From: Steve Ross
Skross@anl.gov
Phone 630 252-9510
fax 630 252 9350
401-c1252
APS/ASD/Electrical Systems Group

ABSTRACT:

Description of electrometer amplifiers applied to PIN diode based x-ray beam position and intensity monitors. This document is work-in-progress, suggestions welcome.

APS/ASD/ESG has developed an electrometer amplifier sensitive in the range of picoamps to microamps. This low cost amplifier has a high (10^6) dynamic range, and also a bandwidth of up to 1600 samples/sec. Its noise floor is in the range of 100 pA RMS (1600 Hz). In one 9" x 5" package, called an ADCMOD2 module, there are 4 such electrometers. The main application at the APS is to monitor the output of 4 detectors, commonly PIN diode detectors, for x-ray beam position monitoring. 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 believe that a basic system would read 16 PIN diode current sources, but there are several variations. We work with the beamline personnel to get a system up and running.

PIN diode Users Guide 12-17-02.doc
Rev date: 12/14/2002

CONTENTS: This memo describes electronics developed to read 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. (X-BPM).

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, based on design and on data taken at sector 32.
(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. 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) 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.
(7) We give information useful to running the system. This section is continually being edited, to be clearer, and as we gain more experience with users running registered mode.
(8) Schematic information. This section will grow into a description of where all the design files are documented. It will mainly be of use to ESG personnel.
(9) Future features. This discussion will be never ending.
(10) APPENDIX. For now this is where I add information mainly relevant to the digital storage scope mode, and sector 32 operation. It will however help me to go over this mode with beamline users who may want to move over to it.

(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. Their electronics collects 12 PIN diode signals, plus the signal from the encoder on the x-ray monochromater. As the energy is swept, plotting one versus the other gives an EXAFs curve. Thus this system works in "digital storage scope mode", as contrasted with "register mode". (These modes of operation will be further discussed below.)

(2) PIN DIODE DETECTOR.

The detector arrangement was based on designs at sector 19 [REF: R.W.Alkire, G. Rossenbaum, 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. I believe that only Randy at sector 19 has actually moved the quad PIN detector in a calibrated way, and this was done without using this set of electronics. But based on his

work, and the specifications of this system, it is likely that the position of the beam can be estimated with sub-micron precision.

The red wire of the PIN diode connects to the signal input of the amplifiers; the black wire connects to the ground. The PIN diodes have a common (analog) ground at the printed circuit board. Four pairs of wires go from the amplifiers to the four PIN's.

Randy works with companies which supply the foil xxxxname?. 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. These were done for the convenience of other APS CAT's. Some beamlines have unique mechanical constraints**. The point is that 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,E]

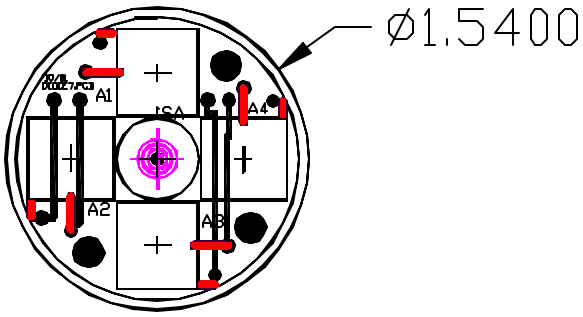
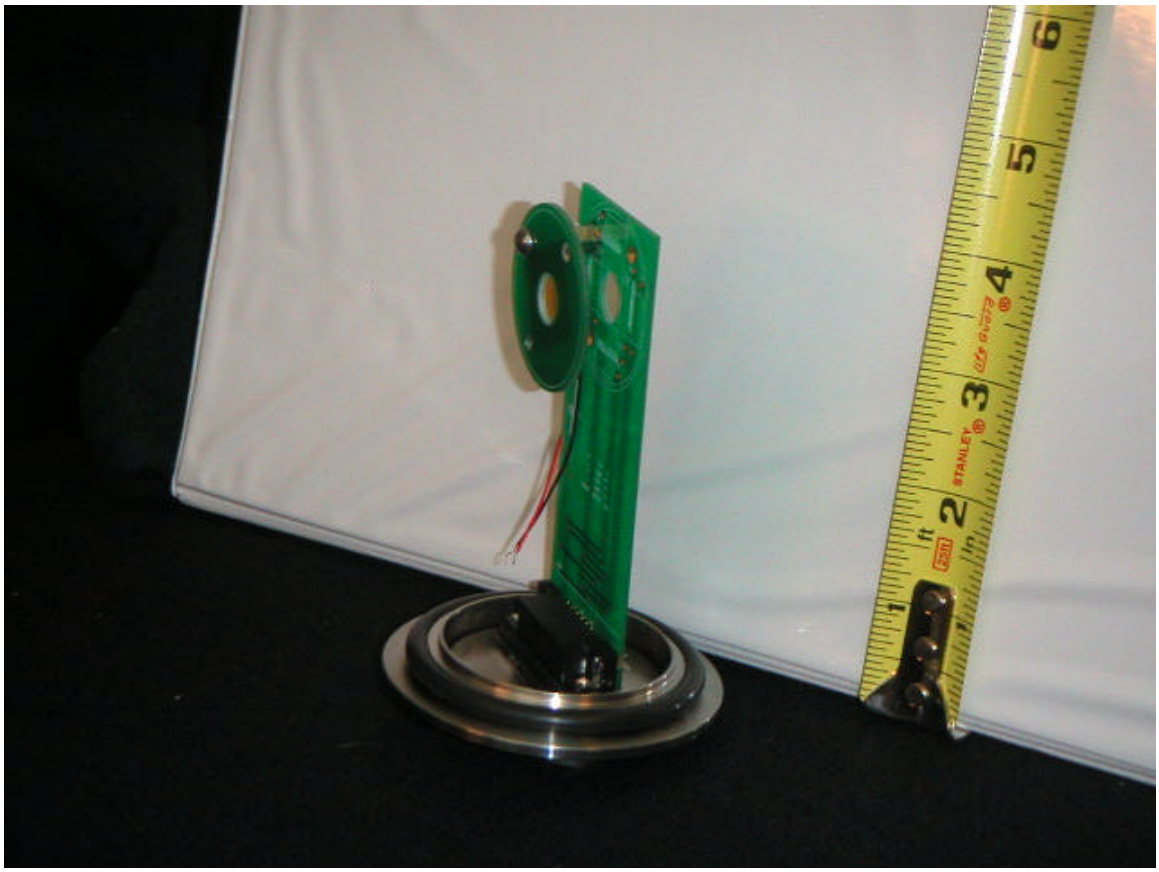


FIG 1A This is the basic printed circuit board type design (DIODE5) holding the four UDT 1 cm2 PIN diodes. It is about a small as possible with these diodes. The center hole in milled out to about 0.5 inch diameter. This circuit board fits into a KF-40 or KF-50 size vacuum system.



