UBLKs (2 CLKs)
 2. 2 BLKs (2 CLKs)
 - Trailer Status words (4 CLKs)

The PSI43 data read out is encapsulated in a packet by the TBM as follows

1. TBM header (as described above)
2. Data read out by the TBM
3. TBM trailer (as described above)

Data from a chip is read out in the following order

- One header for each chip consisting of an UBLK, BLK, and last DAC (3 CLKs)
- Address of the pixel hit (5 CLKs) consisting of
 1. C1 double column address, MSD, 5+1 levels
 2. C0 double column address, LSD, 5 levels
 3. A2 row address, MSD, 5 levels
 4. A1 row address, nMSD, 5 levels
 5. A0 row address, LSD, 5 levels
- Data (1 CLK) pulse height

A typical pixel data read out from a chip appears as shown in the following trace. It contains the TBM header, chip data, and the TBM trailer. In this case the pixel(0,0) in double column 0 and row 0 is being read out. The chip is running at 10 MHz.

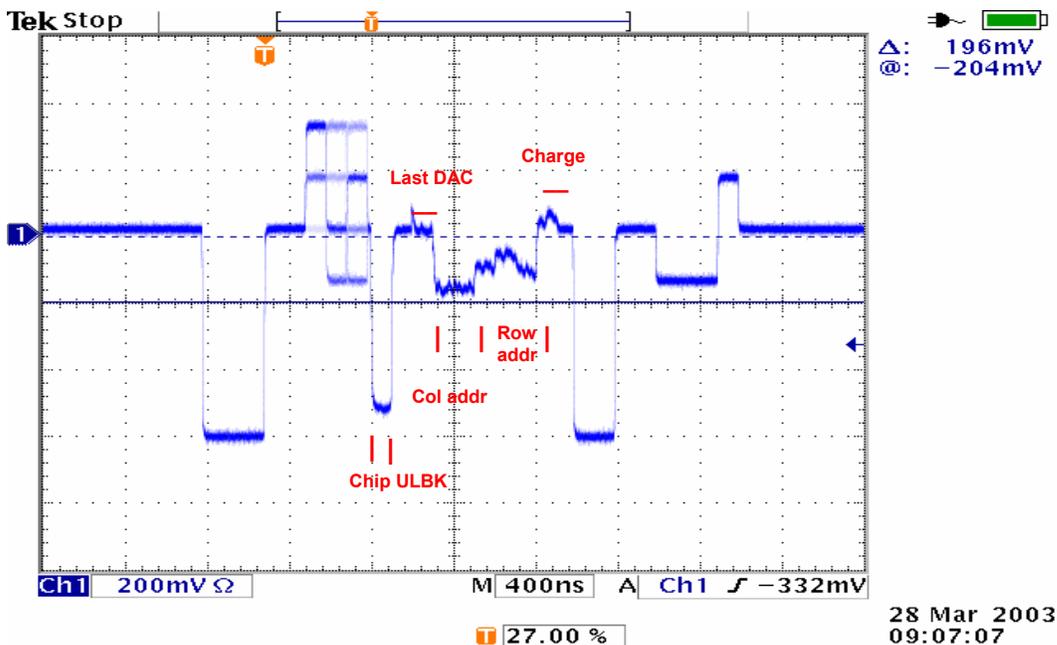


Figure 4 Pixel (0,0) Calibration Signal

If multiple pixels from a chip are read out by a TBM, the data will flow out in the following order

TH U B DAC C1 C0 A2 A1 A0 D C1 C0 A2 A1 A0 D . . . C1 C0 A2 A1 A0 D TT

TH TBM header (8 CLKs)

TT TBM trailer (8 CLKs)

DAC Last DAC (1 CLK)

All other signals are 1 CLK each

Multiple pixel signals from a single chip appear as shown in the following traces. The scope trace in Figure 1 represents 5 consecutive pixels - 0,1,2,3, and 4 – in double column 0 being read out during calibration. Figure 2 is a scope trace of 5 consecutive pixels, double columns 0 through 4, in row 0. Notice that the UBLK is present only before the first pixel readout of a chip. When pixels from different chips are read out, as illustrated in the trace in figure 3, data from each different chip is preceded by an UBLK, BLK, and DAC.

