From: Steve Ross
Skross@anl.gov
Phone 630 252-9510
401-B3212
APS/XSD/Beamline Technical Support Group
Argonne National Lab
WBS = 1.4.3-311110

ABSTRACT:

APS/XSD/BTSG has developed an electrometer amplifier sensitive in the range of picoamps to hundreds of microamps. This low cost amplifier has a high (10^6) dynamic range, and also a sample rate of up to 1600/sec. The noise floor of the electronics is <30 pA RMS (1600 samples/sec, powered by battery). In one 9" x 5" x 2" enclosure, called the ADCMOD module, there are 4 such electrometers. The main application at the APS is to monitor the current output of 4 detectors, commonly PIN diode detectors or split ion chambers for x-ray beam position and intensity monitoring. Digital data from these modules travels over fiber optic links to a VME card set. The data then can be read from the VME bus. We define a basic system as reading 16 PIN diode current sources, but there are variations. We work with the beamline personnel to get a system up and running. EPICs 3.14 drivers exist, written by Dr. Mark Rivers, called quadEM.

CONTENTS: This memo describes electronics developed to read uni-polar currents in the range of picoamperes to microamperes. The primary application is to serve as an electrometer for various arrangements of x-ray detectors. For example this electronics is being used to measure the current from PIN photodiodes, which in turn serve as x-ray beam position and intensity monitors. (XBPM).

This memo is organized as follows:

(1) We give background information about why this electronics was developed and where it was initially applied.
(2) We give a description of the x-ray detector typically used – a four quadrant array of PIN diodes.
(3) We give a **specification for the electrometer**.
(4) We describe the data acquisition system based on a fiber optic link back to a VME card set. We show what VME cards are necessary to make the system

work and discuss some of the options. A basic system is defined to be one running 16 PIN diodes.
(5) We discuss software requirements, including EPICs interfaces. The electronics can be controlled remotely, from the VME bus. One-time output setup scripts are described. We describe how to read the returned data.
(6) Discussion of various LEDs, cables pin-outs, jumpers, and DIP switches.
(7) We list a number of issues that typically must be addressed to get the system at a particular beamline at the APS. We work with you to install a system and we provide almost all parts. Costs are discussed. Section 6.4 discusses day-one, sign of life testing.
(8) Detailed explanation of the commands to the ADCMOD2 or 3.
(9) Design files, WBS information**.**
(10) EPICS quadEM information, courtesy of Mark Rivers.
(11) Appendix, miscellaneous, Sample data, debugging information.

(1) BACKGROUND. This electronics was originally developed per the specifications of ComCAT sector 32 at the APS [REF: Steve Wasserman, Kevin D'Amico]. This work in turn was inspired by an RSI paper discussing results from an earlier electrometer design. A copy of this electronics was installed at ComCAT in early 2000, with the primary application of EXAFS measurements. The electronics collected 12 PIN diode signals, plus the signal from the encoder on the x-ray monochromater. As the energy was swept, plotting one versus the other gave a quick EXAFs curve. The sector 32 system was removed in 2006, as the beamline changed mission. EPICs drivers were written by Mark Rivers, and further maintained by APS/BCDA personnel.

(2) PIN DIODE DETECTOR. The electrometer can work with a variety of detectors. A common one is based on designs at sector 19 [REF: R.W.Alkire, G. Rosenbaum, G.Evans J. Sync Radiation (2000) **7** p.61]). Randy gave me a copy of this article in pdf format. The basic idea is that the monoenergetic x-ray beam passes through a hole in the circuit board holding the PIN diodes, and then on through a thin metal foil, and then on to the experiment. The backscattered fluorescence from the metal foil is detected by the four diodes arranged as a quadrant. As the beam moves closer to one diode, its signal strength will increase. Standard centroiding or difference over sum calculations can then show beam position.

Randy works with companies which supply the foil ACF Metals Co. Inc, Arizona, USA. (520) 325-9557. One issue is that the foil needs to be without "wrinkles", that it be flat. Otherwise, as the APS SR x-ray beam moves across the foil, the signal will change simply from the ups-and-downs of foil itself. The vendor will stretch the foil over a beveled washer, to make it tight like the head of a drum. With micron thick foils, beamlines report current levels of 100's of nanoamps, a good match to the electrometers described here.

Randy also makes the following comments about use of this type detector [REF: email 12/11/02]:

```
1.  All metal foils are pressure sensitive but usually compare well to
the response one gets from an ion chamber i.e., response versus energy.
For kapton backed foils, the response may differ from an ion chamber due
to the multiple elements involved in both fluorescence and scattering.

2.  Separating the diodes from the foil is a good idea but not essential
to performance.  It does allow one to turn the BPM on and off by simply
removing the foil.  Also if the diodes are moved and not the foil during
calibrations the uniformity of the foil becomes less of an issue.

3.  These diodes are light sensitive and that means ANY light, including
light from ion gauges if they are nearby.

4.  Foils can become contaminated with carbon deposits over time and may
require replacement.  Not an issue for operations of less than a year.

5.  This device is only for monochromatic operations, not white light.
```

The mechanical design for the PIN diode holder has since undergone several design changes. For example at sector 1 the PIN diodes are purchased from XIA Inc, and are thicker silicon suitable for higher energy x-rays (70 keV etc). At sector 4 the PIN diodes are purchased to be high-vacuum compatible. **In short order we can work with the beamline users to develop custom printed circuit boards for this mounting.** Three examples of such printed circuit boards are shown. [FIG 1 A,B,C,D]

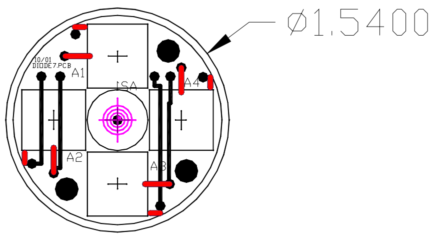
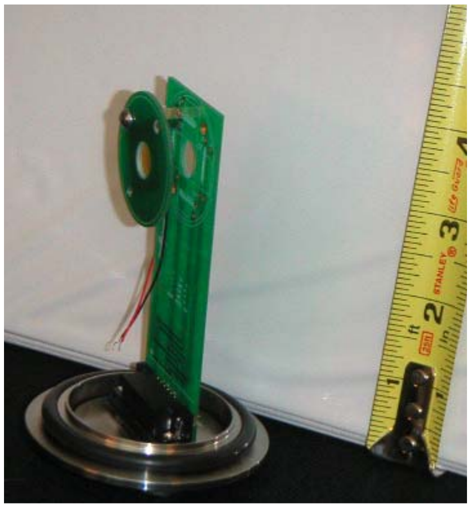


FIG 1A This is the basic printed circuit board type design (DIODE5) holding the four UDT 1 cm2 PIN diodes. It is about a small as possible with these diodes. The center hole in milled out to about 0.5 inch diameter. This circuit board fits into a KF-40 or KF-50 size vacuum system.



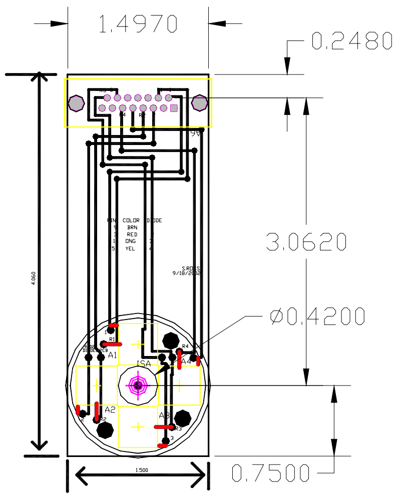


FIG 1 B,C A photograph, and a drawing of the diode6 design. This printed circuit board fits into a KF50 cross and positions the diodes in the center. Standoffs hold a second circuit board approximately 1 cm. above it and this circuit board holds the foil. This is a very simple one-piece design for medium vacuum conditions. Pinout given in section 7.

.

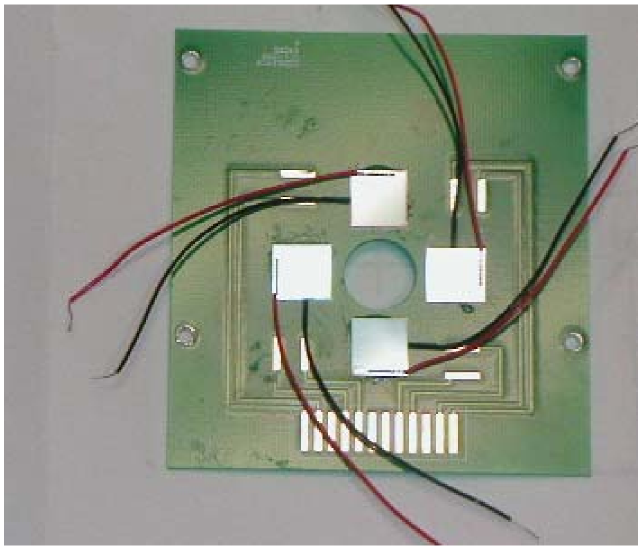


FIG 1D Photograph of DIODE1 layout. Again this is just another variation of how the diodes can be arranged. Red wires signal, black wire are grounded together on the PCB. be changed remotely, so that different foils can be inserted for different EXAFs scans

(3) SPECIFICATION FOR THE ELECTRONICS.

We have two electrometer modules based on an integrated charge amplifier and based on an op-amp voltage amplifier. Most discussion here relates to the charge amplifier, it is the version currently delivered to about 10 APS beamlines. The voltage amplifier is only discussed in section 3.3. We anticipate the use of the voltage amplifier when the signal currents exceed about 100 uA.

3.1 General Information/ Specification. The analog characteristics of the electrometer are based on the Texas Instruments Inc. (formerly Burr Brown Inc.) ddc112 integrated circuit. To see its specifications, go to www.ti.com and search on the ddc112 chip. A photograph of the ADCMOD printed circuit board (version ADCMOD2) is given. (FIG 2A) All the modes and options discussed in the data sheet are implemented in our system. However because ultimately the integrated circuit uses two 20 bit analog-to-digital converters (ADC) we find that most options, most “bells and whistles” are not needed. Twenty bits is a dynamic range of a million, and even this can be extended by making one ADC have a slightly different range from the other. When run in a somewhat standard manner (using the external 220 pf capacitor for maximum rated full well), we observed the following:

•4 Channels – for example currents from 4 unbiased PIN diodes processed in each module
•Maximum readout rate 1600 samples/sec per diode. [1]
•Noise of electronics alone, no connected diodes, <30 pA RMS 1600 Hz (measured, but has not yet been optimized, data sheet claims approximately 6 pA RMS can be achieved).

•Noise of system, including the UDT S100-VL 1 cm^2 diode <100 pA RMS (battery), 200 pA RMS (lab supply)
•Dynamic Range: 20 bit ADC [2]
•Typically all charge collected (one amplifier is collecting while another is processing through ADC)
•Variable gain, variable integration time, set by VME commands [3]
•Fiber optic digital readout back to/from VME, using ST2 connectors, typically using multimode 125/62 micron fiber. APS can supply this, or companies sell commercial fixed length cable.
• EPICs/ Software controllable (see below)

Notes:
[1] See discussion of conv command (conv is short for “conversion”). There are really two interleaved sets of data, each coming at 800 samples/sec.
[2] This dynamic range is only a true dynamic range if the least significant bits are not noise. But this is easily accomplished, see below, section 3.1.1. Also this dynamic range can be extended further by making the “ping” ADC and “pong” ADC have different ranges.
[3] See the range command and the conv command. .

We found that the twisted pair wire that connects the PIN diodes to the electrometer can be up to 2 meters in length, we have not pushed this limit yet. Thus the electrometers do not need to be located right at the detector.

Each ADCMOD module requires about 0.35 amps. The supply voltage should be in the range of 5.5 to 6.5 volts, I typically use +6.5 volts lab supply. This supply is “cleaned up” with a voltage regulator chip. For best noise performance, the units (digital and analog portions) can be powered by a +6 VDC battery, (really +6.5 when fully charged). The analog part of the circuitry uses <80 mA. We could run only this analog part off of the 6VDC battery, while continuing to power the digital part of the board off of a lab supply. I have not yet done this. I have developed but not deployed a battery re-charge circuit, essentially a relay to charge one battery as the other discharges.

Look at the ddc112 data sheet block diagram (data sheet FIGURE 1, from www.ti.com.) Each IN signifies input, and is wired to a **uni-polar source of current** (typically the x-ray diode). This input charges either capacitor A or capacitor B. As time passes, the circuitry toggles between capacitor A and B, each integrating for the same time duration. While an electrometer is charging its capacitor A, it is holding a charge in its capacitor B. On the B channel, while it is holding the charge, the 20 bit analog to digital conversion takes place. Then the reverse occurs, that it, A and B reverse. The point is that we never loose any charge, any coulombs (in continuous mode). The other important point is that each electrometer input creates two digital output numbers, one the result of integrating across A capacitor, one the result of integrating across the B capacitor. We refer to these as “ping” and “pong”, they alternate ping-pong fashion.

The ddc112 data sheet discusses several parameters that can be varied, the most important being gain control and its own readout mode (not to be confused with readout modes of the overall system, discussed below). Varying the feedback capacitor performs gain range control. (See range command.) More often used is gain control implemented by varying the integration time. Of course longer integration times slow down the overall data rate. The two-ddc112 readout modes are “continuous” and “non-continuous” modes. The charge amplifier can either integrate all charge coming to it (continuous), or, if the signal level is high, it can integrate short windows, then go insensitive (discontinuous). These short windows of operation really do not speed up the sample rate however, as the internal conversions and processing still consume a fixed overhead time. A readout rate of about 1600 Hz is about as fast as things go.

Grounding for optimum noise performance needs to be addressed at each beamline. We have seen improvements when the battery is grounded to the optical bench. The ADCMOD runs off low voltage and is inherently electrically safe per APS rules.

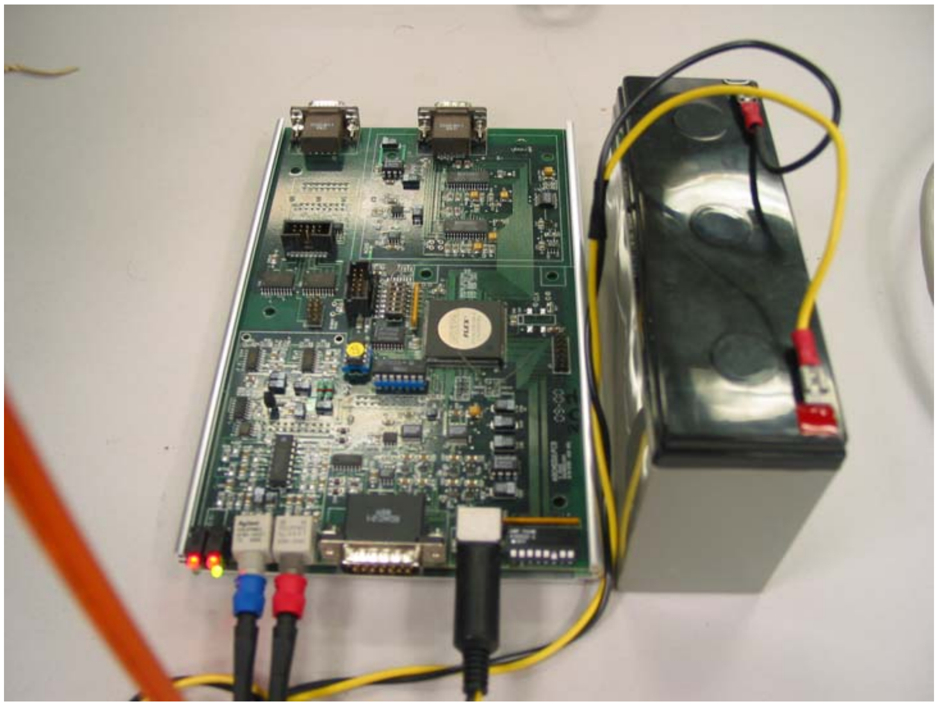


FIG 2A. Charge amplifier four channel electrometer. It is placed into a box measuring 9” x 5.5” x 2.5”. This is the ADCMOD2 version, at 13ID (photo by Mark Rivers). Battery power lowers the noise.