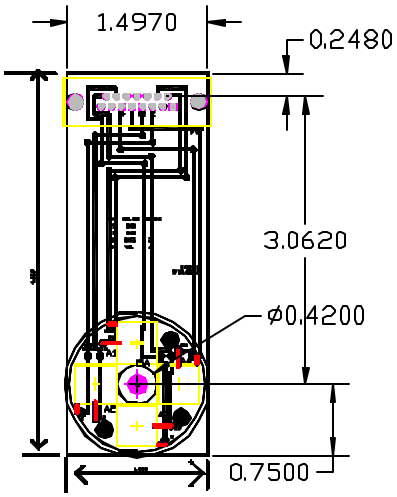


FIG 1 B,C A photograph, and a drawing of the diode6 design. This printed circuit board fits into a KF50 cross and positions the diodes in the center. Standoffs hold a second circuit board approximately 1 cm. above it and this circuit board holds the foil. This is a very simple one-piece design for medium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

.

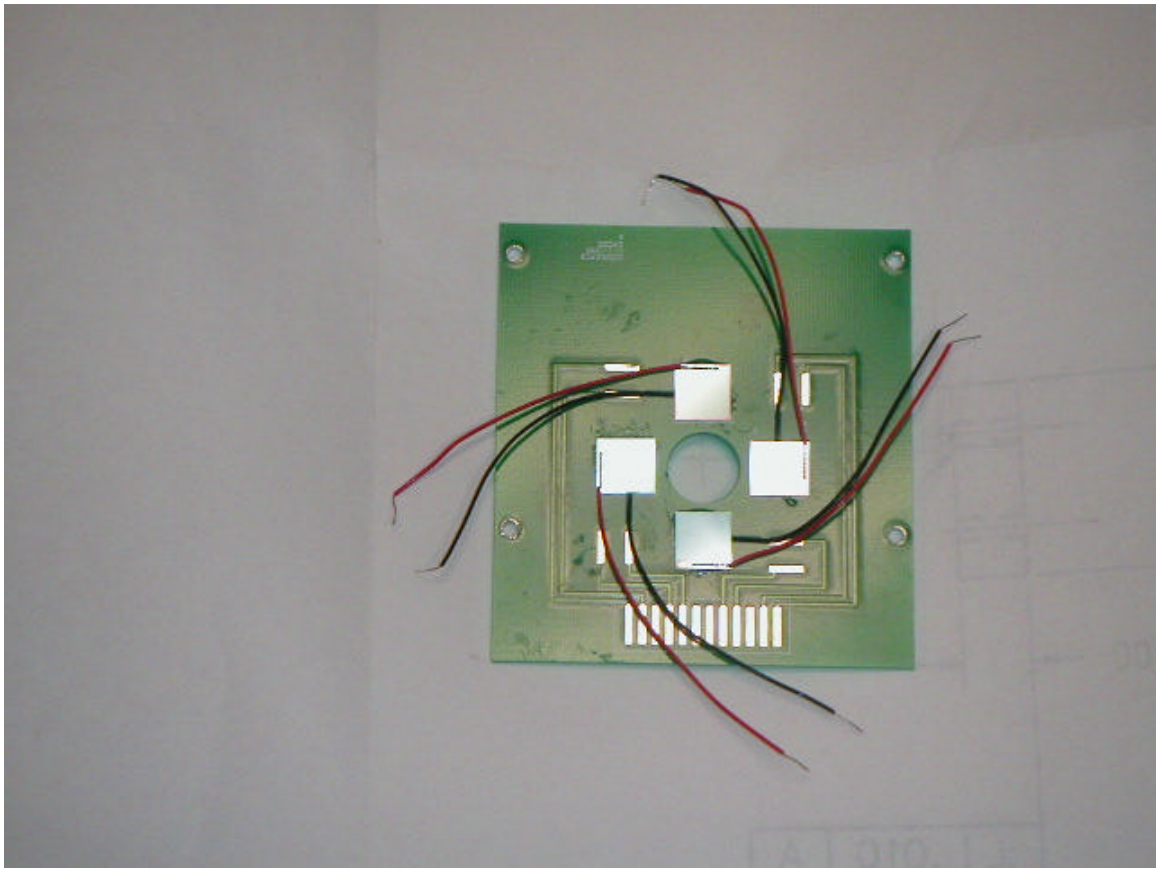


FIG 1D Photograph of DIODE1 layout. Again this is just another variation of how the diodes can be arranged. Red wires signal, black wire are grounded together on the PCB.

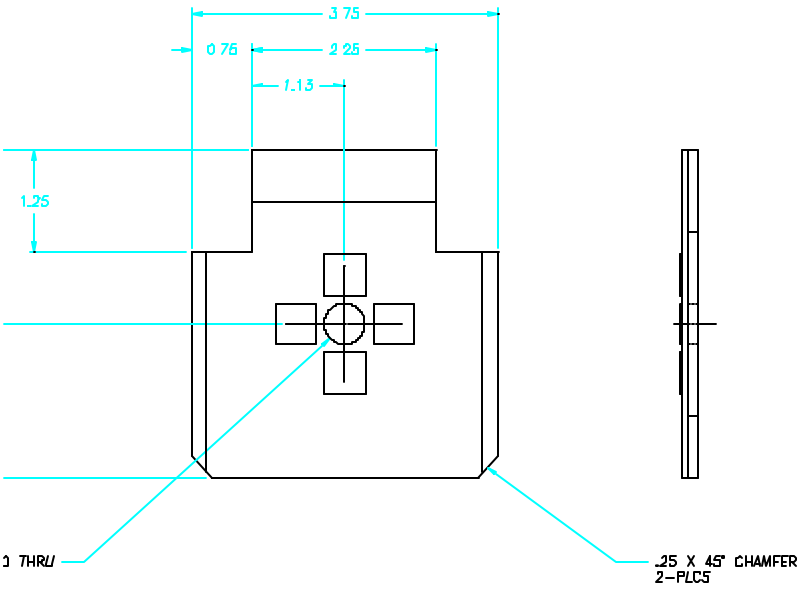


FIG 1E this is the design for the diode detectors at sector 32. (DIODE1) The diodes are on a circuit card, which can slide into a cardholder type mount. At this sector, the metal foils can be changed remotely, so that different foils can be inserted for different EXAFs scans

(3) SPECIFICATION FOR THE ELECTRONICS.