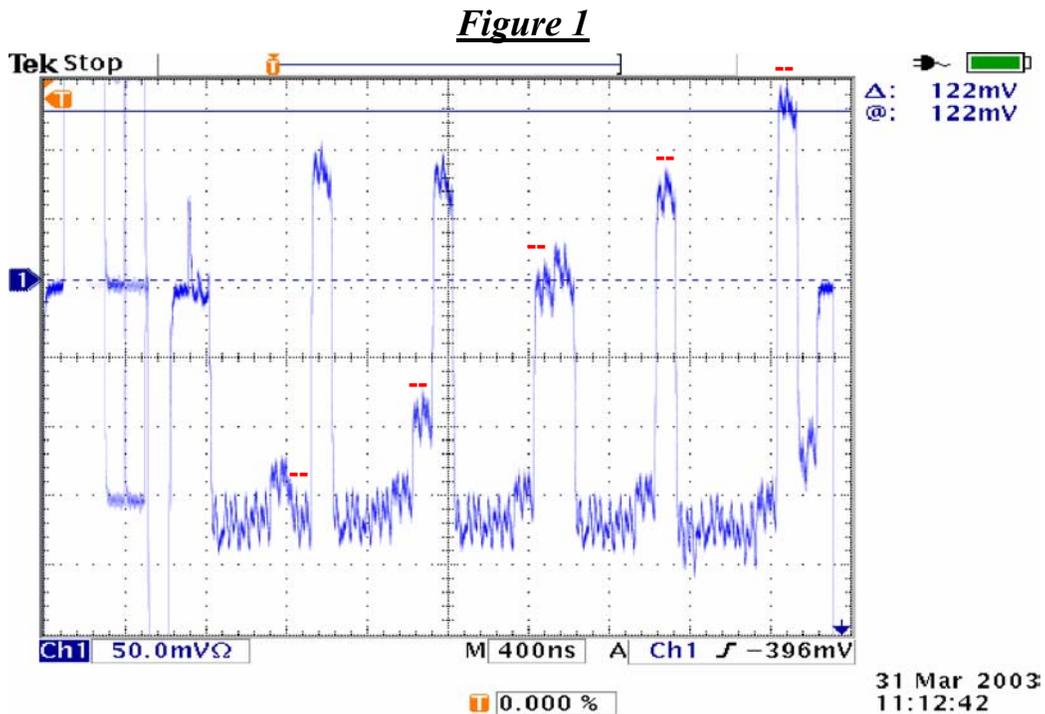


Figure 5 A_0 : Pixels (0,0) (0,1), (0,2), (0,3), (0,4)