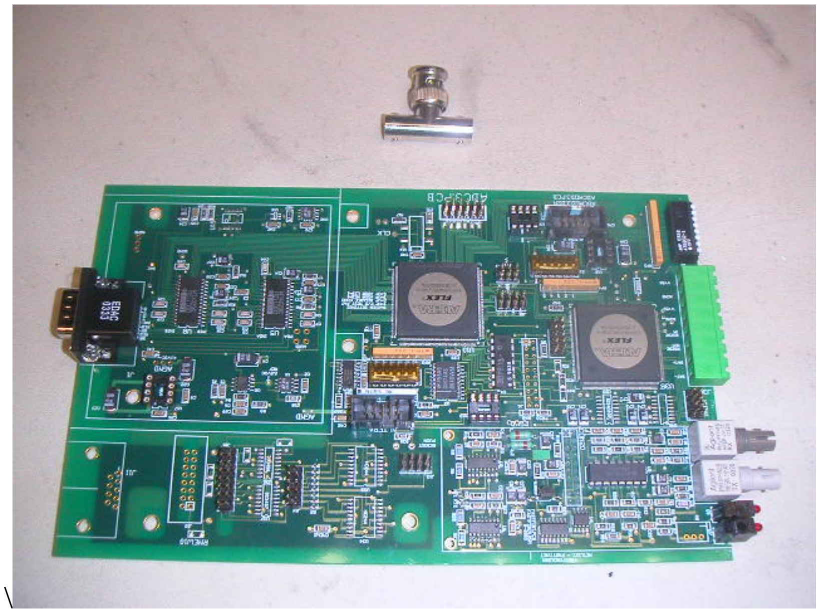
\

FIG 2B ADCMOD3 module's printed circuit board. Current signals come from left. The right side has the terminal block for power, the fiber optic connectors, LED's, and so-far un-used DIP switches.

DISCUSSION OF GAIN RANGES, SIGNALS, NOISE, AND LINEARITY.

The signal out of a charge amplifier V_{OUT} results from an integration (or average) over time Δt of the input current $I_{detector}$:

$$V_{OUT} = 1/C_{FB} * (I_{detector} * \Delta t)$$

Where C_{FB} is the charge feedback capacitor. To vary the conversion gain then we can change Δt or we can vary C_{FB}. See also the ddc112 data sheet "Determining the Integration Capacitor C_F Value" with the knowledge that my reference voltage is 4.1 volts, plus/minus 5 %.

VARIATION OF THE INTEGRATION TIME. The integration time Δt can be varied from about 50 us to 13 ms in my electronics. With integration time between 50 us and 600 us, the ddc112 circuit is being run in "non-continuous" mode. This happens when the conversion time exceeds the current collection time. Some of the charge is lost, but this is presumably acceptable – you set the integration time to be low because you had too high of a signal, and one way to mitigate this is to discard signal. After this integration time exceeds about 600 us, the ddc112 circuit is being run in "continuous" mode (see its data sheet). The 13 ms firmware limit can be increased to that allowed by the ddc112, 1-sec if users need this. (See table iv of ddc112 data sheet.) This would allow the measurement of very tiny currents and is a region of parameter space I have NOT explored yet.

VARIATION OF THE FEEDBACK CAPACTOR. The ddc112 integrated circuit has 7 internal capacitors, any one of which can be selected. Internal capacitors are in 12.5 pf to 87.5 pf (not a particularly wide range). Additionally, one "range" tells the converter to use an external $C_{FB.}$ This external feedback capacitor C_{FB} can be varied over a wide range, extent TBD. I have so far used values as high as 5000 pf, but have the most experience with about 220 pf.

I use the rule of thumb for calibration: 1 count = 1 pA on range scale 0 with integration time Δt = 820 us, C_{FB} = 220 pf. This gain varies +/- 20% depending on the amplifier, due to the tolerance of C_{FB}. DC offsets are much more tightly held, to about 4096 (digital), 0x1000 (hex) counts.

SETTING OPTIMUM GAIN. To make the best use of the 20 bit (million to one) dynamic range, we would set the LSB equal to the noise floor. Let this be about 10 pA RMS. Set the integration time at 820 us (0x200 on "conv"). Then from the above discussion, C_{FB} = 220 pf * 10pA/1pA = 2200 pf. The 20 bit dynamic range sets the highest current at 10 uA, with integration time 820 us. (I am gaining practical experience with 1000 pf, close enough.) But if we then drop the integration time down to about 50 us, then we pick up more top end dynamic range, up to a hundred microamps or so.

CAN THE NOISE BE FURTHER REDUCED? I note from the data sheet that with the largest feedback capacitor (range 0), that the noise is quoted in the data sheet to be about 5 ppm RMS. This would lead us to the hope of a fantastic noise floor of 5 pA RMS, bandwidth about 1.5 KHz. I have not had time to explore this yet, and have “stopped” at 25 pA RMS. See “TYPICAL PERFORMANCE CURVES” page 4 of ddc112 datasheet, particularly those on noise.

I made use of all the suggestions to minimize noise – guard rings, separate grounds planes analog and digital, extensive supply buffering etc. With battery power, we’ll see how low noise can go.

LINEARITY. Linearity from the data sheet is quoted at +/- 0.005%. See the data sheet “Specifications” for a better description. I have not yet verified this, it is hard to guarantee my standards to this level. I have NOT checked this with the various higher values of external feedback capacitor. Things seem ok, but this is not data.

EXTENDED GAIN. If the two amplifiers, ping and pong, have different feedback capactitors, then the gain can be extended. Each gain would be sampled at the half-rate (e.g. 800 Hz).

3.3 CURRENT AMPLIFIER. Another electrometer printed circuit board has been fabricated and tested. At present I am not spending a lot more time on this path, but can return to it in the future. The main driver to create this system was from users who wish to monitor higher levels of currents – microamps on up through milliamps.

Basically this amplifier consists of an electrometer grade operational amplifier (Burr Brown OPA129U) run as a current amplifier (large feedback resistor). There are three gain ranges, remotely selectable. When a different gain range is selected, a micro relay switches in a different feedback resistor for the amplifier. The analog amplifiers then send signal into a 24 bit 40 KHz ADC (ADS 1252U see www.analogdevices.com). (FIG 2C)

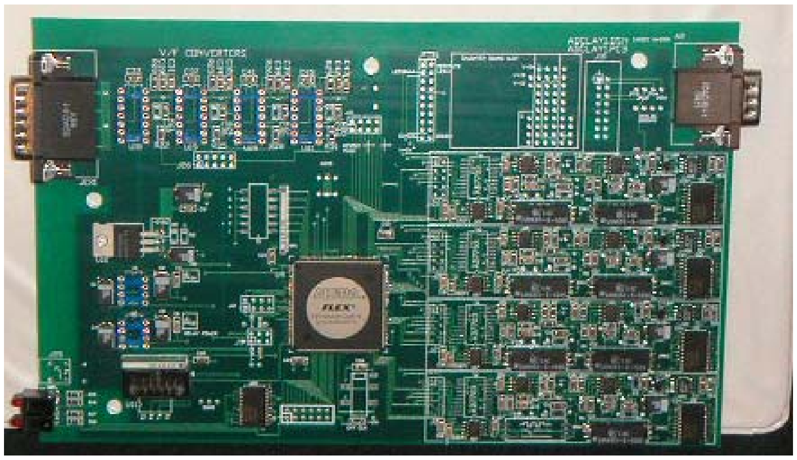


FIG 2C. A photograph of the current amplifier method for electrometers. This board holds four chains of electrometer current amplifiers. Data is digitized, and can be packaged in a way consistent with the charge amplifiers. There are V/F converters on this board but this is not a desired development path, ADC's and digital signal averaging are better.

This hardware exists but work needs to be done to program the firmware. This hardware is mechanically compatible with the charge amplifier approach. We could build another circuit which would allow toggling between the charge and voltage amplifiers – more bells and whistles for the future.

(4) VME BASED DATA ACQUISITION – WHAT YOU NEED FOR BASIC SYSTEM.

We discuss a basic system – what you need. Keep in mind that these cards really just form a VME based data acquisition system. In other words, as an option, we can input into the data stream "other" sources of data such as the timing of the APS top off signal. There is much flexibility here both in terms of hardware (connectors to input/output the data) and firmware.

4.1 **BASIC SYSTEM**.
We can define a very basic system as supporting four PIN diodes, one ADCMOD card (version ADCMOD2 or ADCMOD3). The next level up is to have a system that supports16 PIN diodes, and this is the system we describe here. To do this the beam line needs:

(1) Qty 4: the quad PIN diode mechanical holders. .
(2) Qty 4: Four channel electrometer electronics boxes, "ADCMOD2/3". These will reside in the hutch, near the detectors. They are the amplifiers, and ADC units with fiber communication.
(3) Qty 4: terminated fiber optic cable pairs run from the hutch, to the control rack. Each ADCMOD board needs a pair of fibers, one transmits, one receives. Typically the fiber is bundled as 4,8,12 fibers in a cable. For example the breakout fiber (orange color) has 4 125/62.5 um fibers in it. We can provide the fiber.
(4) Qty 1: an APS supplied VME card FIBER2 (or FIBER2A) to receive the fiber optics One of these cards can support four (4) ADCMOD's.
(5) Qty 1: an APS supplied VME card PASSTH4, or PASSTH6, which can be read by a VME host, such as an IOC running EPICS.

As discussed in section 3, the ADCMOD amplifiers read in the PIN diode signals, convert to digital levels, and output via a fiber. Each FIBER card can support 4 ADCMOD cards, which in turn read 16 PIN diodes (e.g. four quad arrays). Each FIBER2 card needs the PASSTH4/6 card to format the data to be read over the VME backplane.

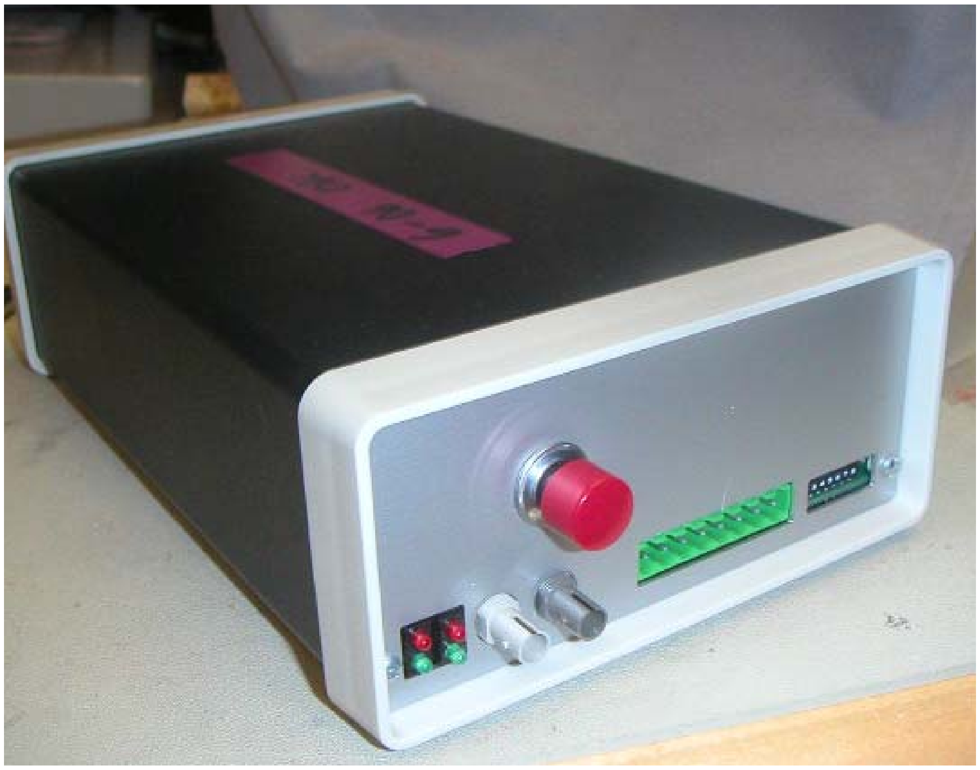


Figure 4A. ADCMOD, here version 3. Version 2 has a different power connector. The red button is for reboot of the firmware. Across the bottom are the indicator LED's, the fiber optics, the green terminal block for power/control of batteries, and (un-used) DIP switches.

Fig 4B FIBER2, or FIBER2A VME board
This board brings in 4 sets of fiber (transmit/receive) from outlying adc modules (or other sources of data). The fifth fiber set show (at the top of this view) is not used. The lower fiber pair is channel 0 and is always active. As additional channels are made active, the user must set DIP switches. These DIP switches eliminate spurious inputs on un-used channels. See also jumpers section concerning VME bus grant pass-alongs.

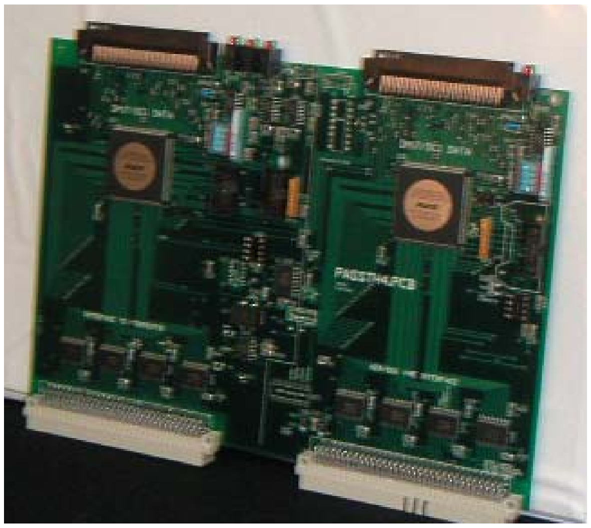


Fig 4B PASSTH4 VME board. This board brings data from the fiber2a board, performs digital signal averaging, and presents the data to the VME bus (A24/D16). It also moves setup data from the VME bus out, to the fiber2a board, and then on to the modules. PASSTH6 looks similar but has additional DIP switches in the upper right, discussed below. There is a (grey) ribbon cable between the top two 80 pin connectors, not shown. See also jumpers section concerning VME bus grant pass-along. This board can be used as a general interface into VME.

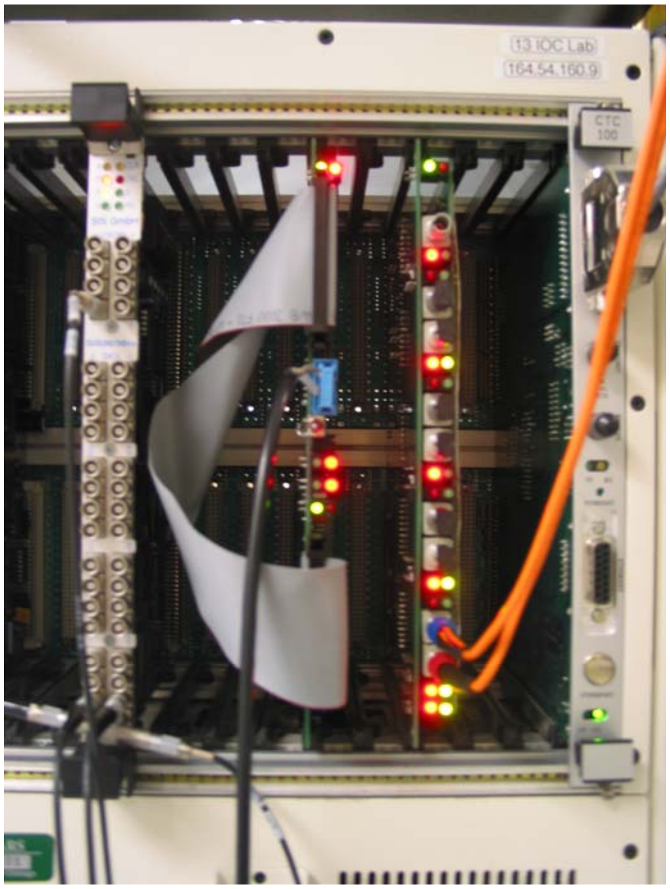


Fig 4C Both card in action, receiving data, at APS Sector 13 (photo from Mark Rivers).
.
4.2 REGISTER MODE The purpose of the register mode of operation is to take the data from the PIN diodes and get it to a location readable over the VME bus. The data comes out of all the ADCMOD's, travels over the fiber back to the control VME rack. The multiple channels are collected by the FIBER2/A card which then passes all the data to a PASSTH4,6 VME card. (This is done over the VSB bus, and thus requires a small cable to be installed on the backplane.) Traveling over a ribbon cable from the bottom half of this card to the top half, the data is saved into a static RAM memory. This RAM can be read over the VME bus by an A24/D16 parallel input/output (PIO) command. The data is continuously being updated. Thus the register values will continuously change as new data arrives from the electrometers. An output pulse can indicate this update.

Thus to read the signal of a given PIN diode, the VME controlling computer would access a specific VME A24 address. For example read the D16 values from hex address 0xf000, 0xf004, 0xf008, 0xf00c and you obtain the MSB 16 bit PIN values. This would typically be a parallel input/output VME operation (PIO).

The original COM-CAT sector 32 electronics was meant for quick EXAFS, and therefore recorded monochromater encoder position signals simultaneous with the electrometer currents. A user-specified number of samples were recorded into a large (64MB) VME memory from Chrislin, Inc.

(5) SOFTWARE REQUIREMENTS

. I run an entire stand-alone software system based on a PC, Microsoft Windows, and MSVC++ 6.0 code, and using the Mark Rivers/BCDA EPICs interface. (See section below.) From1999-2006, at COM-CAT sector 32, all this was done through SPEC. Other real-time operating systems can be used.