We have two electrometer modules based on an integrated charge amplifier and based on a voltage amplifier. Almost all the discussion here relates to the charge amplifier. It is the version currently going out onto the APS storage ring beamlines. We anticipate the use of the voltage amplifier when the signal currents exceed about 5 uA. Grounding for optimum noise performance is an open issue to be addresses at each beamline.

CHARGE AMPLIFIER. 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 printed circuit board is given. (FIG 2) All the modes and options discussed in the data sheet are implemented in our system. However because ultimately the integrated circuit uses a 20 bit analog-to-digital converter (ADC) we find that most options, most "bells and whistles" are not needed. Twenty bits is a dynamic range of a million. When run in a somewhat standard manner (using the external 220 pf capacitor for maximum rated full well), we observed the following in digital storage scope mode:

•4 Channels – currents from 4 unbiased PIN diodes processed in each module
•Maximum readout rate 1600 samples/sec per diode [1]
•Noise of electronics alone <25 pA RMS (has not yet been optimized)

•Noise of system, including UDT S100-VL 1 cm^2 diode <100 pA RMS (battery), 200 pA RMS (lab supply)
•Dynamic Range: 20 bit ADC, LSB = 1 pA, full = 5 uA [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
• Software controllable (see below)

Notes:
[1] See discussion of conv command.
[2] Will change with signal averaging.
[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 ADCMOD2 module requires about .38 amps. The supply voltage should be in the range of 9-14 volts, I typically use +10 volts lab supply. This supply is "cleaned up" with a voltage regulator chip. The units can be powered by a +12 VDC battery, which can last for about 12 hours. (See also future-features section)

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). Gain range control is performed by varying the feedback capacitor. (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 twoddc112 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. We mention that we have not yet pushed the limits of gain control. Eventually we will place on the circuit board feedback capacitors larger than those specified in the data sheet, with the goal of being able to measure higher current levels with acceptable linearity.

As we discussed this system with other APS users, we realized that most users actually want continuous readout of the XBPM. However about 1600 samples per second per PIN typically is too high for many beamline data systems. Data cannot be read into EPICS at this rate. We mitigate this with digital signal averaging in the PASSTH4 card.

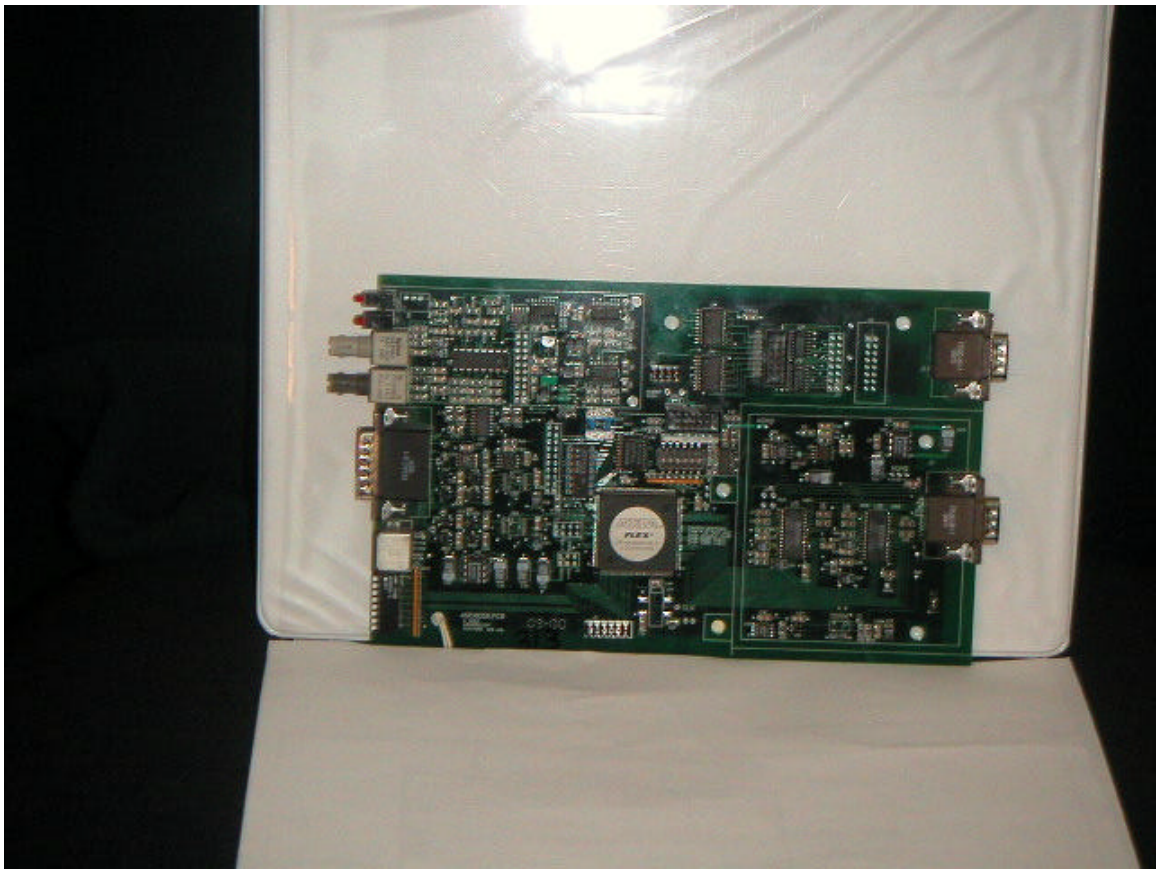


FIG 2. Charge amplifier four channel electrometer. It is placed into a box measuring 9" x 5.5" x 2.5".

VOLTAGE AMPLIFIER. Another electrometer printed circuit board has been fabricated and partially 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. The system can work at lower currents, but when we cross back into nanoamps, the charge amplifier works very well.