Figure 6

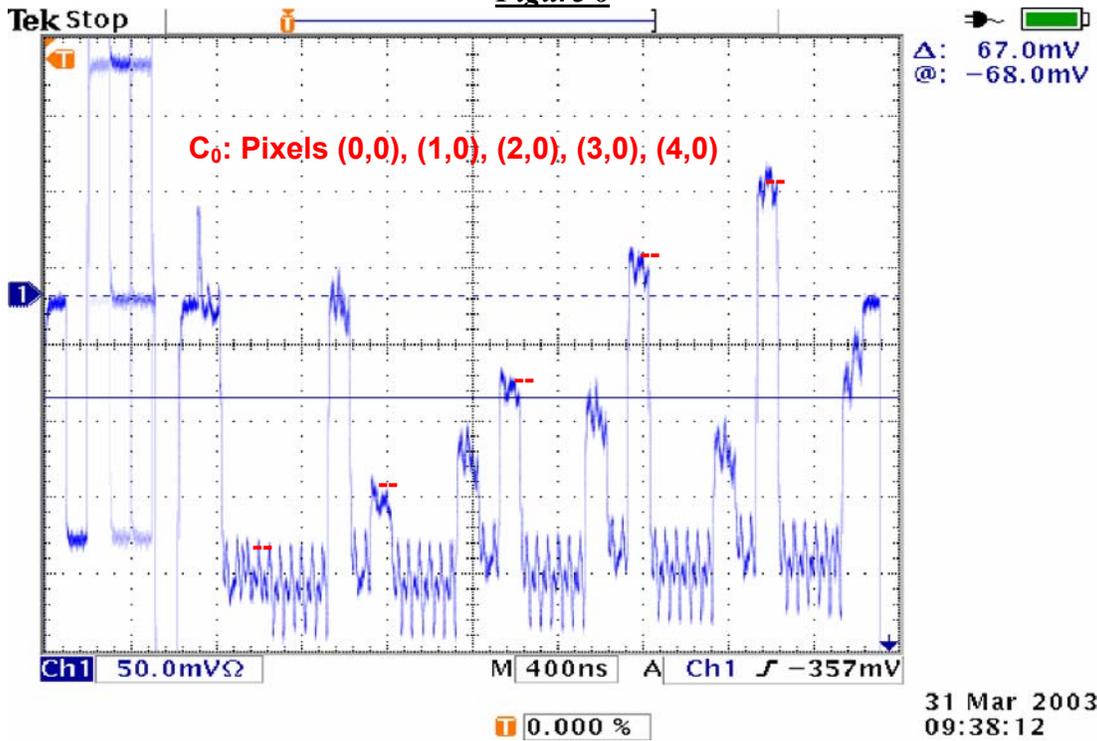
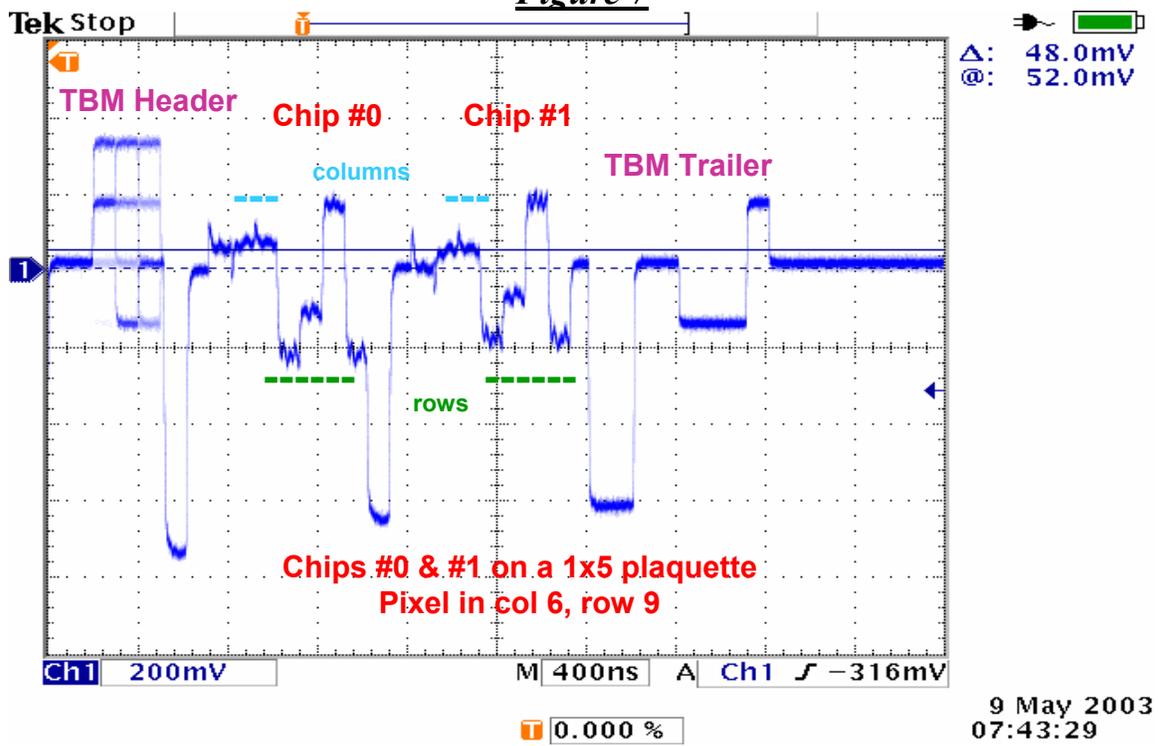


Figure 7



An important indicator of whether a chip is good or bad is the current it draws when powered up by the setup with one PSI43 chip. A sample current drawn is

- 0.544 amp @ +5 V
- 0.239 amp @ -5 V

The presence of a readout chip UBLK signal indicates that the chip is “not dead”. The change in the last DAC setting when a DAC is altered indicates that the I2C interface of a readout chip is functional. In the absence of response from the chip, these signals acquire some degree of importance in troubleshooting.