5.1. DATA SENT OUT TO THE ADCMOD2 or 3 MODULES FOR THEIR SETUP. The user must supply initializing information to the ADCMOD electrometer. This data sets the electrometer's feed back capacitor gain range, the period for the charge collection measurement, some details of for the DDC112 integrated circuit's operation related to continuous or non-continuous modes, and a go-signal to begin to take data. The ADCMOD3 added a feature, and hence a command, to remotely reset, or reboot, the amplifier module. On the ADCMOD2, this function must be performed manually by pushing a button on the box. See below.

This initializing need only be done upon power up. If you are happy with the gains then it never needs to be repeated. This is a common occurrence with the 20-bit ADC with a10^6 dynamic range. At present, all ADCMOD modules are set up identically (assuming they are cabled in, and powered up.) .

Internal to the system, over the fiber, between FIBER and ADCMOD, data is passed as serial bit streams 48 bits long. A command to the ADCMOD is 48 serial bits long. **For the VME host to command the ADCMOD it must write three 16 bit hex words to three specified "mailbox" addresses, then write a fourth word to a fourth address to trigger the serial burst. This process is then looped to send additional 48 bit serial bit commands.** The specific addresses here can be set in the firmware, and need to be discussed. This system was implemented to give some flexibility – part of the 48 bit command is a device address. Most users at present will only have one TYPE of device out on the fiber link – a varied number of ADCMOD's.

The following is a basic script. **Only 6 commands are needed: reboot, (ADCMOD3 only) range, pulse, period, conv, go. There is no real order, but I recommend this script.**

Write A24,D16 by PIO transfer to the VME bus. Write to address 0x00f018. The data is 0xa000. The upper data lines (VME supports 32) are don't-care. The upper addresses lines (VME supports 32 bit addressing) are "don't care". VME supports address modifiers (AM5:0), and various signals to say the data is a LONG, which style ENDIAN in use etc. While I bring these signals into the Altera chip, as of now I do not decode them. Likely this will be added, as I understand any issues users have with VME bus contention – it is not difficult.

Thus the reset command is (ADMCOD3 only – ADCMOD2 will simply ignore this command with no harm done. To reboot an ADCMOD2 (or 3), push the button.)
a000 000e 0000

the range command
a000 0001 0000

the pulse command is
a000 0006 100f

the period command is
a000 0007 ffff

the conv command is
a000 0005 0280

the go command is
a000 0004 0001.

This test script is an input to my C++ code. (Version 1/2006, added re-boot). Read the following script as AAAADDDD comment. For example **f018a000 2range** from the table below means to my system: send data a000 to VME address 0x00f018. (See also DIP switch settings to move cards around in address space.)

```
f018a000 reboot
f010000e reboot
f0080000 reboot
f020cccc send
f018a000 2range
f0100001 3range
f0080000 4range
f020cccc 5send
f018a000 6pulse
f0100006 7pulse
f0081000 8pulsefff0
f020cccc 9send
f018a000 10period
f0100007 11period
f008ffff 12period
f020cccc 13send
f018a000 14conv
f0100005 15conv
f0080280 16conv
f020cccc 17send
f018a000 18go
f0100004 19go
f0080001 20go
f020cccc 21send
```

5.2 DATA FROM ADCMODs to the VME. Under normal operation only the ddc112 data is presented into the VME accessible registers. There are diagnostic modes not discussed here.

The DDC112 analog to digital converter output values go from0x00000 to 0xfffff. (See its data sheet). These would be treated as unsigned numbers in your computer code. This entire 20-bit value is given to PASSTH (version 4, or 6) from the ADCMOD, and from the FIBER VME card. But since we are using a VME A24 D16 transfer, from PASSTH4,6 to your VME master, only 16 bits are passed at a transfer. A new firmware feature has been added to PASSTH4 or 6, as of 6/2006. MSB Bits [19..4] are given to one VME address, LSB bits [15..0] to another, offset from the first by VME address decimal 32, hex 0x20. You may choose only to take up the most significant 16 bits, and live without the lower 4 bits. But if you want all 20, it is your responsibility to make sure they are the **same** overall 20 bits. There are 12 bits of overlap so you can check. But it is your responsibility to read the data in a timely manner before it is overlaid with the next time's data, perhaps about 600 us later. This firmware change is backwards compatible, so existing beamline code will continue to read the most significant 16 bits.

Every new update of data is marked by a pulse available out of the PASSTH board. This may not be set up for you, just ask me to do so. This pulse can be fed back into the VME system and used as an interrupt. Mark Rivers does this with an IP UNI-DIG card. Mark wants me to actually create my own interrupt, and PASSTH6 has this hardware. But it is not yet fully tested and hence you do not have it yet. So stay tuned.

The DDC112 INTERNAL DC OFFSET. When no current is going into the module, the output value is 0x01000, decimal 4096. I tend to see this a little low, e.g. 0x00f7a or thereabouts. You can get a quick indication of the system noise by watching the variations here. See section 3 under optimum gain, to see a rule of thumb about the ampere value of the least significant bit, out of the 4096. If you read the most significant 16 bits (out of the 20), you will see this as decimal 4096 / 16 = 256, hex 0x100. If you read the least significant 16 bits out of the 20, you will see this as the decimal 4096.

5.3 VME BASE ADDRESSES. The data from the electrometer is placed in 24 bit VME memory starting at a base address selectable by the switches on PASSTH. A typical base address is 0x00,f000.

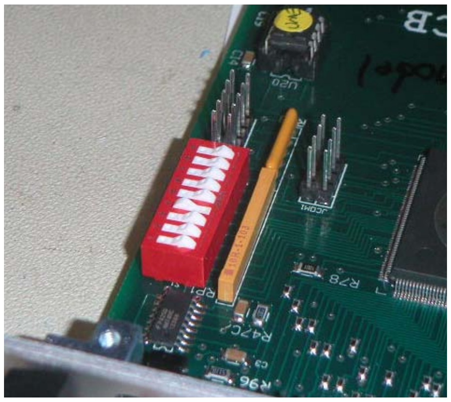
SETTING THE A24, A23 VME BASE ADDRESS: PASSTH4. A23/A22 addresses for the data table can be set by DIP-switches. These switches are labeled “ADDRESS 1,2,3,4” and are located at the top of the PASSTH4 circuit board. This allows the user to move the address table of the read-out data to 4 different places in A24 address space. This base address can be further modified, but it takes some Altera firmware changes (and hence a new Altera PROM).

MSB address table VME A24 BASE address	**DIP switch 3**	**DIP switch 4**
0x 00, fxxx	Down	down
0x40, fxxx	Down	up
0x80, fxxx	Up	down
0xc0, fxxx	Up	up

Where “up” means the side of the switch near the top of the VME card, that is, near the resistor pack, is UP.

SETTING THE A24, A23 VME BASE ADDRESS: PASSTH6. A23/A22 addresses for the data table can be set by DIP-switches. See table below. In photo, switch 1 is to left.

MSB address table VME A24 BASE address	**DIP switch 6**	**DIP switch 7**
0x 00, fxxx	Down	down
0x40, fxxx	Down	up
0x80, fxxx	Up	down
0xc0, fxxx	Up	up


PASSTH6 set for base address 0x00f000.

5.4 MEMORY MAP OF VME DATA, GIVEN THE BASE ADDRESS. Each ADC conversion is 20 bits. However the VME transfer is only 16 bits. So it takes 2 transfers, and hence 2 VME addresses, to contain the 20 bits. One VME address contains bits [19:4]; the other VME address contains bits [15:0]. Note there is overlap, redundancy.

Each PIN diode is wired to 2 ADCs, referred to before as “ping” and “pong”. Again this is so that one ADC can be converting while the other is reading out. And again, ping/pong are displaced in time usually by 100’s of microseconds.

Each ADCMOD supports 4 PIN diodes (i.e. 4 sources of current)..

Each FIBER2 card supports 4 ADCMOD’s on the fiber channels 0 – 3, pair 0 being on the bottom, and working up.

So the total amount of memory needed for one ADCMOD is 2 bytes (16 bits) * 2 (MSB/LSB) * 2 (ping/pong) * 4 ADC’s, equals 32 bytes, or 16 2-byte words.

So the total amount of memory needed for all four ADCMODs is this 32 bytes * 4 = 128 bytes, or 64 2-byte words.

Finally, for historical reasons, **I waste memory**. Originally I had intended to interleave the MSB/LSB (16 bit, 2 bytes, 1 word) values. But later I decided just to offset the LSB values, hoping not to make too big of mess with the software, and keep things backwards compatible. Maybe a mistake, but anyway….

Table 5.3A Address table to read out MSB data bits [19..4], of fiber channel 0. Assumes a PASSTH VME base address = 0x 00, f000

VME A24 address	FIBER CHAN	Ping/pong	Diode number ”typical” position	Suggested Wire color	Integrated Circuit, input pin	Capacitor, pins on the IC’s
0x00 f000	0	Ping	1 top	Brown/black	U5.1	1B U5.3 to U5.4
0x00 f004	0	Pong	1 top	Brown/black	U5.1	1A U5.5 to U5.6
0x00 f008	0	Ping	2 inboard	Red/black	U5.28	2B U5.26 to U5.25
0x00 f00c	0	Pong	2 inboard	Red/black	U5.28	2A U5.24 to U5.23
0x00 f010	0	Ping	4 outboard	Yellow/black	U8.1	1B U8.3 to U8.4
0x00 f014	0	Pong	4 outboard	Yellow/black	U8.1	1A U8.5 to U8.6
0x00 f018	0	Ping	3 bottom	Orange/black	U8.28	2B U8.26 to U8.25
0x00 f01c	0	Pong	3 bottom	Orange/black	U8.28	2A U8.24 to U8.23

Fiber channel 1 then begins at 0x00,f020, fiber channel 2 at 0x00,f040, fiber channel 3 at 0x00,f060.

The LSB values are offset from the MSB values by 0x0080, as follows:

Table 5.3B Address table to read out LSB data bits [15..0], base address = 0x 00, f000

VME A24 address	FIBER CHAN	Ping/pong	Diode number "typical" position	Suggested Wire color	Integrated Circuit, input pin	Capacitor, pins on the IC's
0x00 f080	0	Ping	1 top	Brown/black	U5.1	1B U5.3 to U5.4
0x00 f084	0	Pong	1 top	Brown/black	U5.1	1A U5.5 to U5.6
0x00 f088	0	Ping	2 inboard	Red/black	U5.28	2B U5.26 to U5.25
0x00 f08c	0	Pong	2 inboard	Red/black	U5.28	2A U5.24 to U5.23
0x00 f090	0	Ping	4 outboard	Yellow/black	U8.1	1B U8.3 to U8.4
0x00 f094	0	Pong	4 outboard	Yellow/black	U8.1	1A U8.5 to U8.6
0x00 f098	0	Ping	3 bottom	Orange/black	U8.28	2B U8.26 to U8.25
0x00 f09c	0	Pong	3 bottom	Orange/black	U8.28	2A U8.24 to U8.23

Fiber channel 1 then begins at 0x00,f0a0, fiber channel 2 at 0x00,f0c0, fiber channel 3 at 0x00,f0e0.

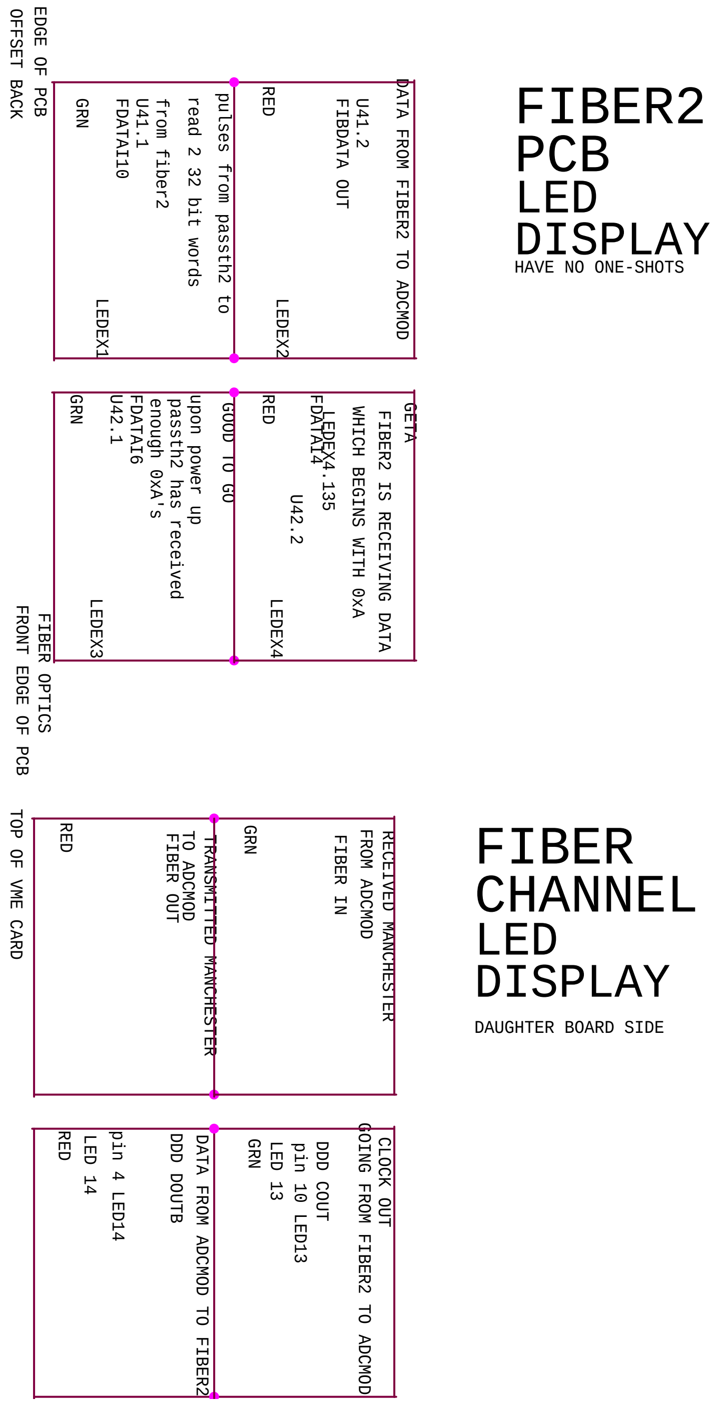
Another way of saying this, is to look at the function of each address bit, in the address:
[Ref 1/18/2002 exafs #1 notebook, data acq, and the ADCMOD firmware]

A7	0 = upper MSB 16 bits of data[19..00], 1 = lower, LSB 16 bits of data[15..00]
A6	fiber channel MSB
A5	fiber channel LSB
A4	0=DOUTA=U5, 1=DOUTB=U8 Which DDC112 integrated circuit.
A3	1 = input on pin 28, 0= input on pin 1. Which pin on same integrated circuit.
A2	1 = first CONV cycle, CONV=1, 0 = second CONV cycle CONV=0, (PING/PONG)
A1	0
A0	0

6.0 ISSUES RELATED TO OPERATING THE SYSTEM

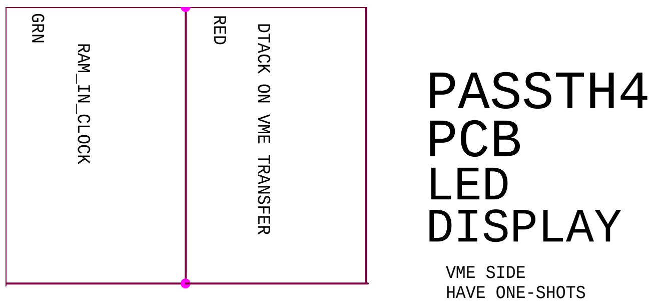
6.1 . LEDs. This section describes the LED displays on the FIBER2 and 2A, PASSTH4 and 6, and ADCMOD2 or 3 board. The LED function has not been changing with PCB updates. LED’s are meant to be useful for monitoring operations, and for debugging.

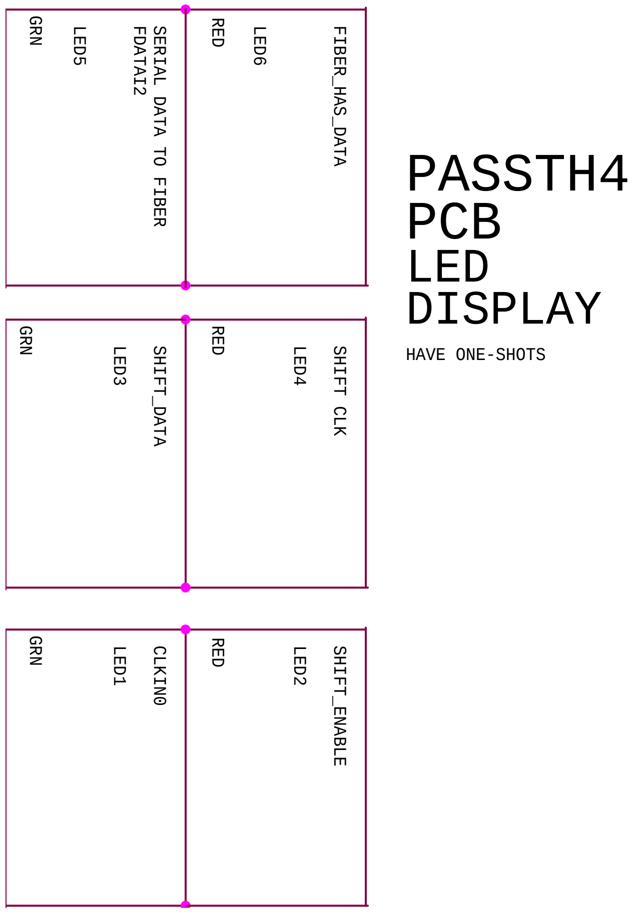
6.1.1 FIBER2 LED’s.