Basically this amplifier consists of an electrometer grade operational amplifier (Burr Brown OPA129U). 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). (FIG 3)

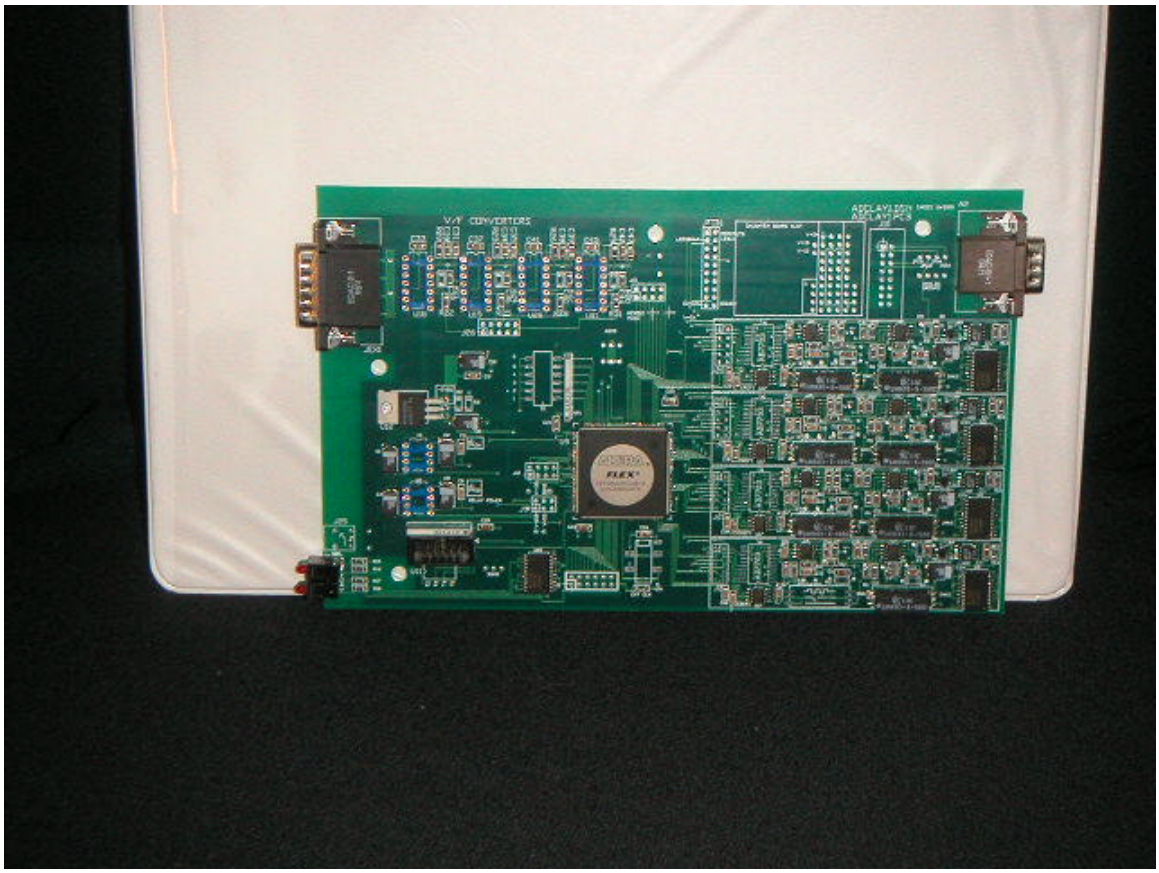


FIG 3. A photograph of the voltage amplifier method for electrometers. This board holds four chains of electrometer voltage amplifiers. Data is digitized, and can be packaged in a way consistent with the charge amplifiers. There are V/F converters on this board but this is not a desired development path, ADC's 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. We also discuss here two system readout modes, with the emphasis on the registered mode. 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 ADCMOD2 card. 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. See Section 2 of this memo.
(2) Qty 4: Four channel electrometer electronics boxes, "ADCMOD2". These will reside in the hutch, near the detectors. They are the amplifiers. (FIG 2,3)
(3) Qty 4: terminated fiber optic cable pairs run from the hutch, to the control rack. Each ADCMOD2 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" to receive the fiber optics (FIG4) One of these cards can support four (4) ADCMOD2's.
(5) Qty 1: an APS supplied VME card "PASSTH4", which can be read by a VME host, such as an IOC running EPICS. (FIG 5)

As discussed in section 3, the ADCMOD2 amplifiers read in the PIN diode signals, convert to digital levels, and output via a fiber.. Each FIBER2 card can support 4 ADCMOD2 cards, which in turn read 16 PIN diodes (e.g. four quad arrays). Each FIBER2 card needs the PASSTH4 card to format the data to be read over the VME backplane.