1x1 Chip Test

- In a 1x1 chip test, we found the scope trace to be very unstable. A lot of jitter in the signal.
- Putting a 15 pF capacitor in parallel to the 1.6 MΩ resistor connecting V_{vref} to ground stabilized the signal
- When testing the chip, you may not see any signal when turning the power on. You may have to adjust the various DAC settings until you get a signal trace.
- Once you get a signal, save the DAC settings to a file for future use.
- The levels of the row and column addresses, together with that of the signal level, can be adjusted with different DACs. In our adjustments, we first adjusted the PUC (pixel unit cell) DAC settings – V_{rgPr} , V_{rgSh} , V_{trim} , and V_{hldDel} . We followed this with adjustments to the DAC settings related to the address levels – V_{Bias} , $V_{\text{Bias}_{\text{sf}}}$, V_{Cas} , V_{OffsetOP} , V_{BiasOP} , V_{OffsetRO} , $V_{\text{IBiasAachen}}$, V_{OffsetCA} , V_{BiasAddr} , V_{Ion} , V_{Ioff} , V_{BiasOff} .
- V_{OffsetOP} (V-): increasing decreases offset of row address and analog signal; double column address unaffected.
- V_{OffsetCA} (V-): affects double column address offset
- V_{OffsetRO} (V-): affects offset of row, double column, and signal.
- $V_{\text{IBiasAachen}}$ (μA): affects offset levels of double column, row, and signal
 determines total current of output stage.
 Total current $\sim 10 \times V_{\text{IBiasAachen}}$
 Higher values add a DC offset to output
 Lower values result in an insufficient output range
- $V_{\text{Bias}_{\text{sf}}}$ (μA): increasing value decrease offset of row address and analog signal
 double column address levels unaffected.
- V_{BiasOp} (μA): increasing raises offset of row address and analog signal; no effect on double column address
- V_{BiasAddr} (μA): affect double column address; no effect on row address and signal levels.
- V_{IonAddr} (μA): affects double column offset level; no effect on row address and signal.

- We were informed by Roland Horrisberger that the procedure should be to adjust the double column address first, followed by the row address, and then the PUC settings.

Summary:

- Double column address (only) affected by:
 V_{BiasAddr} , V_{OffsetCA} , and V_{IonAddr}
- Row address and signal affected by:
 $V_{\text{Bias_sf}}$, V_{BiasOp} , and V_{OffsetOP}
- All three (double column, row, and signal) affected by
 V_{OffsetRO} and $V_{\text{IBiasAachen}}$

When testing the chip the clock signal (into FEC) was 20 MHz (50 ns)

The control register (CTRL-REG) on the readout chip has 2 bits

- Bit 0: readout speed
- Bit 1: stop data acquisition
- If bit 0 is set to 1, the readout speed is the same as the input clock speed. In this case it is 20 MHz.
- If bit 0 is set to 0, the readout speed is $\frac{1}{2}$ of the input clock speed, i.e. 20 MHz if the clock is 40 MHz.
- In the program Cosmo, clicking on 40 MHz sets this bit to 1, and clicking on 20 MHz sets the bit to 0.

If bit 0 is set to 0 and the input clock is 20 MHz, the readout speed is 10 MHz.

1x5 Module Test

- The first chip was glued, wire-bonded, and tested. In a 1x5 module the first chip is always set to I2C address 0 and the last to 4 irrespective of how the I2C address pads are connected.

The V_{ref} pad of chip 0 was wire-bonded to the VHDI. This, in turn, was connected to a 1.6 M Ω resistor and 15 pF capacitor in parallel.

For chips 1 through 4, V_{ref} of chip was connected to V_{H} of the VHDI, i.e. pad 8 of ROC was connected to pad 16 of the VHDI.

When only one chip is present, the token_out of one chip has to be connected to the token_in of the next chip. For example, when chip 1 is added, the connection from token_out of chip 0 to token_in of chip 1 has to be removed.

- Chip 1 was then added to the VHDI and wire-bonded. Measurements were carried out for both chip 0 and chip 1.
- With each additional chip, measurements were repeated for all chips present on the VHDI.