The top set of 4 LED’s are at the top of the FIBER2 VME card. The bottom set of 4 LEDs exists under each fiber pair, so there are 5 copies. Note: you can ignore flickering LED’s on un-used FIBER VME card channels. These lights are triggering off of noise in the un-capped, un-terminated fiber input.

6.1.2 PASSTH LED's (version 4 and 6).





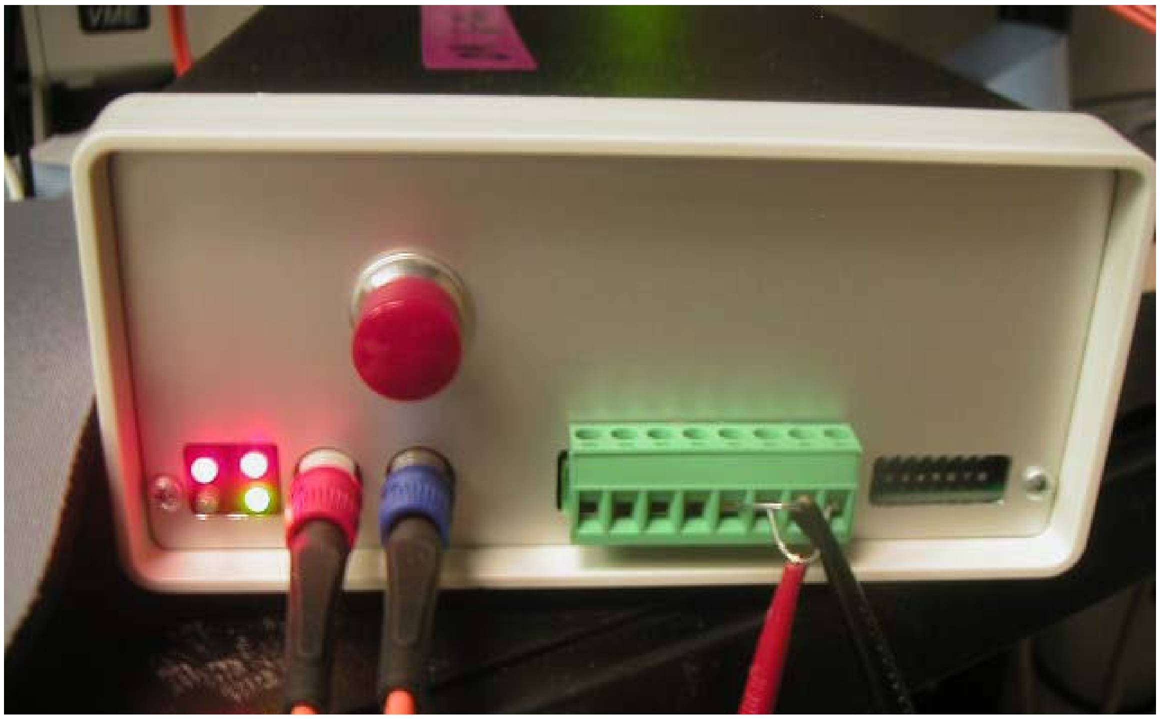
The top set of 2 LED's are on the top of the passth4 card. The bottom set of 6 LED's are just below the toggle switch (which in photo is just below blue).

6.1.3 ADCMOD LED's (version 2,3).

ADCMOD2
LED
DISPLAY

DATA FROM DDD TO FIBER2 J8.4 ALT.150 LEDBB1 DDD DIN U7.2 LED14 RED	TRANSMITTED MANCHESTER TO FIBER2 J8.6 FIBEROUT TX DDD.8 U3.2 RED
DATA FROM FIBER2 TO ADCMOD2 J8.10 DDD DOUTB U7.1 LED13 GRN	RECEIVED MANCHESTER FROM FIBER2 J8.16 FIBERIN RX DDD.3 U3.1 GRN

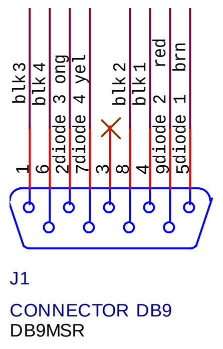
EDGE OF PCB FIBER OPTICS



6.2 **Cabling pinouts.**

6.2.1 Current Input DB9 male connector. The cable from the diodes which plugs into the ADCMOD has the following pinouts.

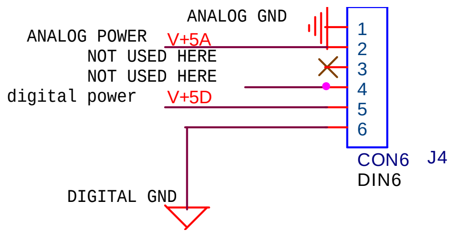
Pin	function	cable color
1	return diode3	blk
2	signal diode3	ong IN2B U8.28
3	not connected	
4	return diode1	blk
5	signal diode1	brn IN1A U5.1
6	return diode4	blk
7	signal diode4	yel IN1B U8.1
8	return diode2	blk
9	signal diode2	red IN2A U5.28



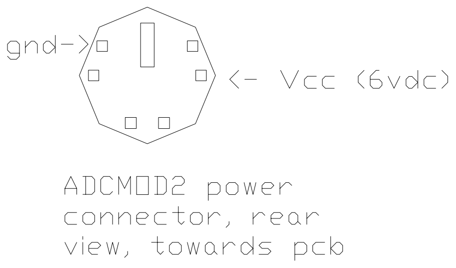
looking into the male connector
on the adcmod2 pcb

6.2.2ADCMOD2 DC power connector:

The DC power connector.



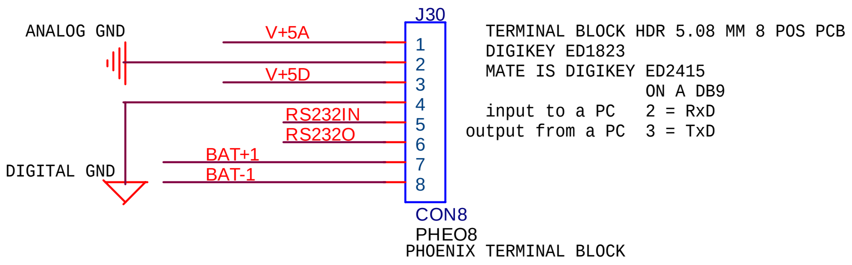
In ADCMOD2, this is a round connector: (adcmod2_pwr_connector.dwg). Typical Vcc(+6v) uses a yellow wire, gnd a black wire.



6.2.3 ADCMOD3 DC power connector: In ADCMOD3, this is a terminal block connector.

In photo, black leads = gnd, red leads = +6.5vdc. External jumper maps analog and digital power in parallel.

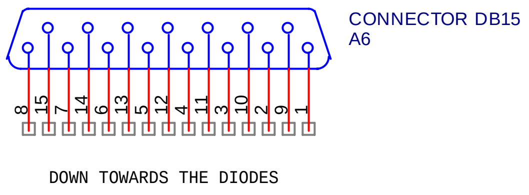
The pinout for this terminal block:



Pin 1 is on the right in the photo. At present pins 5-8 are not used – they will eventually control the battery supply.

6.2.4 Connector into rectangular PIN diode board, using db15. See also figure 1B,C

```
see diode6.pcb
DIODE6, DIODE6T.PCB

15  RETURN DIODE 3
14 DIODE 3

9   DIODE 1
10  RETURN DIODE 1

3  DIODE 2
4   RETURN DIODE 2

5   DIODE 4
6   RETURN DIODE 4
```



```
DB15 CONNECTOR, PLACED ON THIS SIDE OF PCB
DIODES ALSO ON THIS SIDE OF THE PCB
on the pcb, use (per sector 5 design) DA15S-FRS, newark 89N1585)
```

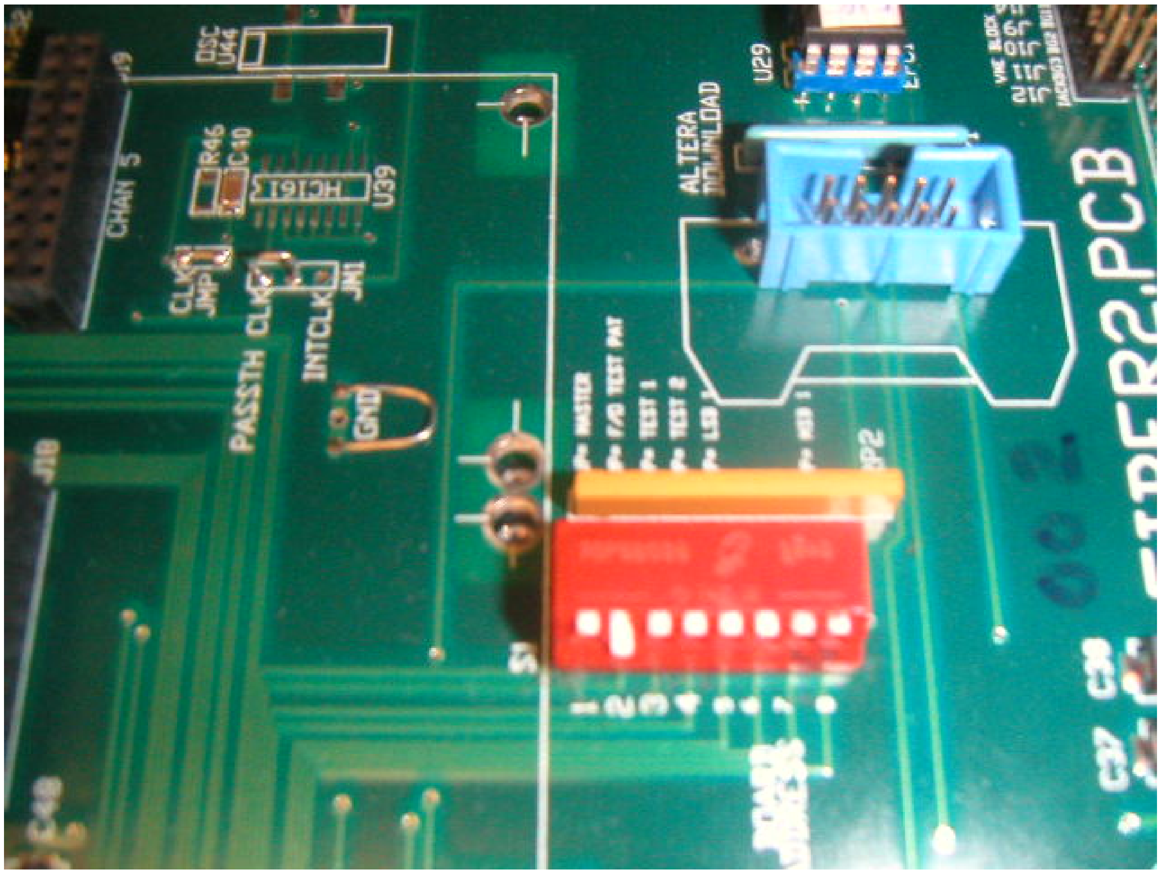
6.3 DIP SWITCHES.

DIP switches serve the purpose of choosing VME address space, and setting diagnostic tests. The photos always show a “normal” operation.

6.3.1 FIBER2, FIBER2A

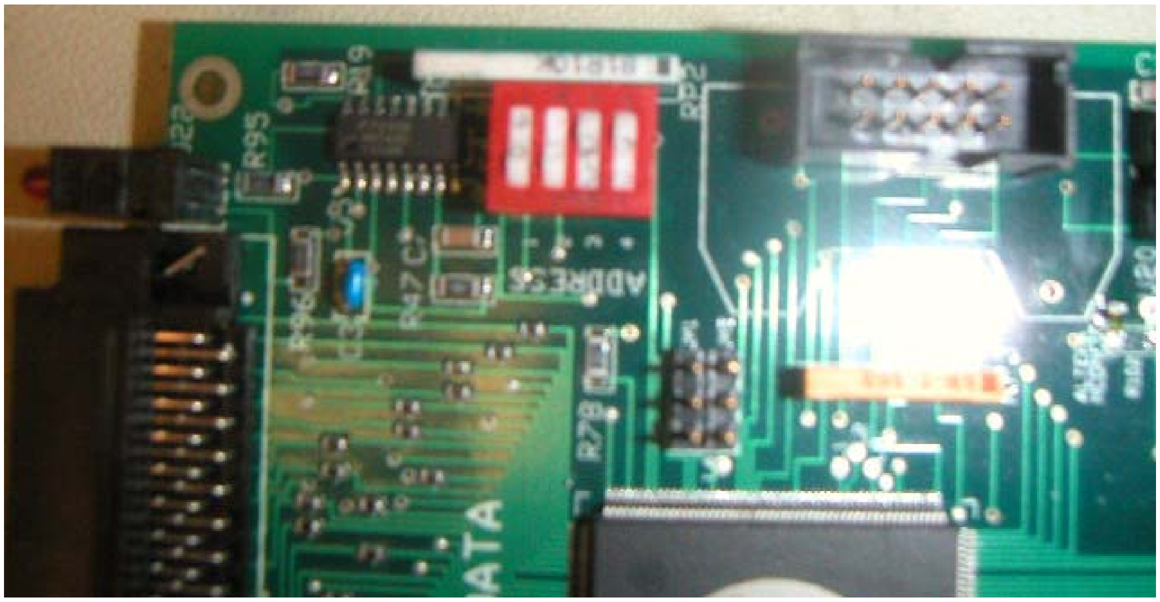
A FIBER2 card can support up to 4 ADCMOD’s. However if there is no ADCMOD attached, we typically set DIP switches to block the data, which is likely junk from random switching of the un-used fiber optic receivers. So the setting of the DIP switch is AND-ed with the incoming data, for fiber channels 1, 2 and 3. (Channel 0 is assumed to be used.)

Photo shows normal setup when only one ADCMOD is used, channel 0. As additional ADCMOD’s are fiber-pair cabled in, then change DIP switches on the far right to allow data into system. The far right DIP switch 8 allows channel 1, the switch to its immediate left (#7) allows channel 2, then the next switch (#6) channel 3. Other DIP switches not used, or used for tests. You can skip a channel if you wish, just set the switches accordingly, but I never do.



6.3.2 PASSTH4 VME side. When the side of the switch closest to the VME side edge of the PCB is UP, then the switch is a logical low (I think!) Photo shows normal settings. (I think!)

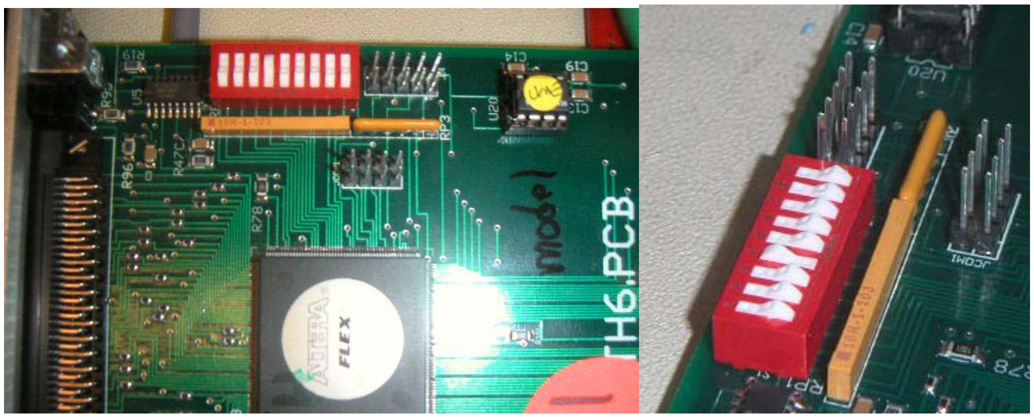
Switch 1 sw136, normally low. When High, sends clock out on SHIFTDATA, used to test that data is going out to set up ADCMOD's. Pictured to left on photo.
Switch 2 sw135 Normally low. When High, go to test mode for data.
See section on VME base address for switch 3, switch 4.



6.3.3 PASSTH6 VME side.

Switch 1 sw141, not used, leave low, on left side of photo, which shows normal settings.
Switch 2 sw140, not used, leave low
Switch 3 sw139, not used, leave low
Switch 4 sw136, on left on photo, normally low. When LOW (??) sends clock out on SHIFTDATA, used to test that data is going out to set up ADCMODs. Normally high
Switch 5 sw135 When High, go to test mode for data. Normally low.
See section on VME base address for switch 6, switch 7
Switch 8, 9 not used, leave low.