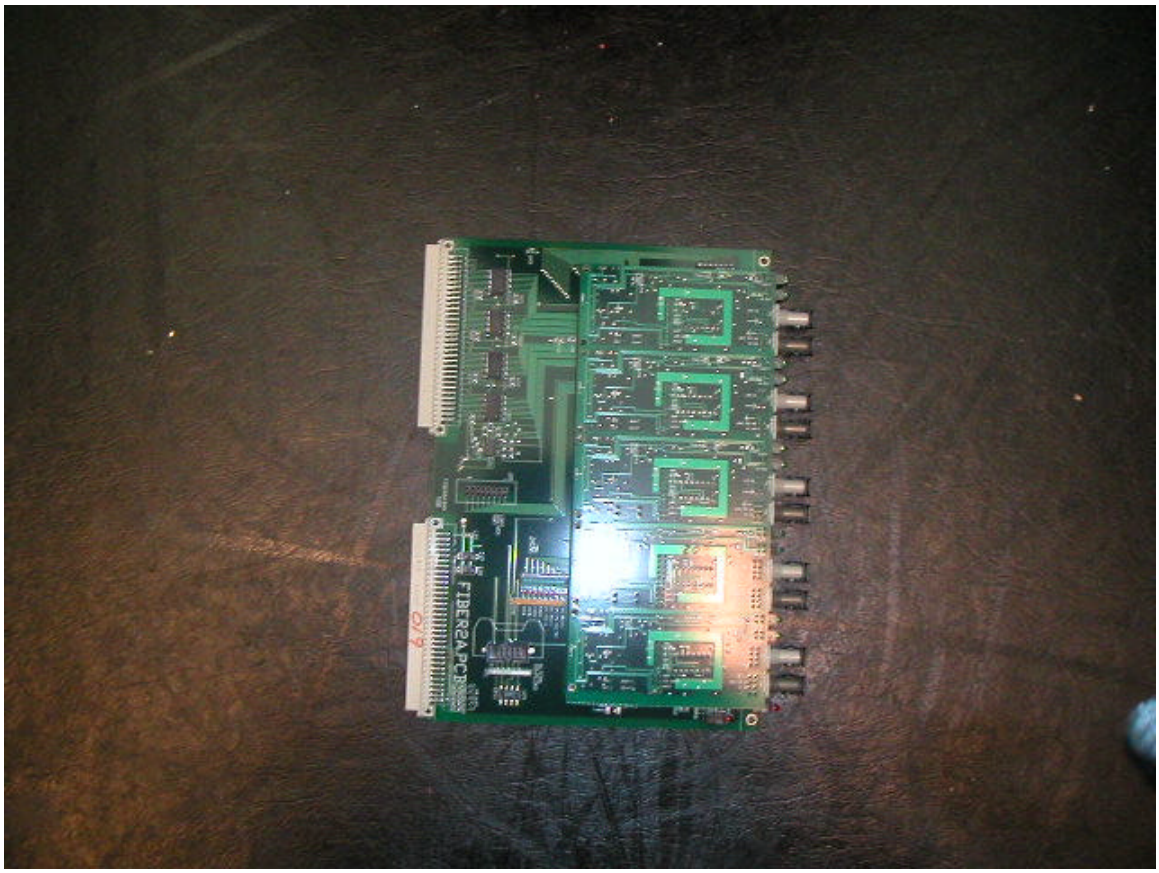


Fig 4 FIBER2 VME board

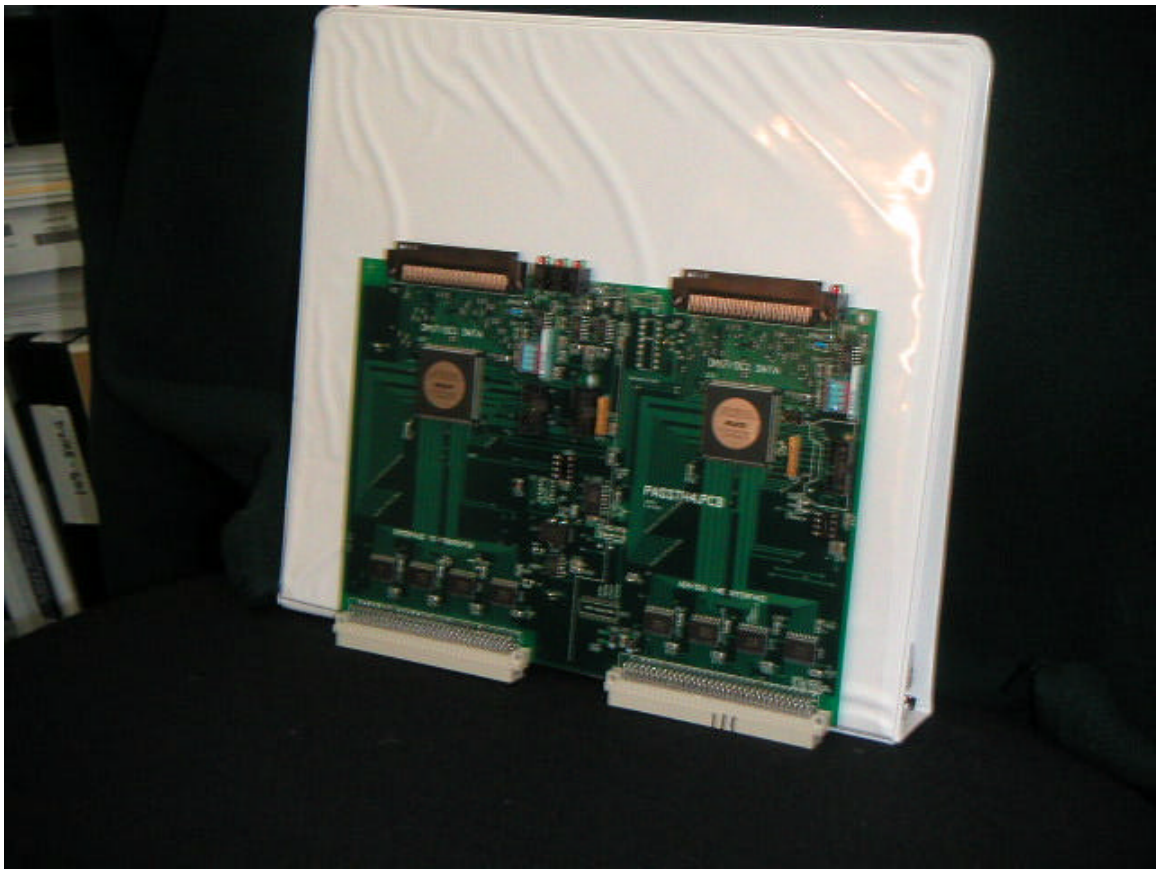


Fig 5. PASSTH4 VME board.

4.2 REGISTER MODE vs. DIGITAL STORAGE SCOPE MODE. Data can either continuously come out of the ADCMOD2 cards (registered mode), or can be output for a time, and then stop when a large memory is filled (digital storage scope mode). The emphasis in this memo is on the "register mode" of operation.

Registered 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 ADCMOD2's, travels over the fiber back to the control VME rack. It is moved into a static RAM memory in the PASSTH4 card. This memory 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.

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 16 bit PIN values. This would typically be a parallel input/output VME operation (PIO).

This is how the system stands now (12/2002). Two problems have now become obvious to me: (1) few beamlines can read the data OUT of the VME addresses as fast as the

electrometers can put the data IN. (The values in the registers would likely change about every 800 us.) Thus I have added signal averaging. PASSTH4 has the functionality that it sums up the signals from a given electrometer (PIN diode), and then updates the VME register at a slower rate. This averaging also improves the noise performance (at a trade off of bandwidth). The exact number of samples that are summed is TBD, not to exceeed 4096. This parameter will become more "user friendly" to change as I improve the firmware. (2) The data samples, or averaged data samples are not co-registered in time. If the VME system is not fast enough, it may read the PIN signal from diode 1 and 2 (for example) at time t1, but not get back to reading the signal from diode 3 and 4 until later. This can be fixed without much trouble but it is not done yet. I am learning user requirements as I go.

Digital Storage Scope Mode. The original sector 32 electronics was meant for EXAFS, and as such recorded simultaneously monochromater encoder position signals. A user-specified number of samples were recorded into a large (64MB) VME memory. (At a typical time between samples of 800 us, 1300 HZ, perhaps 50,000 samples), When the specified number of samples is collected, system stops acquiring data. The user is then expected to read all this data from the VME memory into the host computer memory.

To implement this mode of operation, we need to add two more VME cards to the system, for a total of four cards. The data from the PASSTH4 VME card discussed in section 4 is passed to a similar card, called the DC2. . The purpose of the DC2 card is to handle all memory access. From the DC2, data is pumped into a commercial VSB/VME dual ported D-RAM memory. Data flows into this memory over the VSB bus. This flow does NOT interfere with VME commands on the VME backplane. Once it is set up, and told to acquire, it will handle the movement of all data to the Chrislin memory. The Chrislin memory can be read out whenever the users wishes, as it is dual ported. The intent is that the user wait until the "storage scope" type scans are complete before commencing this long transfer out.