General Test Procedure

The following is a general procedure for starting up a chip

1. Power up the setup
2. Download DAC settings from the appropriate file
3. Set hub id to 0, port id to 0, and I2C address to 6
4. Perform a ROC init after the chip is powered up with calibration off and pixels disabled
5. Reset the System
6. Set Readout speed to 20 MHz.
7. Set mode to CAL
8. Set TRIM settings to 3
9. Enable/disable a double column by clicking on the double column desired
10. Enable/disable pixel by clicking on the pixel desired
11. Click the ROC Reset button to inject a calibration pulse. You should see a trace on the scope.
12. In the ADC menu of Cosmo, use the *select filename* option to name and open a data file.
13. In the ADC tools option, do the following:
 - Select Load Cosmoscript option
 - Select binary mode data
 - Click on the “**Press to run VCal...**” button to start taking data
14. The program goes through 32 steps (Vcal voltage levels) taking 500 data points per level. A total of 16000 data points is collected.
15. Close the data file using the ADC menu.
16. Suppose the name of the file is *Roc0_Pixel69_Run00_bin.dat* where Roc0 indicates it is chip #0, Pixel69 indicates it is a pixel in double column 6 and row 9, Run00 indicates that it is the first run for this pixel, and bin indicates data is in binary format.
17. An executable named **1x5** was run with the above binary data as input. An output file in ASCII format, *Roc0_Pixel69_Run00.output*, was produced.
18. Two Kumac files were created
 - *Roc0_Pixel69_Run00.kumac*
 - *Roc0_Pixel69_Cal00.kumac*
19. Run PAW and execute the command
 - *exec Roc0_Pixel69_Run00*
20. Follow this with the following commands to display various ROC related entities
 - *exec Roc0_Pixel69#ub* to display the UBLK page (Figure 1)
 - *exec Roc0_Pixel69#roc* to display ROC page (Figure 2)
 - *exec Roc0_Pixel69#cal* to display the 8 CAL pages (Figure 3 is one such page)
21. Collect (note) all the means of cal histograms and input them into the Cal kumac file and execute the PAW command
 - *Exec Roc0_Pixel69_Cal00*

22. This plots and fits a straight line. The intercept is the pedestal.
23. To print out postscript files of the plots, do the following from within PAW
 - Fort/file 10 ***Paw.ps***
 - Meta 10 -111
 - Execute Roc0_Pixel69_Cal00
 - Close 10
24. This produces a postscript file, ***Paw.ps***, containing the plots.