Typical PASSTH6 settings, for base address 0x00f000:



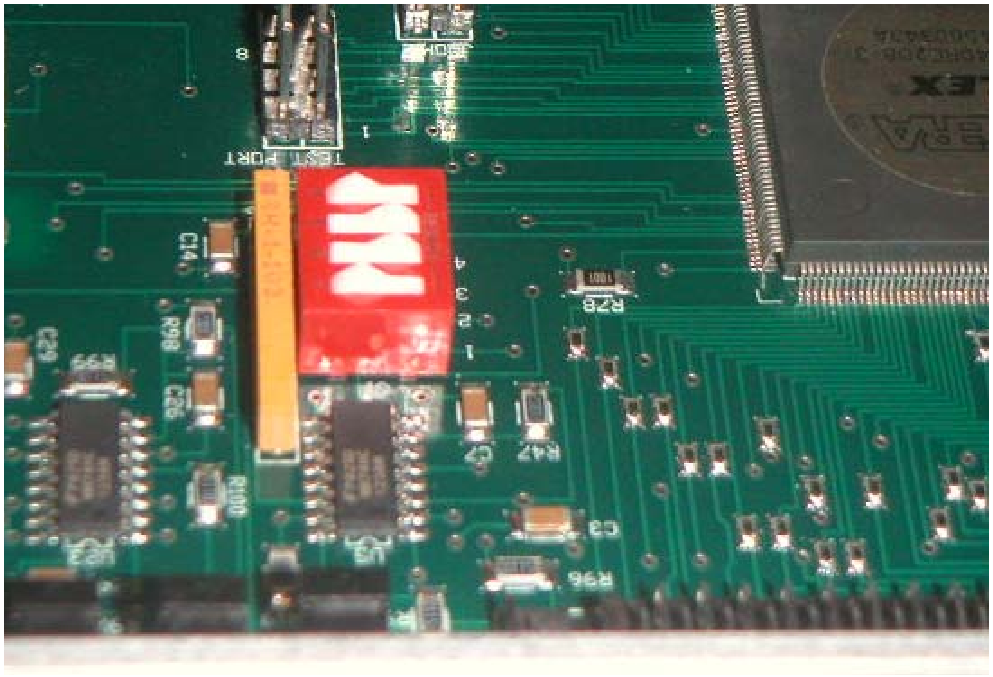
6.3.4 PASSTH4,6 VSB side. When the side of the switch closest to the VME side edge of the PCB is UP, then the switch is a logical low.

Switch 1 sw136 not connected (at bottom of photograph).

Switch 2 sw135 normally low. This switch is connected to the fiber card at reset, and thus when pulsed lo-hi-lo, it will reset fiber2.

Switch 3 sw134 normally low. Under control of switch 4, used to reboot fiber2.

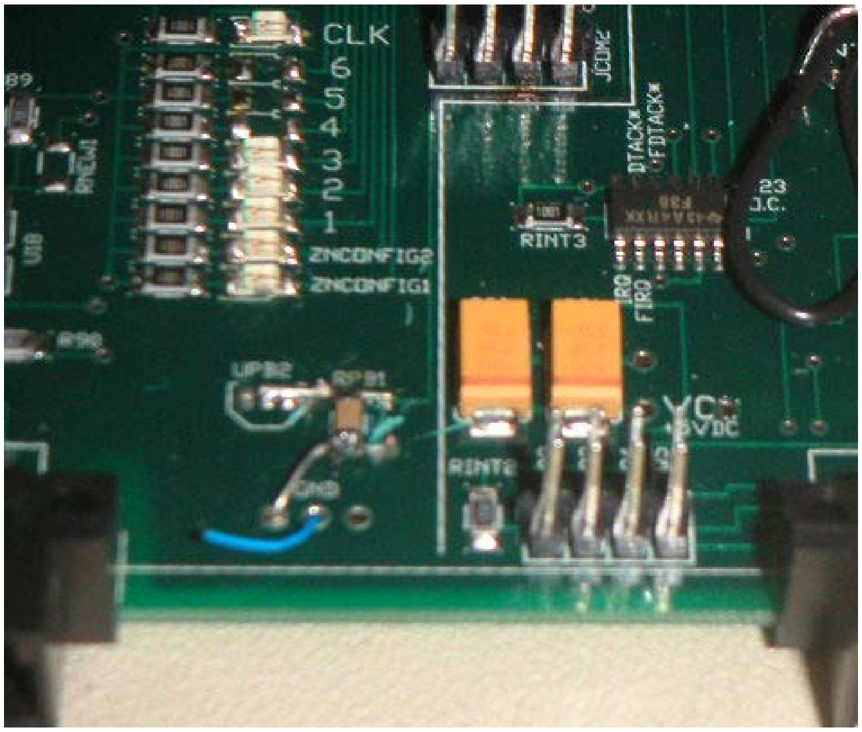
Switch 4 sw133 normally hi. Hi means auto-reboot, low means use sw134 to reboot.



6.4 JUMPERS.

Jumpers have no effect on my VME boards, but are necessary for other VME cards placed further to the right of mine in the VME crate. For initial tests, I recommend a mostly empty crate, so these are not issues. But as the crate fills up, they become important. The jumpers must be installed to pass the VME bus grants, and interrupt acknowledge to VME cards to the right of mine.

On PASSTH6, jumpers exist to pass VME bus grants, and interrupt acknowledge, on to cards to the right of mine in the VME crate. Jumpers on lower right of PCB, near VME connector. Normally all are jumpered. In photo below, none are jumpered.



On PASSTH4, these connections must be made by hand. (See Steve, who will solder on a small wire onto the connector, a simple “jumper” to allow cards to the right to receive signals.

On FIBER2B the VME bus grant and interrupt acknowledge signals are passed by default, by small traces on the PCB. No action necessary for normal operation.

On FIBER2 these connections must be made by hand. (See Steve, who will solder on a small wire onto the connector, a simple “jumper” to allow cards to the right to receive signals.

(7) INSTALLING A SYSTEM AT AN APS BEAMLINE.

7.1 COSTS As I understand my job per APS management, our primary mission is to support the CAT beamline personnel. This electronics is thus made available for minimal costs of manufacture about $2000 - $3000 total, for the 4- to 16-channel system. This breaks down to $1500 per VME board, and about $600 each for an ADCMOD2,3 board. Thus the incremental cost of adding four more electrometers is $600.

7.2 HARDWARE MAINTENANCE/ SUPPORT. We in APS will keep your system working on into the future. This is not a trivial undertaking. We have a small staff, but it is more than one person. Our hope is that a number of beamlines and sectors will standardize on how to read the data out, and this will help a lot with configuration management, and keeping spare parts. Support will **not** be 24 hours/day, 7 days per week, but more like “working hours”. The best way to have a reliable system is to have some backup channels. It will take us a while to get there.

We follow ASD quality control and documentation guidelines, again not trivial.

7.3 INSTALLATION ISSUES.

FIBER OPTICS.
Each adc module communicates over a pair of fibers. How long is your length of fiber? We buy them commercially made now, maybe $50 each, for example from Newark. Longer cables can be purchased from Fiber Instruments Sales, Inc. 800/445-2901, 62.5/125 micron multimode fiber, with ST connector, a given length long. Part number for 14 meter, 4 fibers in bundle is C288M14FIS/04, $60.

PIN DIODE HOLDERS.
I need to work with each sector to mount the PIN diodes, and make sure the wires are brought out. Then if it is different I can fabricate a new PCB (cost about $400 qty 10). Please get me this mechanical design info, or use the existing ones. I already have several designs for the holder for the PIN diode: See sector 2 for an Autocad drawing for two designs .

I will work with people who need ultra high vacuum (e.g. Sector 4ID) Typically they will guide me on how this can be implemented. Otherwise people hold the diodes on with superglue, 5 minute epoxy etc.

FOIL HOLDER.
Using diode5 or diode6 design, some sectors just want to have another pcb mounted to the first, and onto it glue down the foil. These "blanks" have no diodes on them, but just simply hold the foil. They can be mounted on stand-offsontodiode5 or diode6. Typical stand off distance is 1 cm. Will you get your own foils? I can tell you the vendors, but typically I don't worry about the actual foil. I learn much of this from Randy Alkire at sector 19. Other sectors will coat kapton with metal.

WIRING TO PIN DIODES.
I can supply the hookup wires from the pindiodes to the adc module, though it is just ordinary wire. I have a minor preference for wire colors (black, brown, red, orange) to try to "standardize". At sector 32 I was able to tightly twist 2 wires together and get good noise levels even with up to 2 meters of twisted pair, something I did not expect at the time. Of course you can also split out, and run 2 pin diodes at one place, 2 at another etc. The PIN diodes are completely floating. Connectors are always a concern. For the designs I consider "standard”, I have a stock of connector. Don't be surprised if we have to order stuff from Newark.

CAUTION: PIN DIODES AND LIGHT.

The pin diodes respond very well to visible light. For example if you have a vacuum gauge it will have an orange glowing filament. If the pin diode can see it, it will happily put out lots of false signal.

POWER SUPPLY.
The ADCMODs require power. Typically this is about 6 VDC, about 0.48 Amps per module. The exact voltage is not critical, as I clean it up. People might want to start out with a lab supply such as a BK Precision 1760 $600 from Digikey, which can run 3 modules.

When the systems are powered off of 6 volt batteries (e.g. lead acid cells) then the noise drops. I am always trying to deploy an automatic battery re-charger, to toggle between one cell powering and another cell charging.

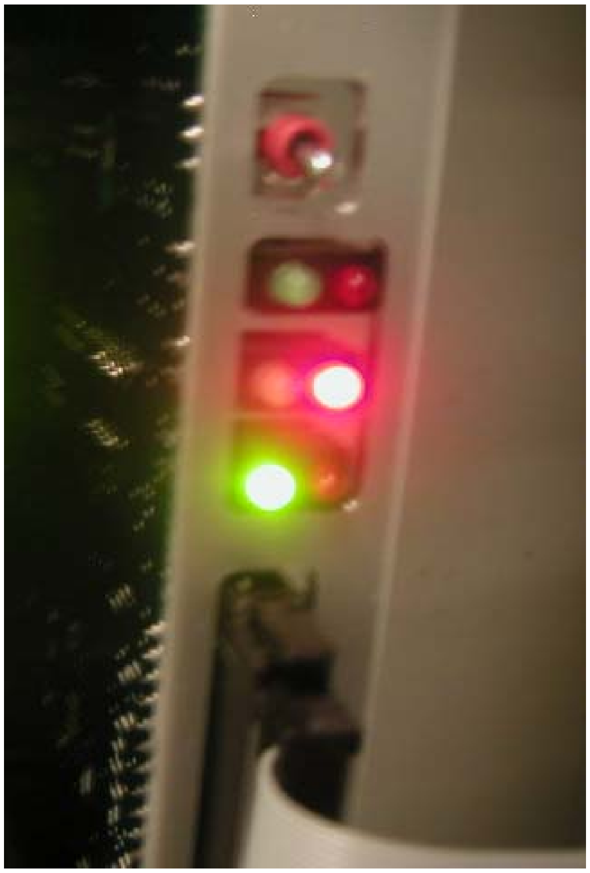
7.4 GETTING STARTED WITH YOUR SYSTEM OUT OF BOX.

7.4.1 This section gives a brief discussion of “day one”. After this we’ll have to talk, or you read more of this guide. You don’t have to keep things in the configuration described but it helps to see a sign of life.

7.4.2 TO TEST VME WRITES TO THE SYSTEM:

(1) Insert the FIBER (version 2, or 2A) card and the PASSTH (version 4 or 6) card into the VME rack. Perhaps place them with one empty slot in between them just to make sure nothing shorts out at first. It would be nice at first if the only other VME card is the slot 1 controller and perhaps a VME bus monitor, to avoid any bus contention issues.
(2) Install the VSB cable between the two boards, on the back of the VME backplane. It is keyed to only fit in one orientation.

(3) Power up the VME rack.
(4) The PASSTH4 or 6 board should light up LEDs. The LED’s for SHIFT CLK (red), and for CLKIN0 (green) should be ON. Others might be as well, until re-booting below.)



(5) The FIBER2 board should also light up top 2 LEDs on each cluster of 4. Photo
(6) .

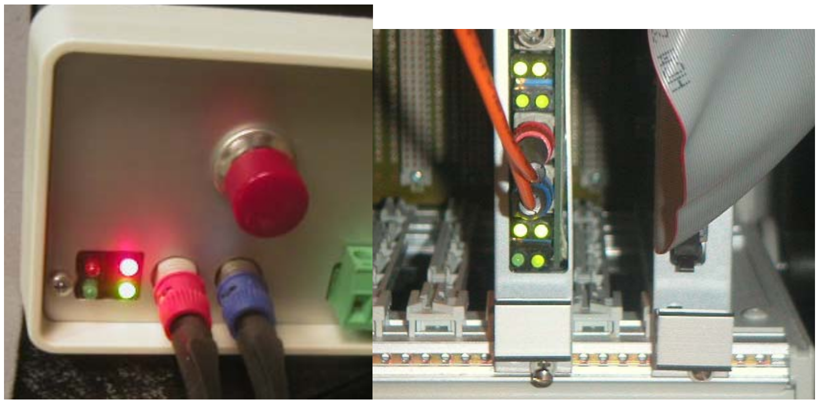
(7) Try to write to the PASSTH card. If there is a successful DTACK*, the red DTACK PASSTH4 LED will flash.
Xxxxxxx Upper right, red, in photo.

If this does not flash, either you are writing to the wrong VME address, or you have not correctly set the VME address DIP switches on the PASSTH board. You must fix this before proceeding.

7.4.3 TO TEST COMMUNICATION WITH AN ADCMOD2/3.

Now you must connect up an ADC electrometer module. It would be best to start with the ADCMOD right next to the VME crate, with a short piece of patch cord fiber.

(1) Connect up the fiber – transmit to receive and visa versa. In other words, a dark gray fiber connector gets connected to a light grey one and visa versa.
(2) Connect up power to the ADCMOD. This is especially easy with ADCMOD3, as it uses a terminal block.
(3) Power up the ADCMOD. It should take about 0.4 amps. If greatly more, shut it all off!
(4) If the VME cards are running, the ADCMOD's two LED's will come on: TRANSMITTED MANCHESTER TO FIBER2 and RECEIVED MANCHESTER FROM FIBER2
(5) Also note that now 3 LED's on for FIBER2 (the two top LED's red/grn, and the bottom right/green one).
(6) Briefly push up the toggle switch on the PASSTH4/6 card. Let it go. The FIBER2 LED (green/front) for "GOOD TO GO" should light. You are now successfully communicating with the ADCMOD via the fiber.

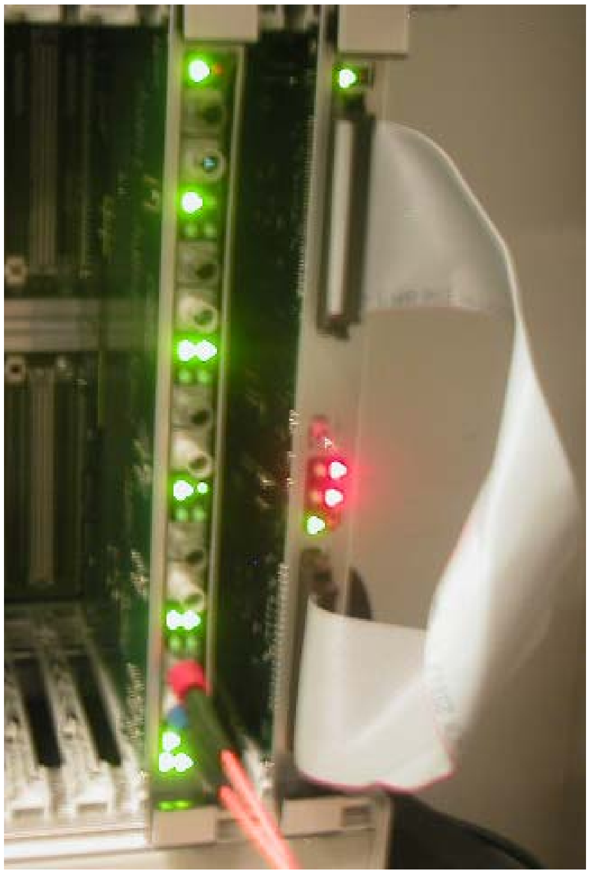


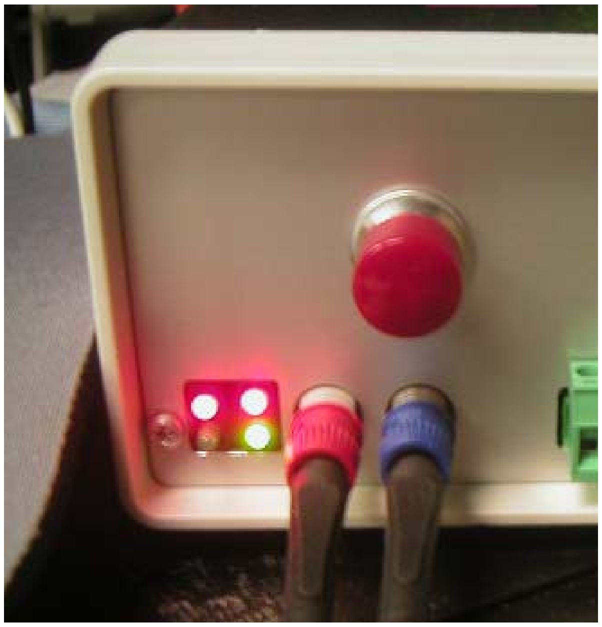
ADCMOD upon power up, but before it starts to transmit data. After the six steps above, 3 FIBER2 lights would be on: the bottom right, now flickering, would be solid. In this photo FIBER2 has only top 2 lights glowing because ADCMOD has not yet been turned on.