In reality, the DC2 card appears identical to the PASSTH4 card from which it receives data, but the firmware inside the Altera chips is different. The Chrislin, Inc.VME40 memory typically costs less that $700 for 64 MB cards. See www.chrislin.com. This company has provided such products for >12 years.

(5) SOFTWARE REQUIREMENTS. I run an entire stand-alone software system based on a PC, Microsoft Windows, and MSVC++ 6.0 code. I don't expect this to be particularly popular around the APS. I can bring it out for set up however. This is how do acceptance testing.

I understand that I need to add more discussion to this section. There are several details which are useful as time goes on, but not to "get up and running". 12/2002.

I am working with Mark Rivers of BIOCARS/14. He and I discussed how to get the data into his EPICS system. Thus we hope you can just obtain his EPICS drivers. Mark is

using EPICS to control the feedback of a piezoelectric crystal in the monochromater, for beam stability. Talk to him about his EPICS PID controller record.

For the past 3 years, Steve Wasserman does all this through SPEC.

We now discuss details of software setup, emphasis on the register mode.

5.1 SOFTWARE REQUIREMENTS FOR REGISTER MODE.

5.1.1 DATA OUT TO THE ADCMOD2 MODULES FOR SETUP. The user must supply initializing information to the ADCMOD2 electrometer. For register based readout, 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. (For dc storage scope mode, this setup data must also indicate how many samples are required before terminating scan.)

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. All ADCMOD2 modules are set up identically (assuming they are cabled in, and powered up.)

Internal to the system, over the fiber, between FIBER2 and ADCMOD2, data is passed as serial bit streams 48 bits long. A command to the ADCMOD2 is 48 serial bits long. **For the VME host to command the ADCMOD2 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 ADCMOD2's. However sector 32 has additional devices.

The following is a basic script. **Only 5 commands are really needed: range, pulse, period, conv, go. There is no real order, but typically the go-command comes last.** As time goes on I will try to cover some of the "bells and whistle" commands, they are often for de-bugging and test.

Read the following script as AAAADDDD #comment.

For example **f018a000 2range** from the table below means to my system:

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 address 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.

Commands below with comments 2range, 3range, 4range combine to form the 48 bit stream internally. The 5send command is simply are VME write to address 0x00f020 and triggers the internal serial stream.

Thus in the example the range command is (hex)
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. Yours may vary slightly. It is nice to have the comments out to the side.

```
40 testdc2
f018a000 2range
f0100001 3range
f0080000 4range
f020cccc 5send
f018a000 6pulse
f0100006 7pulse
f008100f 8pulse
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
```

5.1.2 DATA COMING BACK FROM THE MODULES:

Internal to the system, and transparent to the user, data returning from and ADCMOD2 is again packaged into 48 bits. These bits contain diagnostic, address and parity information and all the data bits from the ddc112. There are dip-switches on each ADCMOD2 printed circuit board. The switch pattern is also embedded in these 48 bits.

Under normal operation only the ddc112 data is presented into the VME accessible registers. There are diagnostic modes. (need to discuss someday…)

We now discuss this VME returned data.

The DDC112 output values go from0x00000 to 0xfffff. (see its data sheet) This entire 20-bit value is given to PASSTH4. But since the VME data word is only 16 bits, we must somewhere loose 4 bits. If there is no signal averaging we simply drop the 4 least significant bits. I note that the bottom 4 bits (16 counts) are noise anyway at this point (16 counts correspond to about 16 pA, the present noise level is above this.) **Hence the VME registers see the most significant 16 bits from the 20 bit amplifiers.**

At the very beginning I'll keep signal averaging off. When it comes on, the following will apply: There must be a match between which bits are given to the user and the number of signals averaged, otherwise there will be a lot of wasted bits. For example if 8 values are added, then in the sum the least significant 3 bits will have little meaning. I need to work on this but the system will eventually give the most significant bits, and just exactly which bit is MSB will depend on the averaging. Averaging is done in a 30-bit adder so we can add about 2^{14} counts, i.e. 4096 counts.

The DDC112 has an internal DC offset. When no current is going into the module, the output value is 0x01000. 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.

To convert digital counts to actual current, you must know the gains. Typically I do this the easy way and inject a known current from a Keithley source. Again 1pA is about one analog-digital unit (ADU) when range = 000.

5.2 SOFTWARE REQUIREMENTS FOR STORAGE SCOPE MODE.

Discussed in the appendix for now.

5.3 DEBUGGING MODES – HOW TO READ DIAGNOSTIC BITS INTO THE VME HOST COMPUTER.

I'll expand on this as time goes by. But the idea is that we can make changes to the system to quickly access its health. A typical thing to do would be to replace electrometer data with bits showing which ADCMOD2 is which, that the parity is ok, pick out the dip-switch information etc**.**

(6) INSTALLING A SYSTEM AT AN APS BEAMLINE.

6.1 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 $1000 per VME board, and about $500 each for an ADCMOD2 board. Thus the incremental cost of adding four more electrometers is $500. This price is expected to drop more as quantities rise.

6.2 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 are currently installing a system on about 11 beamlines, so there is some comfort in numbers.

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

6.3 INSTALLATION ISSUES.

FIBER OPTICS.
Each adc module communicates over a pair of fibers. The fiber opticcable comes in fibers to a cable "4-fiber cable". This cable isbreakout fiber, orange color. I supply it.

(1) How long is your length of fiber?

(2) How many 4-fiber cables do you wish pulled now?

(3) will your technicians pull it, or do you want ASD techs to help (at $70 an hour I think)?

(4) Do you wish ASD to terminate your fibers with the required ST-2 connectors?

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 (one worked out with John Quintana sector 5 which precisely fits inside a KF-50 cross. I call it the diode6 design.)

Also you can notice a sketch of a round pcb 1.54" diameter. I call it the diode5 design.

I have another fairly complicated design which slides into a KF50 cross like a cylinder in a pipe. Call it diode4.

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

(5) Do you wish to work with me to hold the PIN diodes, if so how --standard product or custom?

FOIL HOLDER.
Using diode5 or diode6 design, some sectors just want to have another pcb mounted to the irst, and onto it glue down the foil. These "blanks" have no diodes on them, but just simply hold the foil. (FIG DIODE7.dwg) They can be mounted on stand-offsontodiode5 or diode6. Typical stand off distance is 1 cm.

(6) Do you wish to work with me to hold the foils? If so how?