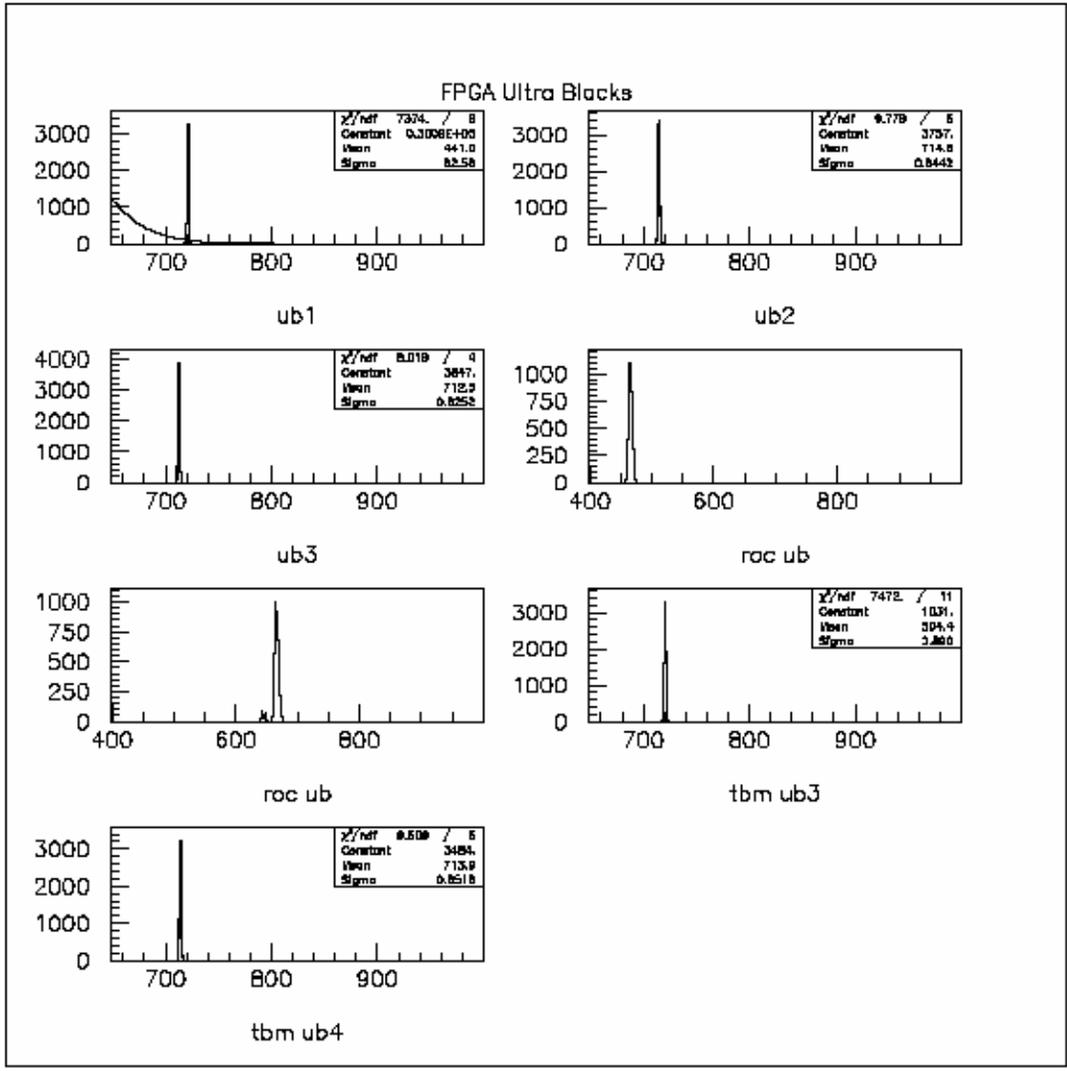


Figure 8 UB Page

The number of events is displayed along the vertical axis and the ADC counts along the horizontal axis. The first 4 plots display the UBLK's of the TBM header. This is followed by the UBLK's of the readout chips (in this case there are 2 chips), and the last 2 plots represent the UBLK's of the TBM trailer.