7.4.4 TO READ DATA FROM THE SYSTEM.

(1) It never hurts to again toggle the switch on PASSTH4, 6 just to reset the system.
(2) Run the software scripts discussed – that is, write to the four VME addresses.. It will help to watch each VME-setup-WRITE go out, using a bus monitor.
(3) Just as with the TO TEST VME WRITES TO THE SYSTEM description, as each "SEND" command goes out, watch that the serial data does in fact go out. The ADCMOD's DATA FROM FIBER2 TO ADCMOD2 LED should also briefly flash at this time. It is somewhat dim – look carefully. (Bottom left/green LED).
(4) When the "SEND" command for the "GO" command is sent, the ADCMOD should be told to send continuous data. The bottom left (red) fiber channel LED display DATA FROM ADCMOD TO FIBER2 LED should now be lit solid.

If there are problems, it is likely because of initialization. Here are the lights with all working, data flowing including all 4 of cluster on a FIBER card, on the in-use fiber channel, and 3 on the PASSTH card's set of 6.





After step (4) above, the VME receiving continuous data via the orange fiber, to the bottom channel 0 of the FIBER2. The ADCMOD’s top left light indicates data is flowing from it.

7.4.5. Remote re-booting. The ADCMOD3 cards are re-booted by sending the re-boot command, or by pushing the re-boot push-button. The ADCMOD2 cards can only be re-booted via the push button. The two VME cards are re-booted by pushing the toggle switch manually. This signal can also be wired into the VME crate’s SYSRESET* line, so that it is possible to re-boot the VME cards remotely. Typically not the initial configuration, see Steve.

8.0 DETAILED EXPLANATION OF THE COMMANDS TO THE ADCMOD2 or 3

Set up all of the pindiode ADC modules via the fiber optic communication line, to provide the correct data, and to then command the ADC modules to begin taking data.

I write informative comments after a ! symbol. Delete them for your actual script.   All numbers are hexidecimal.

8.1.1 RANGE

```
a000 2range               !command #1 to adc module
0001 3range               !command 0001 sets the adc's gain range
0000 4range               !gain being set to 0000
```

Allowed values for adc gain range:

a000 0001 0rrr set the gain of the ADC unit.
rrr can be

```
000  least gain range , refer to it as gain = 1, assume external
feedback capacitor of 220 pf (see also section 3.1.1).
001  gain = 220 / (1*12.5) = 17.6
010  gain = 220 / (2*12.5) = 8.80
011  gain = 220 / (3*12.5) = 5.87
100  gain = 220 / (4*12.5) = 4.40
101  gain = 220 / (5*12.5) = 3.52
110  gain = 220 / (6*12.5) = 2.93
111  gain = 220 / (7*12.5) = 2.51
```

The gain **is calibrated absolutely** under the following conditions: With gain range set to 000 (command a000 0001 0000), the conversion time set to 0200 (command a000 0005 0200), a 220 pf feedback capacitor and 10 nA going into the system from a Keithley 263 calibration source, then the ADC counts read 14260 (digital, approx). There is a baseline of approximately 4096 counts. So the signal is 14260 - 4096 = 10164 digital counts in 820 us. **Scaling the current, 1 pA would give about 1 count (220 pf CFB, 820 us).**

The external feedback capacitor (range 000) can be changed. It has been set as high as 1000 pf. The question then becomes is the ADC still linear. A quick check shows good performance, but more testing is necessary.

Currently a software command will set all ADCMOD2 modules to the same gain – there is no present way to address them individually. Another firmware improvement that is likely needed. This improvement will come soon.

8.1.2 NUMBER OF DATA POINTS BACK FROM ADCMOD - digital storage scope mode only

```
a000 6pulse              !command #2 to adc module
0006 7pulse              !command 0006 sets number of data points
1000 8pulse              !here we ask for 1000 data points
```

8.1.3 PERIOD

```
a000 10period            !command #3 to adc module
0007 11period            !command 0007 is currently not supported
ffff 12period            !so leave this exactly as shown
```

8.1.4 CONVERSION TIME – conv command

```
a000 14conv              !command #4 to adc module
0005 15conv              !command 0005 sets conversion time
0280 16conv              !here we set to 0280 (comment 5)
```

So handy rule of thumb:
With conversion time 0200 hex (820us), range = 000, 1 count = 1 pA.

Command a000 0005 xxxx sets the conversion time (for a "ping", "pong" is the same). This number, specified by the xxxx is converted to seconds as follows:

```
microseconds = xxxx (hex --> decimal) * 1.6 + 0.6

example:  xxxx = 0280
0280 in hex = 2*16^2 + 8*16^1 + 0
            = 2*256  + 8*16   + 0
            = 620 decimal
time (microseconds) = (620 * 1.6) + 0.6
                    = 992.6
```

The allowed range for this parameter is in hex

```
2000 >= xxxx >= 180     (hex)
```

thus the conversion time is presently limited to
13.107 milliseconds >= conversion time >= 615 microseconds.

Thus the fastest data currently comes at 1/615 microseconds =
1.6 kilosamples/second.

9.0 DESIGN FILES.

There exist files of the following types: OrCAD, AUTOCAD, Altera, PCB-Protel Trax, and NC Milling. See the **aa-Electrometer** file folder, for everything, except the Altera files. See the aaa-altera master (date) folder for the altera files, you just have to find the latest date.

9.1 ADCMOD. Stored in **adcmod** directory. There are OrCAD schematics for ADCMOD2, ADCMOD3. (e.g. ADCMOD2_TOTAL.DSN, or ADCMOD3.DSN) They are the sum of two parts, the adc part itself, and the fast fiber part. The fast fiber part is the fiber optic link, transmit and receive. It is the same here and on the FIBER2, 2A cards.

There are Trax/Protel PCB files for ADCMOD2, ADCMOD3 (e.g. ADC3.PCB) Again these could be considered a paneling of the adc part and the fast fiber part. I used the exact same pcb layout for the fiber link here and on the FIBER card’s daughter board (e.g. motor4.pcb).

There are Altera folders for the ADCMOD’s. ADCMOD2 has only one altera chip on it. This design is the same as the similar ADCMOD3 altera. No change. The root GDF file is adcmod.gdf. ADCMOD3 gained a second altera chip, stored in file folder ADCMOD_NEW. The root GDF file is again adcmod.gdf. I depend on the Windows file folders to keep these straight. In both cases, the relevant pof download file is adcmod.pof, again depending on the Windows file folder to keep straight. Also note that ADCMOD2 uses a 10K30 chip, ADCMOD3 uses 2 10K40 chips. This must be set in the Altera compile option.

There are autocad drawings for the face plates for the Lansing boxes. See WBS below.

9.2 FIBER2. Stored in **fiberN** directory. There are OrCAD schematics for fiber2, and 2A. They are only slightly different, and there are comments.

There is a Protel/TRAX PCB file for fiber2, and for fiber2A.

There is an Altera file for FIBER. Windows folder **fiber2b_10K40**. FIBER2, and 2A are the exact same design, root is fiber.gdf. The only difference is that FIBER2 uses a 10K30 chip, FIBER2A uses a 10K40 chip. This must be set in the Altera compile option.

The fiber cards have daughter boards. See file folder: **fiberN** daughter board motorN for the PCB files. Motor4 is the latest. Essentially these PCB’s are a panel of 5 fast fiber circuits. See the file folder **fastfib** for the schematics of the fast fiber circuits.

9.3--PASSTH4 and 6 Stored in the **passthN** folder. If you look at these PCB's you will see that they are very similar on the left (VSB) and right (VME) sides. Thus this PCB was a paneling of two schematics, two PCB files. There are OrCAD schematics for the VME side and the VSB side of the PASSTH's. There is also a combined schematic for both, PASSTH4.dsn, PASSTH6.dsn.

The altera designs for passth6 VSB side are passth2_10K40_vsb folder, passth2.GDF.

The altera designs for passth6 VME side are passth2_10K40_vme folder, passth2h.GDF.

9.4 WBS-LDN numbering. Some WBS-LDN numbers have been assigned, especially for the AUTOCAD files: WBS = 1.4.3, LDN numbers:

311100	passth6_vme_panel.dwg
311101	fiber2_vme_Panel2.dwg
311102	adcmod_2_3_lansing_face.dwg
311103	lansing_box.dwg
311104	fiber2_motor4_DB.dwg
311105	ORCAD block diagram of electrometer 5/2006

ORCAD LDN numbers are assigned as follows, but this work is less complete:

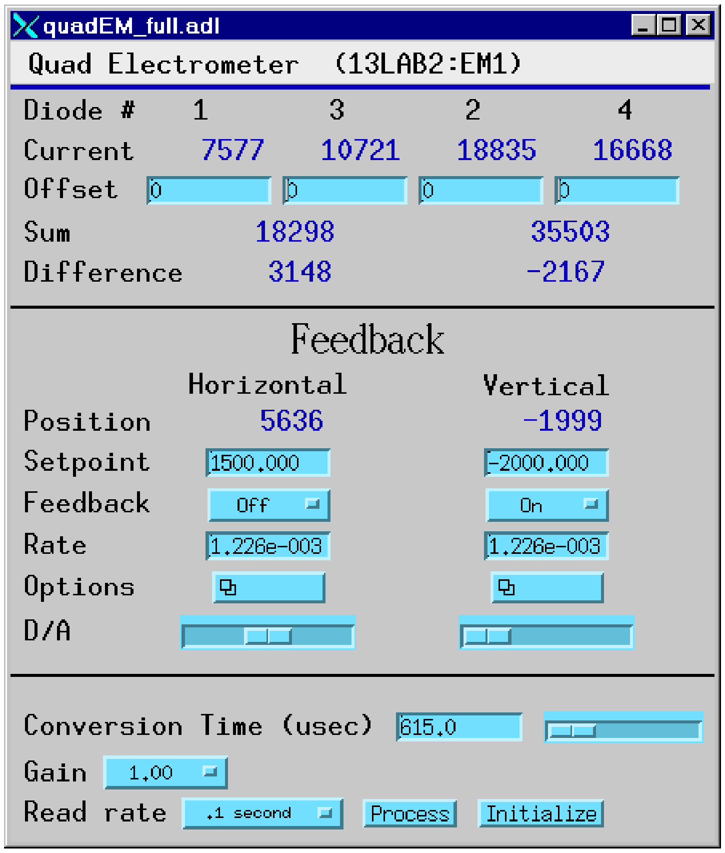
311106	passth6 orcad
311107	fiber2b orcad
311108	adcmod3 orcad
311109	fast fiber orcad

9.5 NC Mill files for front panels. Bob Conley in the shops has made the front panels. He gives a number for the NC milling machine file. I try to write this on the AUTOCAD drawing, but in general they are clustered around my badge number.

10.0 EPICS INTERFACE.

[Reference for this entire section: A PowerPoint presentation by Mark Rivers, EPICS Detector and Feedback Software.]

Mark Rivers (GSECARS, sector 13) has written EPICs drivers for the electrometer, called quadEM. This software is maintained by APS/AOD/BCDA group, who usually are the ones to set things up at other beamlines. Here is a typical EPICs screen. This runs under EPICS 3.14



Quad Electrometer (13LAB2:EM1)

Diode #	1	3	2	4
Current	7577	10721	18835	16668
Offset	0	0	0	0
Sum	18298		35503	
Difference	3148		-2167	

Feedback

	Horizontal	Vertical
Position	5636	-1999
Setpoint	1500.000	-2000.000
Feedback	Off	On
Rate	1.226e-003	1.226e-003
Options		
D/A		

Conversion Time (usec) 615.0

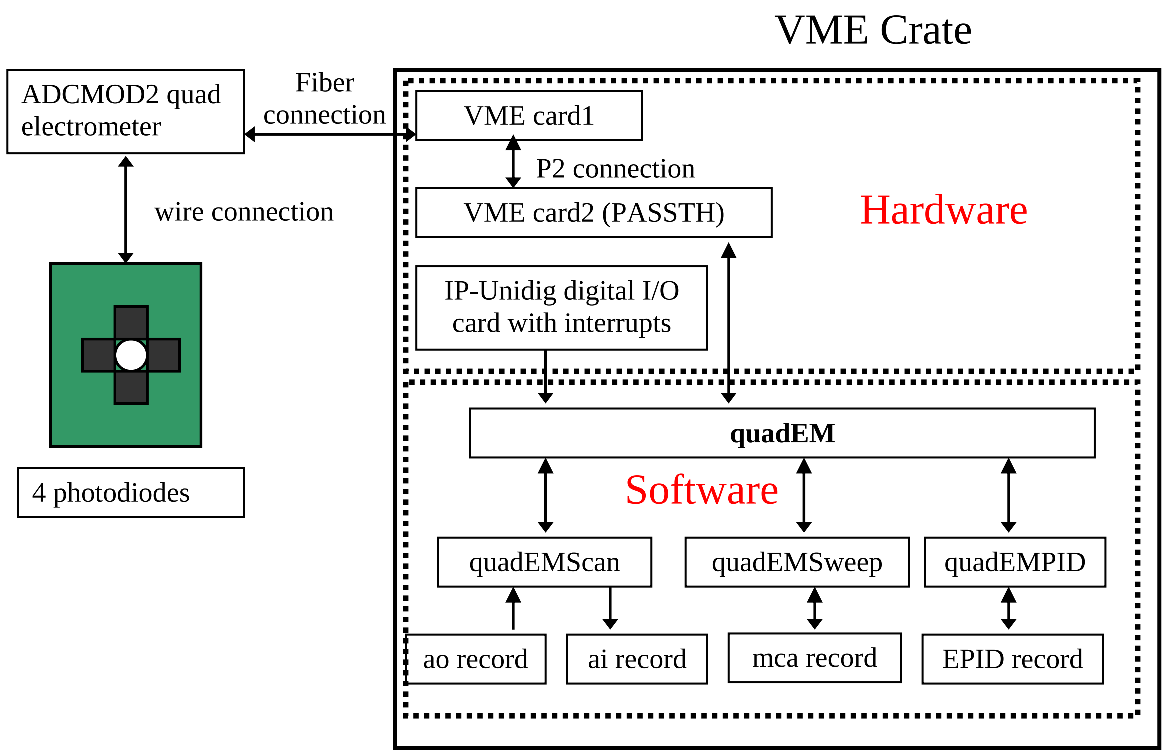
Gain 1.00

Read rate .1 second

- Data comes from the electrometer into the VME system at up to 815Hz.
- Current VME boards do not support interrupts
- They do put out a TTL pulse when new data arrives, up to 815Hz.
- This pulse is input to an IP-Unidig Industry Pack I/O module, which does support interrupts.
- IP-Unidig interrupt routines calls the function to read the quad electrometer VME board.
- On each interrupt up to 3 quadEM functions are called
  - quadEMScan averages the current reading and returns averaged readings to EPICS “analog input” records
  - quadEMSweep puts the current reading into an array for an EPICS “mca” record. Performs averaging if the channel advance time is less than the electrometer clock rate.
  - quadEMPID uses the current reading to perform fast feedback via a Systran IP DAC. Performs averaging if the feedback rate is less than the electrometer clock rate.