(7) Will you get your own foils? I can tell 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 consider "standard", I have a stock of connector. Don't be surprised if we have to order stuff from Newark.

PIN DIODES AND LIGHT.
The pin diodes respond very well to visible light. For example if you have a vacuum gauge, a Bayart -Alpert vac-ion 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 adc modules require power. Typically this is about 10 VDC, about .4Amps per adc 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 adc modules.

I supply the connector which plugs into the adc module, to bring power into the adc module. I bring this connector out to 2 wires (hot/gnd). From there it is your problem.

Note: as 2003 progresses, and I see that the systems are working well for you all, I will likely build a ping-pong battery system. This system would be charging one small lead acid 12 vdc battery while draining another. Batteries are great for noise immunity. I cannot find a commercial device which will do this. I have a prototype of this, it is about 12" x 8" by 6" or so. One can get lead acid 12 v batteries from newark, the ones I use typically will run a module for about 12 hours.

(8) Do you have your own dc power, is this all ok?

(9) Would you be interested in a battery system, cost approx. $400, next year ?

TESTING.

I'll bring a cart out with a PC, running windows 98, and my own C++ code. I'll put the cards my VME crate, along with a LED bus monitor card. (FIG 6) I'll show you a working system. Then we have to work together to get into your software. I display the data in IDL, and use IDL to compute noise etc.

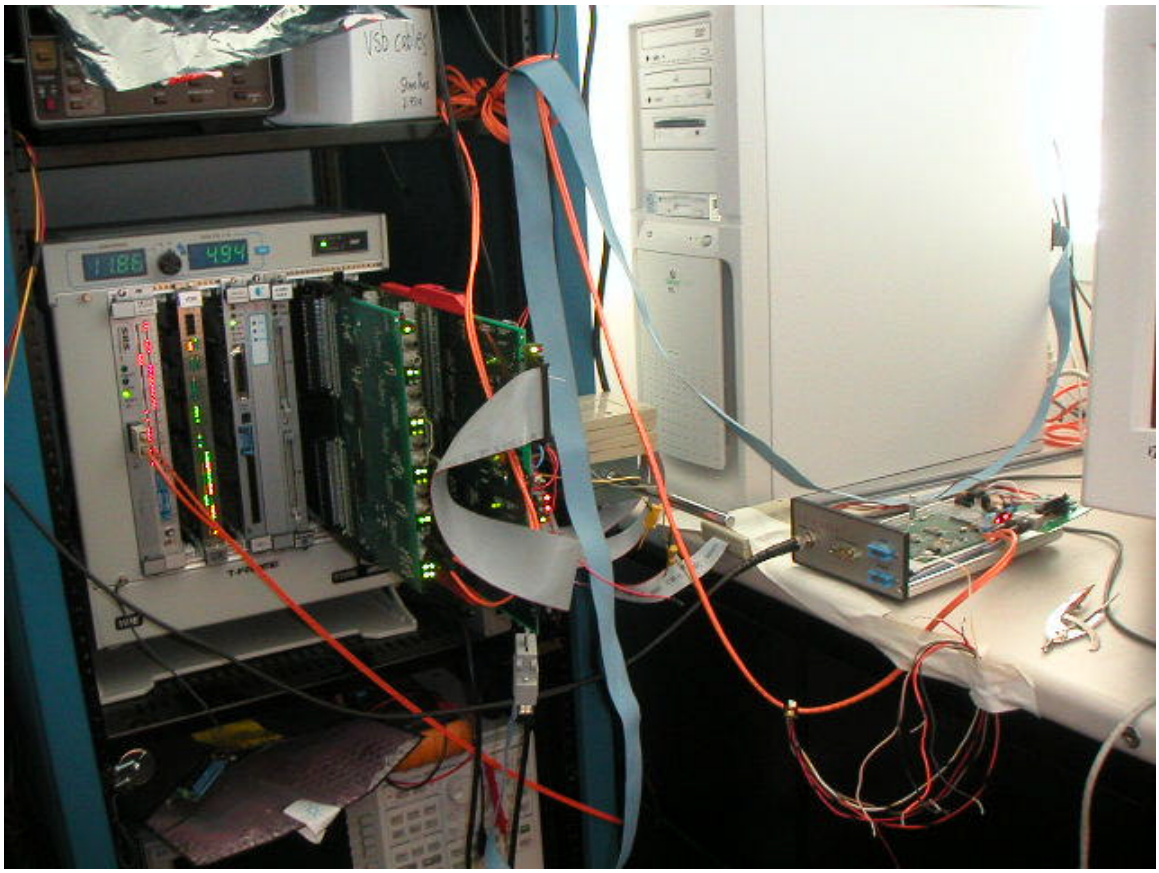


Fig 6. Typical test stand consists of a VME rack and a PC running Windows. The software for my own testing is simply MSVC++ and IDL (which are linked to pass data from one to the other.)

7.0 ISSUES RELATED TO OPERATING THE SYSTEM

Taken from instr4.txt: version 9/29/00, work in progress.
Especially need a better explanation of the LED's.

7.1 REBOOTING THE SYSTEM -- TURN ON SEQUENCE.

(1) Turn off the vme crate, the adc modules and any other modules on the fiber chain (e.g. encoder communications module) (nNte you do NOT have to turn off the Heidenhain encoder module)

(2) Turn ON the VME crate.

(3) Reset the PASSTH4 card by pushing the button on it.

(4) Turn on the encoder communcications module. 3 led's come on.

(5) Turn on the ADC modules. 3 led's come on.

(6) Push the Passth4 "reset" button

7.2 DETAILED EXPLANATION OF THE COMMANDS TO THE ADCMOD2

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.

7.2.1 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
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
```

With gain range set to 000 (command a000 0001 0000),
and with the conversion time set to 0200 (command a000 0005 0200)
and with 10 nA going into the system, the counts will read 14260 (approx).
There is a baseline of approximately 4096 counts.
So 10 nA creates a signal of 14260 - 4096 = 10164 counts in 820 us.
So 1 pA would create just about 1 count.

7.2.2 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
```

7.2.3 PERIOD

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

7.2.4 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.

8.0 DESIGN FILES.

This section will grow into a description of the design files – OrCAD, printed circuit board, Altera and AutoCAD. For now know that there is a block diagram on the ESG shared disk. We have bills of materials into excel spread sheets. The altera designs have been extensively simulated.

9.0 FUTURE FEATURES.

Much will depend on feedback from users. This is just a laundry list.

Rechargeable ping-pong battery supply (while one is draining, the other is charging, the ADCMOD2's never see line current).

Better diagnostics.