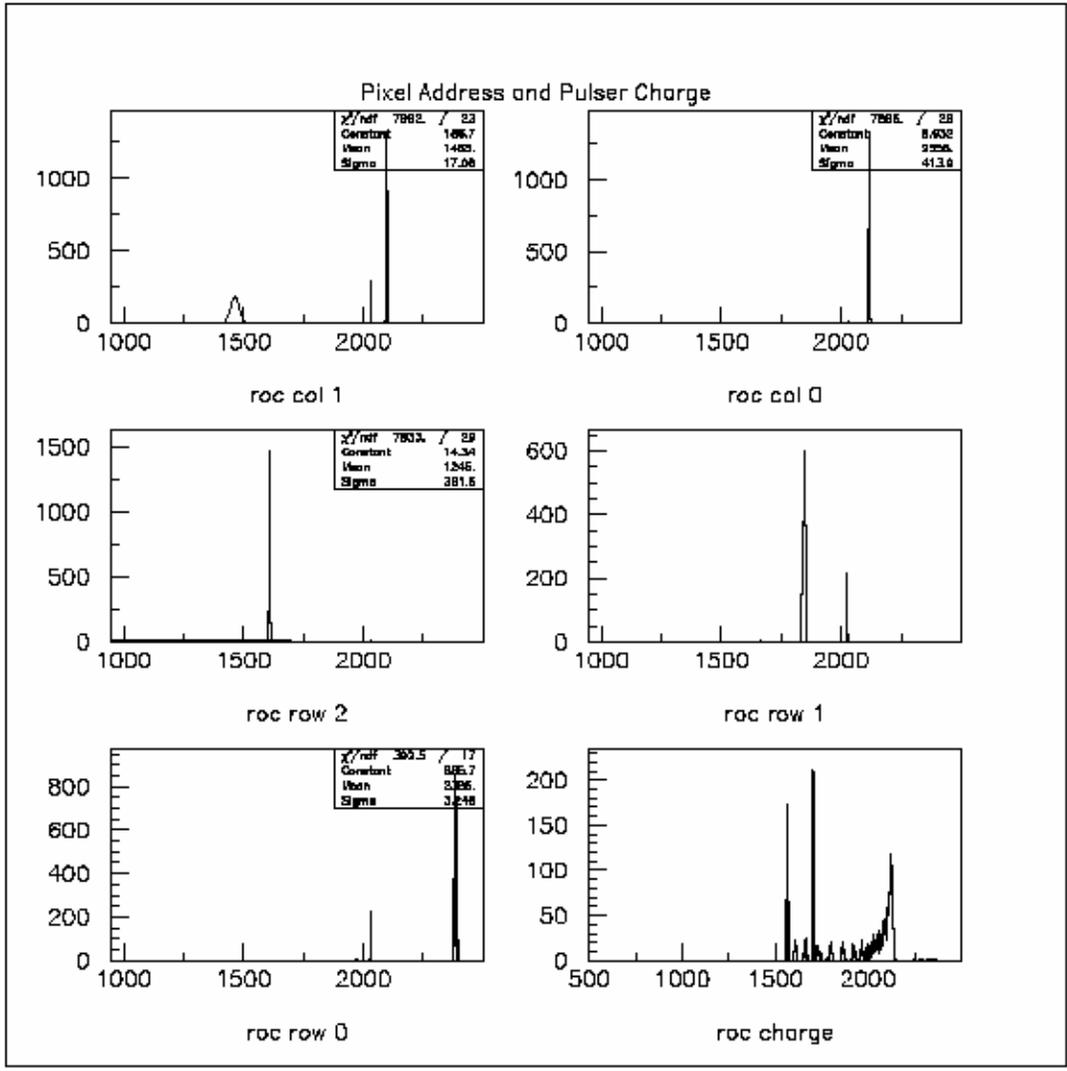


Figure 9 ROC Page

The first two plots represent the signal levels of C1 and C0 (double column address) for column 6 in ADC counts. Remember the levels (5 levels) are analog signals. The next three represent the signal levels of A2, A1, and A0 (row address) for row 9 in ADC counts. The last plot represents the charge injected for each of the 32 calibration voltage levels (0 to 31).

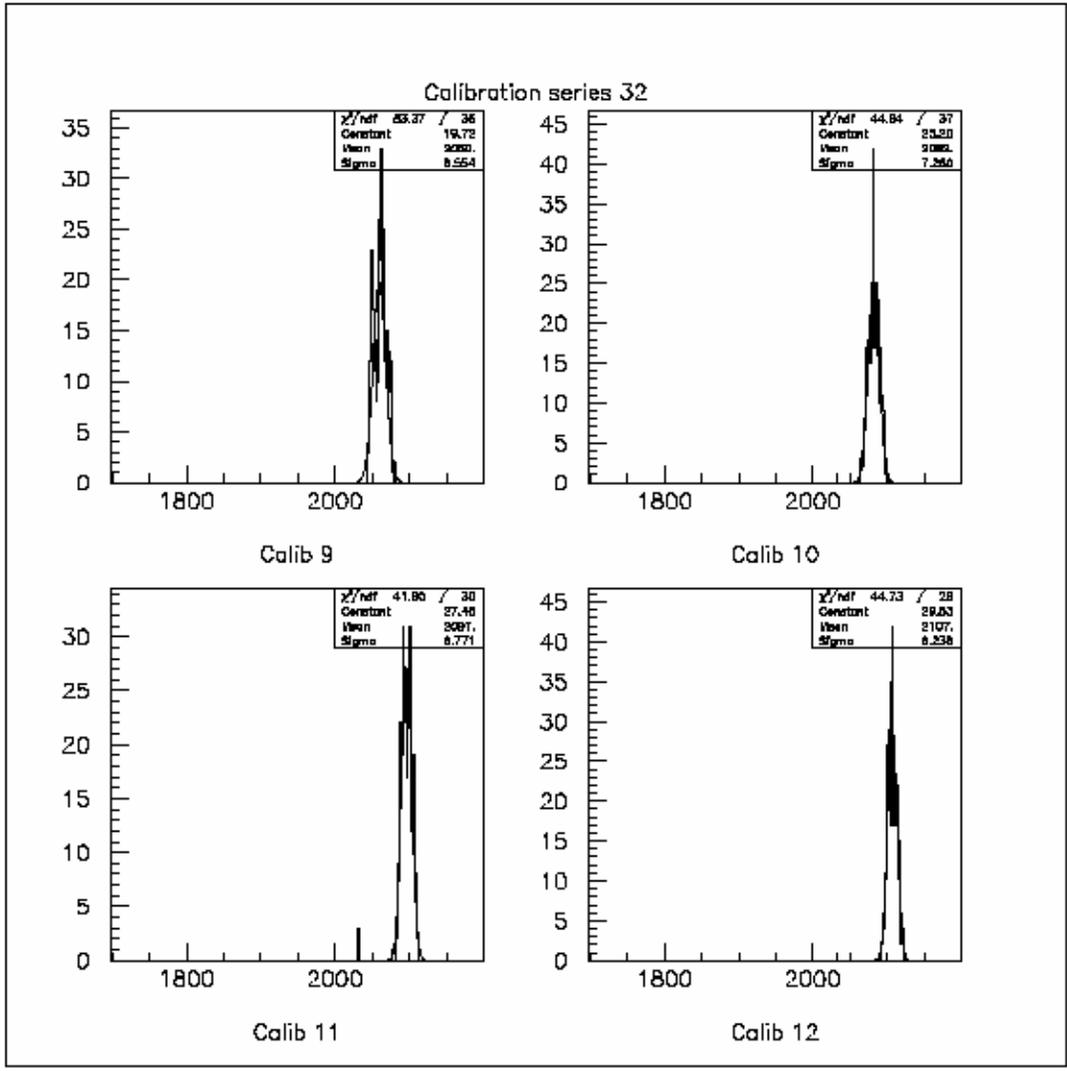


Figure 10 Cal Page

This is one of the Cal pages displaying the mean ADC counts for voltage levels 9, 10, 11, and 12. The means of each of these plots area entered into the Cal00 file.

PCIDAQ 2000 Series ADC Specifics

- Version 2012 – 12-bit analog input resolution
- 32-bit PCI bus
- Up to 20 MHz A/D sampling rates
- 4 single-ended analog input channels
- 4 A/D converters, one per channel
- The A/D data of the 12-bit PCIDAQ 2012 is on the 12 MSBs of the A/D data. The 4 LSBs of the 16-bit A/D data must be truncated by software

The formula between A/D data and the analog value is

$$Voltage = AD_data \times \frac{1}{K} \times \frac{10}{gain}$$

where gain is 1 and K is a coefficient. For a PCIDAQ 2012, $K = 2047 \times 16 = 32752$; and for PCIDAQ 2010, $K = 511 \times 64 = 32704$.

For a voltage of ~ 0 Volt, the ADC output is ~ 2030 counts.

For the UBLK signal (~ -400 mV), the ADC output is ~ 710 counts.

Using the above equation, we have

$$AD_data = (PulseHeight - Pedestal) = (710 - 2030)$$

$$AD_data = -1320$$

Substituting, we have

$$Voltage = \frac{-1320 \times 10}{32752} = -0.403 V$$

Intrinsic Noise

- The capacitance for charge injection is 1.6fF, i.e. 1.6×10^{-15} F
- Knowing the calibration voltage, V_{cal} , applied across this capacitance, the charge injected is known. Hence we know the input charge (injected charge). The V_{cal} DAC is 8 bits wide, i.e. it has 0-255 levels. The range of V_{cal} is 0 to 3.2 Volts. The Cosmo program divides this range into 32 equal values and collects 500 data points for each voltage level.
- If the point selected for noise computation is at level 9, i.e. at 0.9 volts, we have for the injected charge

$$Q_{inj} = C \times V_{cal}$$

$$Q_{inj} = 1.6 \times 10^{-15} \times 0.9 \text{ Coulomb}$$

$$e^- = 1.6 \times 10^{-19} \text{ Coulomb}$$

$$Q_{inj} = 0.9 \times 10^4 \approx 9000 \text{ electrons}$$

- We know

$$Gain = \frac{SignalOut}{SignalIn} = \frac{PulseHeight \times Const}{9000}$$

PulseHeight is the pedestal subtracted pulse height of the output signal and *Const* is a constant that specifies the number of electrons per ADC count.

Because the signal levels are relatively negative, i.e. less than the offset, we do not really know what the pedestal is since we do not know the offset voltage relative to which the A/D conversion is carried out. Do remember that most of the analog voltage levels in the PSI43 chip are negative relative to the offset. A new offset is used such that all signals have relative positive voltages.

One way to determine the pedestal is to plot the pulse heights (in ADC counts) for different voltage levels and determine the Y-intercept. The Y-intercept is the ADC output when the input signal is relatively 0, i.e. this is the pedestal.

Once we know the pedestal, the *PulseHeight* when V_{cal} is 0.9 Volt can be determined.

- Assuming that the width of the distribution, σ , is the result of inherent noise, we have

$$Gain = \frac{\sigma \times Const}{N_e}$$

$$N_e = \frac{\sigma \times 9000}{PulseHeight} \text{ electrons}$$

- N_e is the inherent noise in the pixel in number of electrons.

Measurements with the 1x5 module yielded the following

The following is a table of the noise (electrons) in the various readout chips as a function of the number of the chips on the VHDL.

	NUMBER OF CHIPS ON PLAQUETTE (VHDL)				
	1	2	3	4	5
Roc 0	155	166	179	203	201
Roc 1		132	146	144	191
Roc 2			137	162	162
Roc 3				164	166
Roc 4					150