- Feedback of the pitch and roll of monochromator crystals based on the beam position in the beamline or experimental station.
- Feedback on mirror pitch for stabilizing the position of the beam downstream of a focusing mirror.
- In-vacuum fluorescent foils allow I0, beam position, and energy calibration to always be available
- Replaces:
  - 4 SRS570 current amplifiers
  - 4 ADCs, or 4 V/F converters and 4 scaler channels [per ADCMOD2/3]

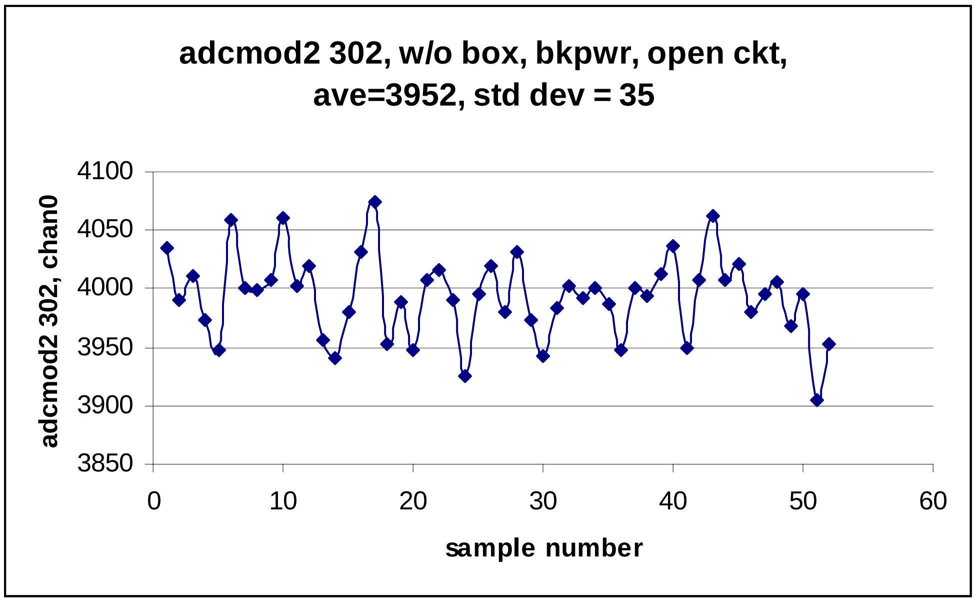
System Architecture



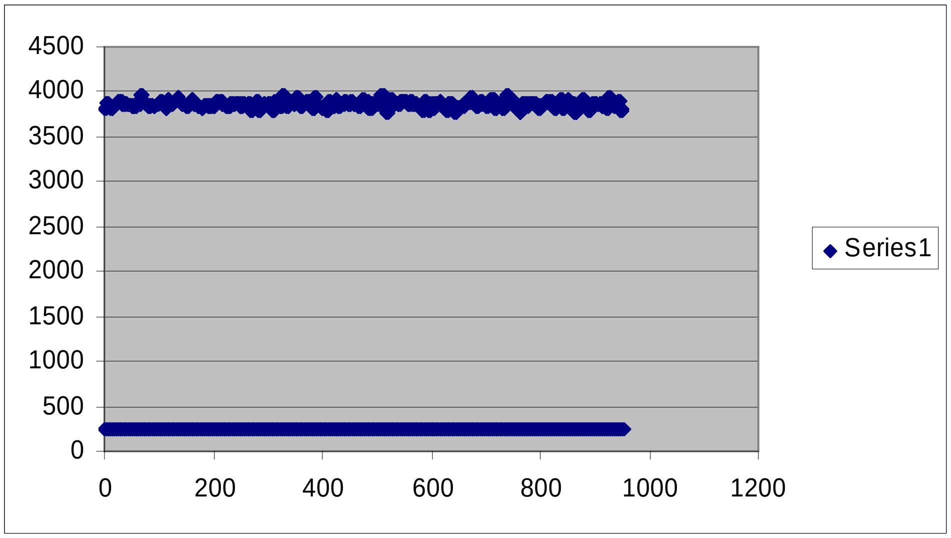
11. APPENDIX. SAMPLE DATA, DEBUGGING INFORMATION (FOR STEVE).

10.1typical data.

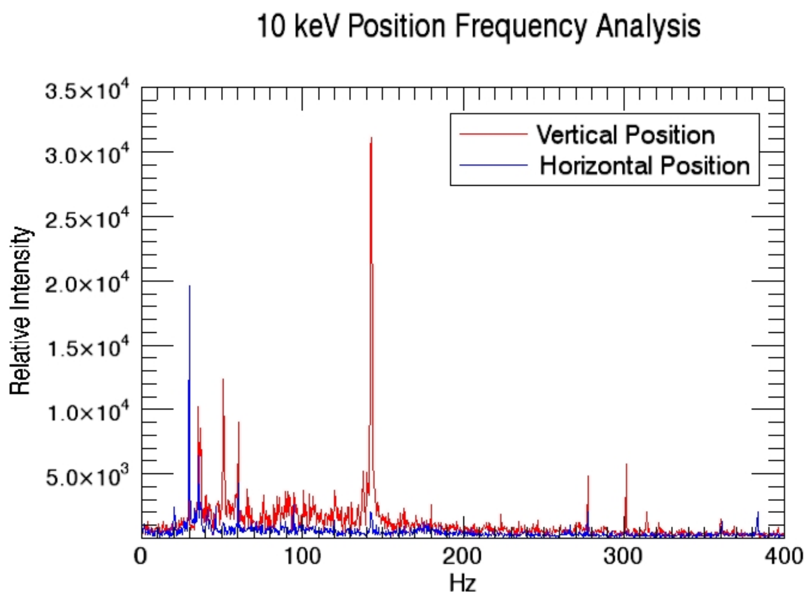
This data was taken using an adcmod2 unit powered by a BK precision power supply -- battery power would be better. The adcmod2 was sitting out bare on a lab bench, in a lab. There is no input connection – no input diodes. This data is for reference only, I expect to continue to improve the noise, which here is about 30 pA RMS. Channel 0, signal vs. sample number. CFB = 220 pf. I have little doubt this can be improved.



The two traces below show the most significant (Table 5.3A), around 256 and least significant bits, around 4096 = 256 * 4. (Table 5.3B) (See discussion section 5.3).



Jim Ciston and Mark Rivers at sector 13 applied the BPM to measure the vibrations, and mechanical stability of their beam. They used 800Hz data. This electrometer system is very low noise at quite good bandwidth. Mark has an EPICS PID control loop system for feedback stabilization of the beam. [Ref: Performance Analysis of a Backscattering Fluorescence BPM on 13ID, James Ciston 12/22/2003, internal report to Mark etc.]



10.2 EMBEDDED DE-BUGGING CODES.

There are a number of bits of data useful for debugging. These include parity bits for the fiber serial transfer.

```
1 adc module, 1 encoder communication module
a-a-d- a-a-d- a-a-d a-a-d
      ping        pong
i.e. quantity 24 32 bit words going into memory per scripted pulse
```

See also "Interpretation of the Data, comment 6. The 8 a- 's listed just now each are tagged by 8 unique tags. All would come in from the same fiber optic channel.

```
2 adc modules, 1 encoder communication module
a-a-a-a-d- a-a-a-a-d-   a-a-a-a-d- a-a-a-a-d-
         ping                 pong
i.e. quantity 40 32 bit words going into memory per scripted "pulse"

3 adc modules, 1 encoder communication module
a-a-a-a-a-a-d- a-a-a-a-a-a-d-     a-a-a-a-a-a-d- a-a-a-a-a-a-d-
         ping                           pong
i.e. quantity 56 32 bit words going into memory per scripted "pulse"
```

10.1 INTERPRETING THE INTERNAL SERIAL DATA. MAINLY FOR DIGITAL STORAGE SCOPE MODE. HELPFUL FOR ME TO DEBUG.

Commands out to the ADC module. The commands out are packaged into a 48 bit serial packet. Their function:

47:44	Device type (= hex a, 4 bits, binary 1010)
43:37	Address (unused at present, can select a adcmod2 unit)
36:21	Unused
20:16	Command (5 bits)
15:00	Data for command (16 bits)

48 bits of data back from the ADC module. Bit 47:00, where bit 47 arrives first in time, in the serial stream.

47:44	Device type (= hex a, 4 bits, binary 1010)
43:40	always set to binary 0001
39:37	binary 111
36	a test mode for the DDC112, see its data sheet. When bit = 0, normal charge amplifier mode, when test bit = 1, we are in DDC112 test mode.
35:32	the switch pattern on the ADCMOD2, LSB switch is right most. Thus order 35:32 is SW160,SW159,SW158,SW157.
31:29	Range. The data is embedded with the range the charge amplifier was using at the conversion

28	bit =0 for amplifier A, bit=1 for amplifier B (see ddc112 sheet)
27	ADC conv cycle bit=1 for first/integrate A, bit=0 for second/integrate B. See ddc112 data sheet, FIGURE 2.
26	ADC input pin. 1=integrated circuit pin 28 (input 2), 0=integrated circuit pin 1 (input 1)
25	parity bit for data 19:16
24	parity bit for data 15:12
23	parity bit for data 11:8
22	parity bit for data 7:4
21	parity bit for data 3:0
20	always 0
19:00	the data, MSB first, LSB last

Data added by fiber2 card. Because the VME transfer is 32 bits, and so far we only have 48 bits, we can add 16 more bits as the data goes through FIBER2, again to help with debugging. This data is added after the LSB 00 bit, and pads the data out to 64 bits, or two 32 bit transfers. The first 32 bit transfer then is bits 47:16 above. The second 32 bit transfer is then bits 15:00, plus the 16 fiber2 pad bits now discussed:

Fiber2 pad bits (16 bits total)

Pad[15:4] A 12 bit counter, incremented each time data comes into this fiber channel. It just rolls over when it overflows.
Pad[3:0] which fiber2 channel was used (channels are numbered 000 (bottom pair on VME card), 001,010,011,110. Channel 110 is not used at this time.

10.3 SAMPLE OF DATA READ OUT, DIGITAL STORAGE SCOPE MODE. I include here a sample readout of the PINDIODE system, with 2 ADC modules connected. The ADC modules are connected to fiber optic channels 1 and 4. The encoder communications module is connected to fiber optics channel 3. We can label the columns of data (printed out in c by a 10%x format) as follows:

```
     LONG    ADDR      CONTENTS              COMMENTS
     WORD    VME
     NUMBER
```

First Note: recall in the tutorial script the lines

```
2000 VSBstatus         !address where DC1 status register goes
0000 status--          !this status register upon success reads 20001
```

which stated that the vme status register should be address 2000 0000

```
        0   20000000      20001           !VME transfer status register
        1   20000004   ffffffff           !junk
        2   20000008   fffffdff           !this area of memory is never
        3   2000000c   fdffffff           !written to.
        4   20000010   7ff7efff
        5   20000014   fffffffe
        6   20000018   dffffbef
        7   2000001c   ffffffbf           !end of junk
```

Second Note: recall in the tutorial script the lines:
2000 VSBaddr2000   !higher order address of where data is going
0020 VSBaddr0020   !lower order VME/VSB address of where data is going.

```
        8   20000020   a1ee0c00          !begin data #1 (first pulse)
        9   20000024         11          !note address 2000 0020
        a   20000028   a1ef0c2f          !was set by these lines
        b   2000002c   fffc0014
        c   20000030   a1ee1c00
        d   20000034         21
        e   20000038   a1ef1c40
        f   2000003c   698a0024
       10   20000040   d0fe0017
       11   20000044   60040013
       12   20000048   a1ee0768
       13   2000004c   2a2c0031
       14   20000050   a1ef07ac
       15   20000054   5d880034
       16   20000058   a1ee1400
       17   2000005c         41
       18   20000060   a1ef17c1
       19   20000064   1edf0044
       1a   20000068   d0fc0017
       1b   2000006c   60040023
                                        !blank line inserted for
clarity
       1c   20000070   a1ee0980
       1d   20000074   29800051         !adc still "settling down"
       1e   20000078   a1ef0940
       1f   2000007c   330c0054
       20   20000080   a1ee1800
       21   20000084         61
       22   20000088   a1ef18c0
       23   2000008c    7400064
       24   20000090   d0fc0017
       25   20000094   60050033
       26   20000098   a1ee0000
       27   2000009c   12cc0071
       28   200000a0   a1ef0180
       29   200000a4   23400074
       2a   200000a8   a1ee1000
       2b   200000ac         81
       2c   200000b0   a1ef1020
       2d   200000b4    5980084
       2e   200000b8   d0fc0017      !encoder is at 176005
       2f   200000bc   60050043      !end of pulse 1's data
                                     !blank line inserted for clarity
       30   200000c0   a1ee0dc0      pulse #2
       31   200000c4   3ae00091
       32   200000c8   a1ef0c80
       33   200000cc   6f940094
```

```
        34   200000d0   a1ee1c80
        35   200000d4    ecd00a1
        36   200000d8   a1ef1c40
        37   200000dc    fec00a4
        38   200000e0   d0fc0017
        39   200000e4   60040053
        3a   200000e8   a1ee0400
        3b   200000ec         b1
        3c   200000f0   a1ef0560
        3d   200000f4   52fc00b4
        3e   200000f8   a1ee14a0
        3f   200000fc    ef000c1
        40   20000100   a1ef1540
        41   20000104   10d300c4
        42   20000108   d0fe0017
        43   2000010c   60040063
                                           !blank line inserted for
clarity
        44   20000110   a1ee0980
        45   20000114   378c00d1
        46   20000118   a1ef0900
        47   2000011c   3e0c00d4
        48   20000120   a1ee18a0
        49   20000124    e3300e1
        4a   20000128   a1ef1820
        4b   2000012c    fa200e4
        4c   20000130   d0fe0017
        4d   20000134   60050073
        4e   20000138   a1ee0160
        4f   2000013c   411c00f1
        50   20000140   a1ef01a0
        51   20000144   3d9000f4
        52   20000148   a1ee10c0
        53   2000014c    db20101
        54   20000150   a1ef1080
        55   20000154    ef30104
        56   20000158   d0fe0017
        57   2000015c   60040083

        58   20000160   a1ee0d80         pulse #3
        59   20000164   3cbc0111
        5a   20000168   a1ef0da0
        5b   2000016c   50100114
        5c   20000170   a1ee1c60
        5d   20000174    fbc0121
        5e   20000178   a1ef1c60
        5f   2000017c    fb20124
        60   20000180   d0fe0017
        61   20000184   60050093
        62   20000188   a1ee0400
        63   2000018c        131       !don't worry,I fixed this
later...
        64   20000190   a1ef05e0
        65   20000194   44480134
        66   20000198   a1ee1400
        67   2000019c    ffc0141
        68   200001a0   a1ef1560
```

```
69   200001a4   10180144
6a   200001a8   d0fe0017
6b   200001ac   600500a3
6c   200001b0   a1ee0940
6d   200001b4   384c0151
6e   200001b8   a1ef0920
6f   200001bc   3e700154
70   200001c0   a1ee18c0
71   200001c4    e230161
72   200001c8   a1ef1860
73   200001cc    fb20164
74   200001d0   d0fc0017
75   200001d4   600400b3
76   200001d8   a1ee0180
77   200001dc   42000171
78   200001e0   a1ef0100
79   200001e4   3e000174
7a   200001e8   a1ee10c0
7b   200001ec    dd20181
7c   200001f0   a1ef1020
7d   200001f4    f570184
7e   200001f8   d0fc0017
7f   200001fc   600400c3
80   20000200   a1ee0d60
81   20000204   3e300191
82   20000208   a1ef0d20
83   2000020c   47480194
84   20000210   a1ee1c60
85   20000214    fbc01a1
86   20000218   a1ef1c00
87   2000021c    fc601a4
88   20000220   d0fc0017
89   20000224   600400d3
8a   20000228   a1ee0400
8b   2000022c        1b1
8c   20000230   a1ef05c0
8d   20000234   422c01b4
8e   20000238   a1ee1540
8f   2000023c   102201c1
90   20000240   a1ef1540
91   20000244   102301c4
92   20000248   d0fc0017
93   2000024c   600400e3
94   20000250   a1ee0940
95   20000254   391c01d1
96   20000258   a1ef09c0
97   2000025c   3c5401d4
98   20000260   a1ee1880
99   20000264    e6301e1
9a   20000268   a1ef1840
9b   2000026c    f2901e4
9c   20000270   d0fc0017
9d   20000274   600500f3
9e   20000278   a1ee0120
9f   2000027c   414001f1
a0   20000280   a1ef01a0
a1   20000284   3d5c01f4
```

```
        a2   20000288   a1ee1080
        a3   2000028c    df10201
        a4   20000290   a1ef1020
        a5   20000294    f620204
        a6   20000298   d0fe0017
        a7   2000029c   60050103
        a8   200002a0   a1ee0d40
        a9   200002a4   3e500211
        aa   200002a8   a1ef0de0
        ab   200002ac   44500214
        ac   200002b0   a1ee1d40
        ad   200002b4   10100221
        ae   200002b8   a1ef1c60
        af   200002bc    f470224
        b0   200002c0   d0fe0017
        b1   200002c4   60040113
        b2   200002c8   a1ee0400
        b3   200002cc        231
        b4   200002d0   a1ef0560
        b5   200002d4   3e840234
        b6   200002d8   a1ee1540
        b7   200002dc   104d0241
        b8   200002e0   a1ef1460
        b9   200002e4    f770244
        ba   200002e8   d0fe0017
        bb   200002ec   60040123
        bc   200002f0   a1ee0920
        bd   200002f4   392c0251
        be   200002f8   a1ef0900
        bf   200002fc   38c00254
        c0   20000300   a1ee18c0
        c1   20000304    e430261
        c2   20000308   a1ef1880
        c3   2000030c    e550264
        c4   20000310   d0fe0017
        c5   20000314   60040133
        c6   20000318   a1ee0160
        c7   2000031c   41c00271
        c8   20000320   a1ef0160
        c9   20000324   38440274
        ca   20000328   a1ee10c0
        cb   2000032c    ddf0281
        cc   20000330   a1ef1080
        cd   20000334    e0c0284


        etc. etc. etc
```

10.4 HOW TO INTERPRET THE DIGITAL STORAGE SCOPE MODE DATA.

I have not finished de-bugging the parity, so don't worry about it yet.