Make sure the LED's are useful to the user.

10.0 APPENDIX. DEBUGGING INFORMATION. USER DOES NOT NEED THIS.

EMBEDDED DE-BUGGING DATA. NOT DISPLAYED IN REGISTER MODE.

```
1 adc module, 1 encoder communciation 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 communciation 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 communciation 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"


================================================================
C.  EXAMPLE SCRIPTS:
```

THESE SCRIPTS ARE WRITTEN IN THE FORMAT FOR SECTOR 32. THEY ARE OF LIMITED VALUE TO OTHER SECTORS, BUT HELP ME DEBUG.

```
Note #1: In these scripts, the number e.g. 0100 on the first line, is to
be sent to vme location b002
Note #2: The English comment just afterwards is a human oriented
comment,
not to be sent out.
Note #3: The line
0000 -----GO-GO
really means send anything (0000 is ok) to vme location b004 to
execute the commands loaded into b002.

Example 1:  Goal: obtain just a little bit of data, to quickly read the
encoder etc.

0100 -----setupacq
0000 commandmode
0000 val
0000 val
0000 val
0000 resv
0001 heartbeat
0000 timeout
0100 blksize32         !very short block
0001 numblks
2000 VSBaddr2000
0020 VSBaddr0020
```

```
0000 blkincr
0100 blkincr
2000 VSBstatus
0000 status--

0200 ----remotecommand
0003 commandmode
0000 zero
001a size+4
0000 1leading_zero
a000 2range
0001 3range
0000 4range
0000 5break
a000 6pulse
0006 7pulse
0100 8pulse
0000 9break
a000 10period
0007 11period
ffff 12period
0000 13break
a000 14conv
0005 15conv
0280 16conv
0000 17break
a000 18flush
0001 19flush
0007 20flush
0000 21trailzero
0000 22trailzero

0300 -----GiveData
0000 val
0000 val
0000 val--

0000 -----GO-GO

------------------------------------------------
Example 2:   Goal:  once you have been taking data for awhile,
and you ONLY want to change the gain range of the ADC:

0200 ----remotecommand
0003 commandmode
0000 zero
0009 size+4             !note size + 4 changed.  9 lines in this command
0000 1leading_zero
a000 2range
0001 3range
0003 4range             !new range, now 011 (3 in binary)
0000 5break

0300 -----GiveData
0000 val
0000 val
0000 val--
```

```
0000 -----GO-GO         !sent to vme address b004

-----------------------------------------------

Example 3:  Goal:  once you have been taking data for awhile,
and you ONLY want to change the integration time of the ADC:

0200 ----remotecommand
0003 commandmode
0000 zero
0009 size+4
0000 1leading_zero
a000 2conv
0005 3conv
0200 4conv              !new conversion time, hex 0200, or 820 us
0000 5break

0300 -----GiveData
0000 val
0000 val
0000 val--

0000 -----GO-GO        !sent to vme address b004


-----------------------------------------------

Example 4:  Goal:  Using the adc module in non-continuous mode.

I have seen that this will allow the unit to process up to 5.8 uA of
current without saturation.Conversion input of hex 24 yields
a conversion time of1.6 * (0x24 --> decimal 36) --> 58.2 microseconds.
Period input of 300 yields a period time of 1228.6 microseconds
(per L092900.D01).  (i.e. 300 hex --> 768, 768 * 1.6 = 1228,
1228 plus 0.6 = 1228.6 microseconds).  [ref notes 09/21/2000]


48 dc1test1
b0020100 -----setupacq
b0020000 commandmode
b0020000 val
b0020000 val
b0020000 val
b0020000 resv
B0020001 heartbeat
B0020000 timeout
B0021000 blksize32c000
B0020001 numblks12
B0022000 VSBaddr2000
b0020004 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
b0020800 8pulsefff0
b0020000 9break
b002a000 10period
b0020007 11period
b0020300 12period
b0020000 13break
b002a000 14conv
b0020005 15conv
b0020024 16conv
b0020000 17break
b002a000 18flush
b0020001 19flush
b0020007 20flush
b0020000 21trailzero
b0020000 22trailzero
b0020300 -----GiveData
b0020000 val
b0020000 val
b0020000 val--
B0040000 -----GO-GO
```

==========

D. INTERPRETING THE DATA. LIMITED USE FOR PEOPLE RUNNING REGISTER MODE, MAINLY FOR DIGITAL STORAGE SCOPE MODE. HELPFUL FOR ME TO DEBUG.

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 communcications 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 WORD NUMBER	ADDR VME	CONTENTS	COMMENTS

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
```

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

-------------------------------------------------------
comment 6
```

The tags for which adc readout is which. If you put together the 4th "q" bit, with the 1st and 2nd "s" bit you get 3 bits. These tag bits come out in the following order:

tag bits:

011
111
001
101
010
110
000
100

The interpretation of these 8 permutations of 3 bits are:
DOUT is pin 16 of either the A or B adc integrated circuit u5,u8
Input 2 is pin 28 of the Burr Brown DDC112, Input 1 is pin 1.
J1 is the connector at the edge of the printed circuit card.
For explanation of conv look at Figure8 of DDC112 data sheet.

tag bits	altera pin	DDC112 pin		J1 pin	wire color		
011	DOUTA	input2 conv=hi u5.28	IN2A	J1.3	RED	PINDIODE2	ping
111	DOUTB	input2 conv=hi u8.28	IN2B	J1.7	YELLOW	PINDIODE4	ping
001	DOUTA	input1 conv=hi u5.1	IN1A	J1.1	BROWN	PINDIODE1	ping
101	DOUTB	input1 conv=hi u8.1	IN1B	J1.5	RED	PINDIODE3	ping
011	DOUTA	input2 conv=lo u5.28	IN2A	J1.3	RED	PINDIODE2	pong
111	DOUTB	input2 conv=lo u8.28	IN2B	J1.7	YELLOW	PINDIODE4	pong
001	DOUTA	input1 conv=lo u5.1	IN1A	J1.1	BROWN	PINDIODE1	pong
101	DOUTB	input1 conv=lo u8.1	IN1B	J1.5	RED	PINDIODE3	pong

So when you for example pull out the tag pattern 011 from the "q" and "s" bits, you have read the "pong" read of PINDIODE2, coming to you over a red wire.

Note that when we look at the printed circuit board containing the pindiodes, looking at the diode side of the board, we see them in the order

PINDIODE1

PINDIODE2 hole PINDIODE4

PINDIODE3

x-rays are coming out of the page.
Pindiode 1 is physically on top.

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:

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

==========

APPENDIX 1.
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
```