Take the last adc data given above:

```
a1ef 1080
 e0c 0284
```

```
parse it:
a = comes from the binary pattern 1010,indicates an adc module
1 = comes from the binary pattern 0001, ignore this
e comes from a binary pattern 1110
        in general nnnn

        nnnn = 1 1 1 test,  test usually is 0, so nnnn = 1110
        the test bit goes hi when we run a specific diagnostic on the
adcmod,
        the "Burr Brown DDC112 ADC test" diagnostic.
        See DDC112 data sheet Figure 8.
        (When test is turned on by a000 000f vvvv, when vvvv = 7,
        adc reads 300,000 decimal.  when vvvv = 3, adc reads 150,000
decimal.)
        clear test with a000 0002 0000.

f comes from a binary pattern 1111
        in general pppp
        pppp = switch switch switch switch
             = switch settings on the adcmod (a tag for this particular
             module, set by the human using switches)

1 comes from a binary pattern 0001
        in general qqqq
        qqqq = range range range tag
        e.g. if you were using range 000 then this should be qqq = 000
        e.g. if you were using range 111 then this should be qqq = 111
        see Burr Brown DDC112 data sheet table I, page 7.
        the last bit, the 4th q bit "tag", indicates which of the two
DDC112
        burr brown chips on the printed circuit board is being read
        0=DOUTA=U5, 1=DOUTB=U8 (see comment 6)

0 comes from a binary pattern 0000
        in general ssss
        ssss =  1st/2nd_readout  ping/pong  parity19:16 parity15:12
        (see comment 6 below for explanation of 1st/2nd, ping/pong,
        as these are the second and third "tag" bits)

8 comes from a binary pattern 1000
        in general tttt
        tttt = parity11:8 parity7:4 parity3:0 notused

Actual data comes next and is 00e0c note the missing leading 0 in
my printout, just an artifact of my c code. Note that the baseline
of the adc converter is 01000 (see Burr Brown data sheet, Table VII
for more details of data format). Hex 01000 is decimal 4096.

next comes
28, the (hex) 28th data coming in

next comes
4 data coming in from fiber optic channel 4

--------------------------------------------------------
TAG BITS
```

3 bits with information on which ddc112, the ddc112 input pin, and which part of the conv (conversion) cycle (ping vs. pong). See LSB address table and cable pin out drawings.

PARITY. A modulo two sum of bits, useful to detect bit errors. Not yet fully de-bugged.

```
For example parity19:16
count number of 1's in the bit pattern.
If even, parity bit = 0
IF odd,  parity bit = 1

example1, (hex) data is
88888, that is
 8      8    8    8     8       hex
1000  1000  1000 1000 1000      binary
19:16 15:12 11:8  7:4  3:0      position
 1      1    1    1     1       number of 1's
 1      1    1    1     1       parity

example2, (hex) data is
cccc, that is
 c      c    c    c     c       hex
1100  1100  1100 1100 1100      binary
19:16 15:12 11:8  7:4  3:0      position
 2      2    2    2     2       number of 1's
 0      0    0    0     0       parity

example3, (hex) data is
5710f, that is
 5      7    1    0     f       hex
0101  0111  0001 0000 1111      binary
19:16 15:12 11:8  7:4  3:0      position
 2      3    1    0     4       number of 1's
 0      1    1    0     0       parity
```

==

E. HARDWARE SYSTEM.
E.1 block diagram.

```
SBS     <-->    DC1  <-->     PASSTH <-->    FIBEROPTIC <-->    ADC
module
Bus
Master  vme          front           VSB                <-->    Encoder
       backplane     grey            backplane
Communication
                     cable                                      Module

                     <-->    Chrislin
                    VSB      Memory VME
                   backplane Card
```

Set up commands come from the user's computer via the SBS VME master, and are passed in a standard manner, over the VME backplane to the commercially supplied DC1 card (from Access Dynamics 505 345 7637). This is accomplished by a VME A16/A24 D16 read/write to the DC1's "mailbox" address, hex b002. The scripts listed above are loaded, one word (16 bits) after another, into the mailbox address. When the user wishes the DC1 to take action based on this script, the user simply writes any word into address b004, which triggers an interrupt on the DC1, and forces it to execute the script.

The DC1 interprets what it receives. This is covered in some detail in the manual "Access Dynamics Inc. Electrometer Interface and Firmware". In essence there are 3 types of commands, which are decoded from the scripts:

- o DC1 Static Setup Commands (we called these Task 1 commands above),
- o DC1 Dynamic commands (not covered yet in this manual, future...), and
- o DC1 remote commands (we called these Task 2 commands above).

The DC1 eventually must pass data to the Chrislin memory. Hence this memory must be setup to have a proper VME/VSB address which matches that in the DC1 script.

E.2 MASTER ADC MODULE.

The ADC modules are set up by Task 2-type commands to the DC1, which in turn passes a serial bit stream to PASSTH and on to FIBEROPTIC card, and on to the (multiple) ADC modules possibly connected in parallel with the encoder communciation module. At the present time all Task 2 remote commands are addressed to the adc module. Thus we must discuss how the encoder module is commanded/queried.

One ADC module is the "master" module. When data is returned from the master module to the VME fiber optic module, a request is sent (by PASSTH) out to all modules requesting Encoder Communciation Module data. No master module, NO requests to the Encoder. There need not be an Encoder Communications module in the system. In reality every other data returned from the master generates an encoder request. Hence the returned data will have patterns such as a-a-d, where the a-'s are returned data from the master. THE MODULE CONNECTED TO FIBER CHANNEL 4 IS THE MASTER MODULE.
An infinite loop would be created if the Encoder Communciation Module were connected here!

E.3 the LED's.

LED's on the DC1 card
Underneath the label "serial port" are 4 small green LED's

DC1 LED1 :	DC1 wants data
DC1 LED2 :	DC1 is getting and processing data

```
        DC1 LED3 :
        DC1 LED4 :              DC1 sending out serial data (remote
command
                                being executed.
```

The DC1 requires the amount of data specified in the block size (times the number of blocks if this is greater than 1). As the DC1 receives 32 bit words, it decrements the quantity. When this reaches 0, the DC1 is
satified, and turns off LED1 and LED2. If LED2 goes off to early, LED1 will remain ON, indicating the DC1 requres more data. The DC1 however will flush whatever data it has received so far on to the Chrislin memory.

LED's on PASSTH VME card.

```
PASSTH LED1     indicates that the crystal clock is running.
                This LED should come on upon power up.
                It is nearest the VME card horizontally.
PASSTH LED2     indicates that the PASSTH card is receiving
                data from the FIBER OPTIC card.  Data is coming into
                the system from SOMEWHERE, either an adc module,
                or the encoder communciations module.
                It is mounted to the right of the first LED.
```

LED's on the FIBEROPTIC vme card.
Front LED, next to PCB: Good to Go
Front LED, away from PCB: Receiving A's data
Back LED, next to PCB: Handshake of data passing from
Fiber2.pcb to passth1.pcb
Back LED, away from PCB: Fiber2 putting out data to adcmod's etc

LED's on the ADC module or Encoder Communcication Module.
4 Red LED's. ???? I may have the order wrong...

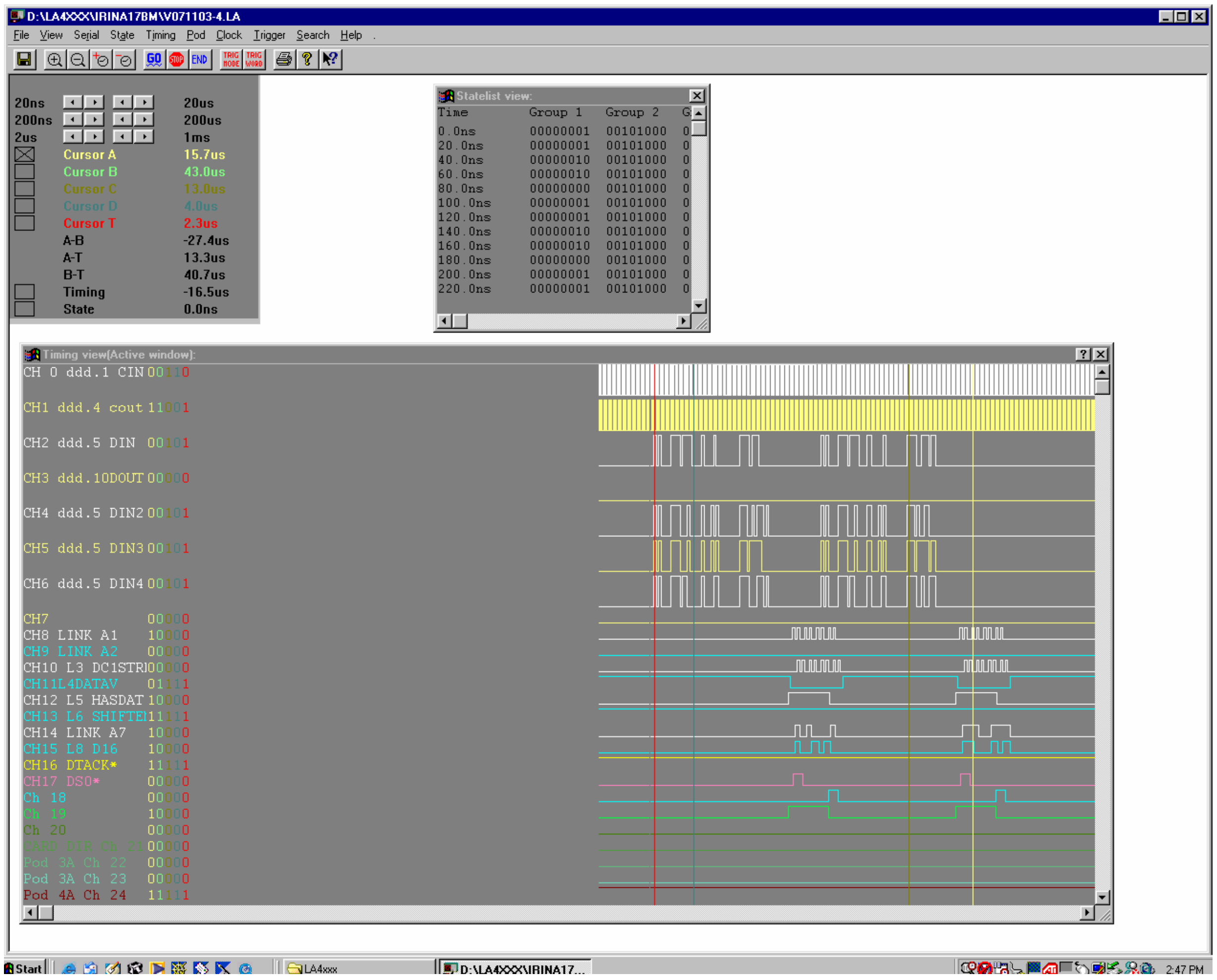
```
LED ADC3        LED ADC1
LED ADC4        LED ADC2
```

```
LED ADC1 =  ON when there is a bit stream coming into the module
            It is a stated fact about fiber optic communications that
this
bit stream
            may or may not be carrying any information.  However it must
BE
THERE,or there is no chance of information flow.
LED ADC2 =  ON when there is a bit stream exiting the module.
            It is a stated fact about fiber optic communications that
this
bit stream
            may or may not be carrying any information.  However it must
BE
THERE,
            or there is no chance of information flow.
LED ADC3 =  ON when there is information (1's actually) coming into the
            ADC module from the fiber optic module.
LED ADC4 =  ON when the ADC module Altera integrated circuit is
```

```
            delivering a clock to its fiber optic transmitter integrated
            circuit.

Thus upon power up, LED's ADC1,2,4 should be ON.  This assumes
that the fiber cables are connected, and it assumes that the fiber
optic module is working back at the vme rack.

DIP switches on fiber2.pcb
switch 1 up indicates that this fiber2 board is the master fiber2 board.
Data coming in on channel 0 of the master fiber2 pcb will result in the
passth1 board sending out commands to read the encoder.

switch 2 up indicates that fiber2 is putting out test data, a pattern
of a000 0000 0000's to the modules.  This is useful for debugging
but should be turned off (switch down) during normal operation

switch 5  up to enable data from channel 4
switch 6  up to enable data from channel 3
switch 7  up to enable data from channel 2
switch 8  up to enable data from channel 1
note that channel 0 is always enabled.

DIP switches on ADCMOD
sw 1  always leave up (on)
sw 2  when "off" puts out test pattern of a000 0000 0000
sw 3  lsb of tag
sw 4  tag
sw 5  tag
sw 6  msb of tag

e.g.
up up up up dn dn

would be normal operation of a module, with the tag or module
address being "c".  We would get data back looking like "a1ec xxxx xxxx"
etc.



==================================================================
DC
This is from the DC1 script I used for many tests dc1_sv2.dat.

48 numberlines
b0020100 -----setupacq
b0020000 commandmode
b0020000 val
b0020000 val
b0020000 val
b0020000 resv
B0020001 heartbeat
B0020000 timeout
B0028000 blksize32c000
B0020001 numblks12
B0022000 VSBaddr2000
b0020020 VSBaddr0020
b0020000 blkincr
```

```
b0020100 blkincr
b0022000 VSBstatus
b0020000 status--
b0040000 --GO--
b0020200 ----remotecommand
b0020003 commandmode
b0020000 zero
b002001a size+4
b0020000 1leading_zero
b002a000 2range
b0020001 3range
b0020000 4range
b0020000 5break
b002a000 6pulse
b0020006 7pulse
b0021000 8pulsefff0
b0020000 9break
b002a000 10period
b0020007 11period
b002ffff 12period
b0020000 13break
b002a000 14conv
b0020005 15conv
b0020280 16conv
b0020000 17break
b002a000 18flush
b0020000 19flush
b0020000 20flush
b0020000 21trailzero
b0020000 22trailzero
b0020300 -----GiveData
b0020000 val
b0020000 val
b0020000 val--
B0040000 -----GO-GO
```

Logic analyzer trace (V071103-4.LA) 4 channels working.


D:\LA4XXX\IRINA17BM\V071103-4.LA
File View Serial State Timing Pod Clock Trigger Search Help
GO STOP END TRIG MODE TRIG WORD
Statelist view:
Timing view(Active window):
Start
LA4xxx
D:\LA4XXX\IRINA17...
2:47 PM

```
20ns                     20us
200ns                    200us
2us                      1ms
[x]  Cursor A            15.7us
[ ]  Cursor B            43.0us
[ ]  Cursor C            13.0us
[ ]  Cursor D            4.0us
[ ]  Cursor T            2.3us
     A-B                 -27.4us
     A-T                 13.3us
     B-T                 40.7us
[ ]  Timing              -16.5us
[ ]  State               0.0ns
```

```
Time        Group 1     Group 2     G
0.0ns       00000001    00101000    0
20.0ns      00000001    00101000    0
40.0ns      00000010    00101000    0
60.0ns      00000010    00101000    0
80.0ns      00000000    00101000    0
100.0ns     00000001    00101000    0
120.0ns     00000001    00101000    0
140.0ns     00000010    00101000    0
160.0ns     00000010    00101000    0
180.0ns     00000000    00101000    0
200.0ns     00000001    00101000    0
220.0ns     00000001    00101000    0
```

```
CH 0 ddd.1 CIN 00110
CH1 ddd.4 cout 11001
CH2 ddd.5 DIN  00101
CH3 ddd.10DOUT 00000
CH4 ddd.5 DIN2 00101
CH5 ddd.5 DIN3 00101
CH6 ddd.5 DIN4 00101
CH7            00000
CH8 LINK A1    10000
CH9 LINK A2    00000
CH10 L3 DC1STR 00000
CH11L4DATAV    01111
CH12 L5 HASDAT 10000
CH13 L6 SHIFTE 11111
CH14 LINK A7   10000
CH15 L8 D16    10000
CH16 DTACK*    11111
CH17 DS0*      00000
Ch 18          00000
Ch 19          10000
Ch 20          00000
CARD DIR Ch 21 00000
Pod 3A Ch 22   00000
Pod 3A Ch 23   00000
Pod 4A Ch 24   11111
```

PCB modifications – on-going, see also notes on schematics.

ADCMOD3 fixes: remove 3 resistors R49, R50, R51, since 2nd altera chip makes these connections.

PASSTH6 pcb fixes:
JCROSS 3, 1 cut trace between the VME and VSB side.
JCROSS3 vme side goes to JCROSS1 vsb side

JCROSS1 vme side goes to JCROSS3 vsb side

Toggle switch cut right hand leg, attach left leg to C27 (top) for ground

RPB2 should be 0 ohms, a wire
RNEW1,2,3,8 should be open circuits.
Cut off UBB2, UPB1, forget these stupid power reset chips
Clock bypass R106 = short circuit
R105 = open circuit
RINT1 = short circuit (on bottom of PCB)
RPB4 = open circuit
U30 pin ?? connected to RPB4 on bottom

Extra notes: