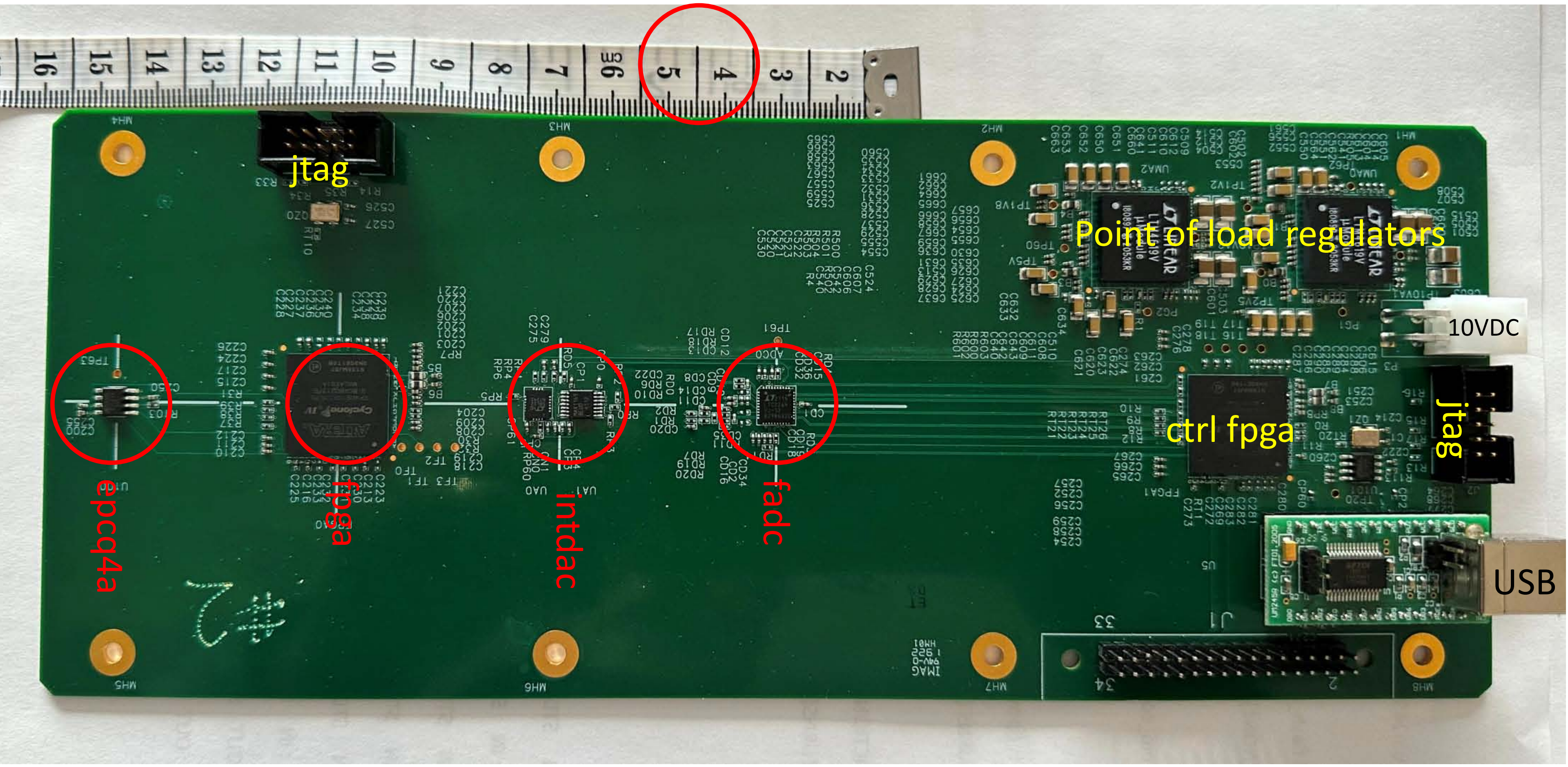
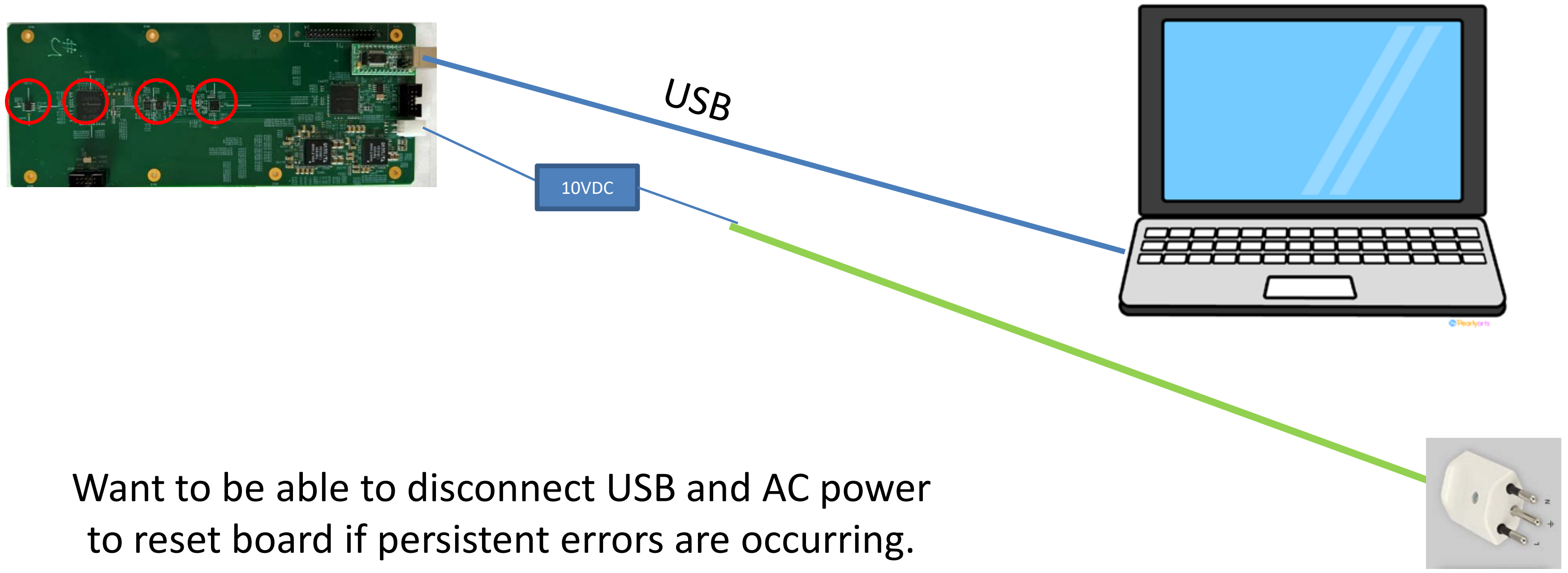
SEE Lot Test Board



Radiate 4 positions with 2cm diameter beam spot in this order:
1- fadc   2 -intdac   3 - fpga   4 - epcq4a (prom)

Long USB and AC power cables from the beam area to the monitoring location.



Want to be able to disconnect USB and AC power to reset board if persistent errors are occurring. This is not expected to happen, but some bit flips that correct in the next write cycle of the fpga monitoring are expected.

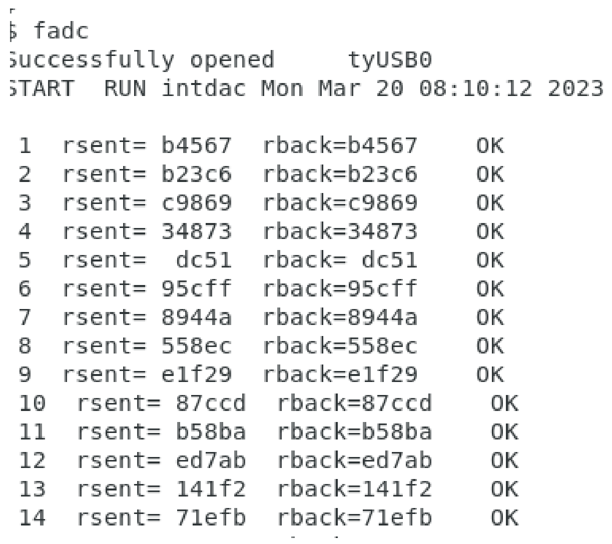
Laptop: user: kelby
pwd idah0an

root (should not be needed)
out of$

Log on as kelby
open a terminal
```
> xs
```
This will open a xterm
window (needed for grapics)
```
> cd SEELotTest
> ls /dev/ttyU*      /dev/ttyUSB0 should be there)
> ./setterm         ( set baud rate … etc)
```

There are 3 routines to run, one for each of the first 3 exposures.
fadc, fpga, and intdac (the prom exposure has no program – more below)
The C routines are in SEELotTest
```
>lnkgd fpga    (will compile and link fpga.c and leave the executable in BIN)
>fpga      (executes the routine)
```
You should not need to compile anything, but just in case …

The first irradiation should be to the Fast ADC
After logging on the laptop as kelby
$cd SEELotTest
$fadc ← starts the fadc monitoring program



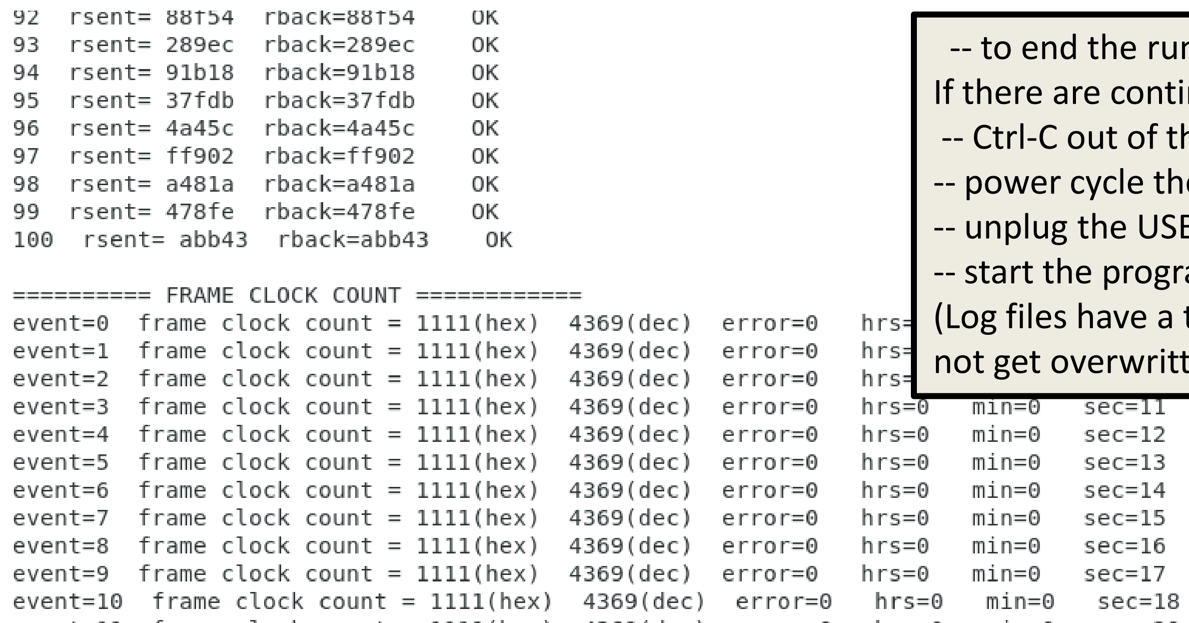
```
$ fadc
Successfully opened      tyUSB0
START  RUN intdac Mon Mar 20 08:10:12 2023

 1  rsent= b4567  rback=b4567    OK
 2  rsent= b23c6  rback=b23c6    OK
 3  rsent= c9869  rback=c9869    OK
 4  rsent= 34873  rback=34873    OK
 5  rsent=  dc51  rback= dc51    OK
 6  rsent= 95cff  rback=95cff    OK
 7  rsent= 8944a  rback=8944a    OK
 8  rsent= 558ec  rback=558ec    OK
 9  rsent= e1f29  rback=e1f29    OK
 10  rsent= 87ccd  rback=87ccd    OK
 11  rsent= b58ba  rback=b58ba    OK
 12  rsent= ed7ab  rback=ed7ab    OK
 13  rsent= 141f2  rback=141f2    OK
 14  rsent= 71efb  rback=71efb    OK
```

First 100 20bit random numbers are sent to the Controller FPGA and then read back. This tests that the USB to controller link is working.

Then the controller counts frame clocks from the FADC and compares to a direct clock count. The direct count stops both counters at

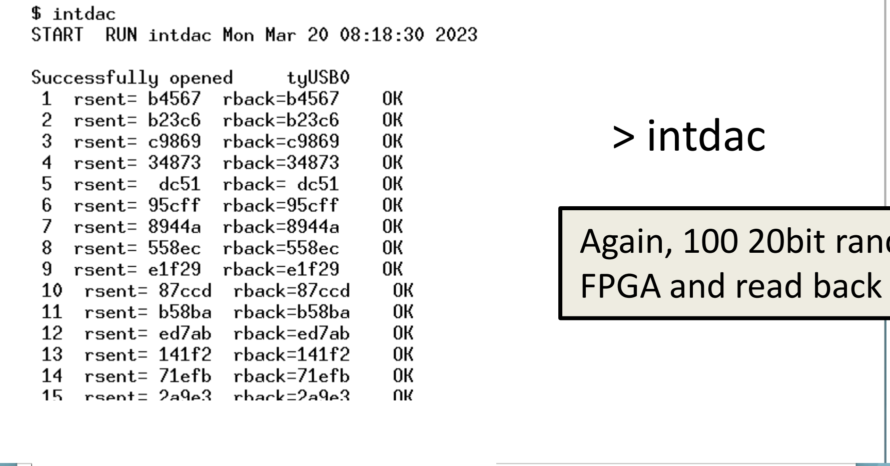
0x1111(hex) = 4369(decimal)

If the counts differ more then 1 count (to account for latency differences)
An error flag is set and the accumulative error count in incremented.
No errors are expected – none occurred in long weekend run tests



```
 92  rsent= 88f54  rback=88f54    OK
 93  rsent= 289ec  rback=289ec    OK
 94  rsent= 91b18  rback=91b18    OK
 95  rsent= 37fdb  rback=37fdb    OK
 96  rsent= 4a45c  rback=4a45c    OK
 97  rsent= ff902  rback=ff902    OK
 98  rsent= a481a  rback=a481a    OK
 99  rsent= 478fe  rback=478fe    OK
 100  rsent= abb43  rback=abb43    OK

========== FRAME CLOCK COUNT ============
event=0  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=
event=1  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=
event=2  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=
event=3  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=11
event=4  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=12
event=5  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=13
event=6  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=14
event=7  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=15
event=8  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=16
event=9  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=17
event=10  frame clock count = 1111(hex)  4369(dec)  error=0   hrs=0   min=0   sec=18
```

-- to end the run just Ctrl-C out
If there are continuing errors:
-- Ctrl-C out of the run
-- power cycle the board (long extension cord)
-- unplug the USB from the computer - then back in
-- start the program again to continue data collection
(Log files have a time stamp in their name so data does not get overwritten.)

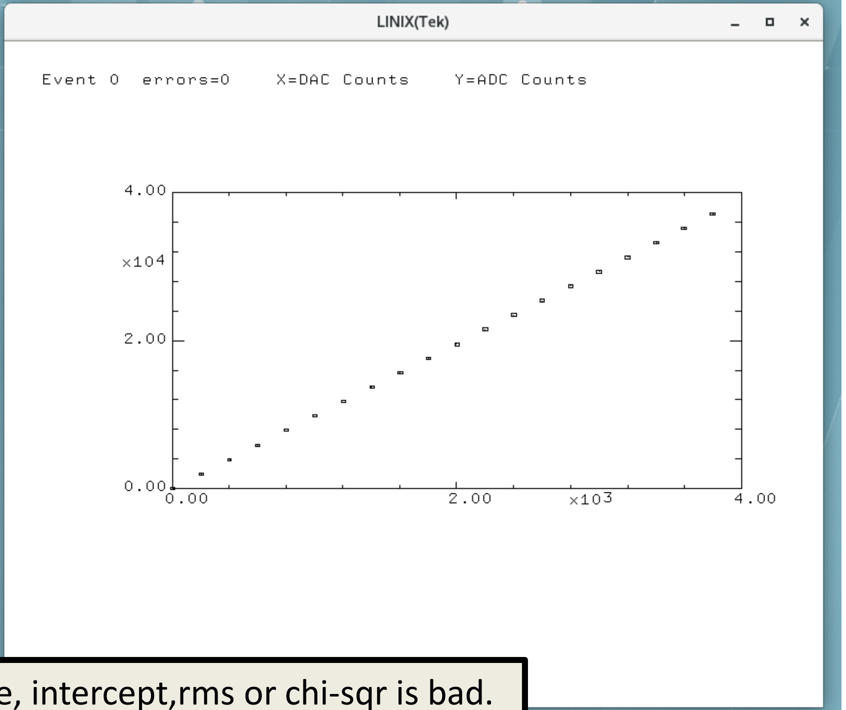


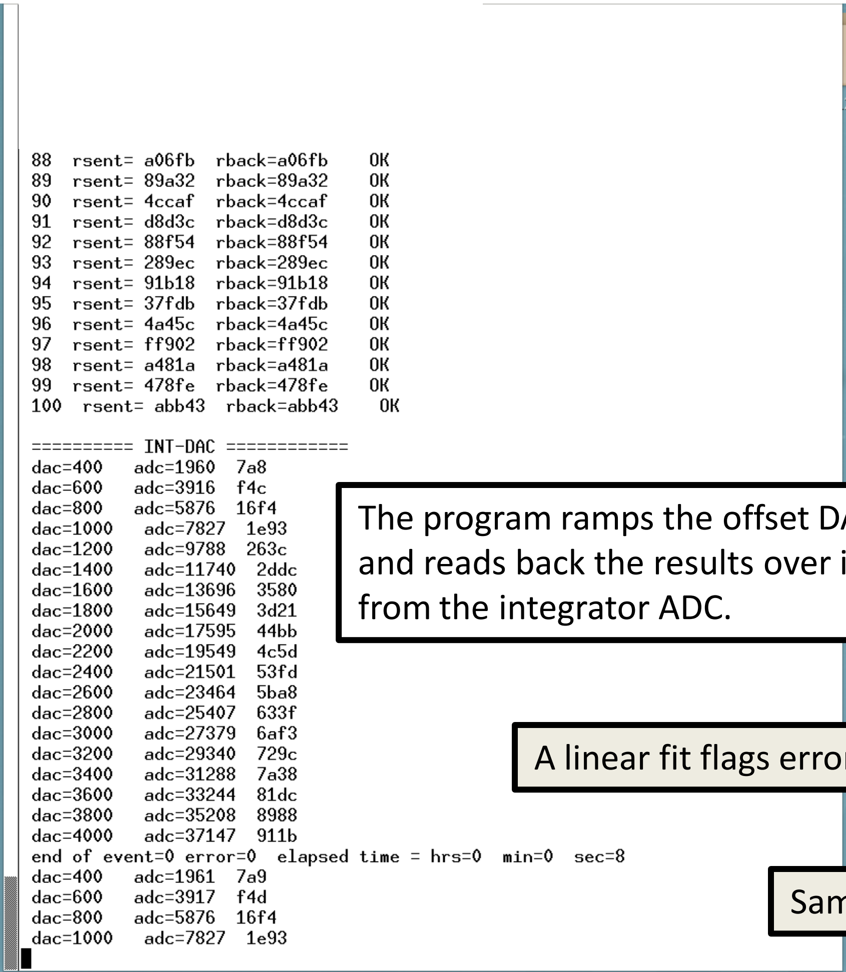
```
$ intdac
START   RUN intdac Mon Mar 20 08:18:30 2023

Successfully opened     tyUSB0
 1  rsent= b4567  rback=b4567    OK
 2  rsent= b23c6  rback=b23c6    OK
 3  rsent= c9869  rback=c9869    OK
 4  rsent= 34873  rback=34873    OK
 5  rsent=  dc51  rback= dc51    OK
 6  rsent= 95cff  rback=95cff    OK
 7  rsent= 8944a  rback=8944a    OK
 8  rsent= 558ec  rback=558ec    OK
 9  rsent= e1f29  rback=e1f29    OK
 10  rsent= 87ccd  rback=87ccd    OK
 11  rsent= b58ba  rback=b58ba    OK
 12  rsent= ed7ab  rback=ed7ab    OK
 13  rsent= 141f2  rback=141f2    OK
 14  rsent= 71efb  rback=71efb    OK
 15  rsent= 2a9e3  rback=2a9e3    OK
```

> intdac

Again, 100 20bit random numbers sent to the ctrl FPGA and read back to check the USB communication

This display updates with each voltage ramp.





```
 88  rsent= a06fb  rback=a06fb    OK
 89  rsent= 89a32  rback=89a32    OK
 90  rsent= 4ccaf  rback=4ccaf    OK
 91  rsent= d8d3c  rback=d8d3c    OK
 92  rsent= 88f54  rback=88f54    OK
 93  rsent= 289ec  rback=289ec    OK
 94  rsent= 91b18  rback=91b18    OK
 95  rsent= 37fdb  rback=37fdb    OK
 96  rsent= 4a45c  rback=4a45c    OK
 97  rsent= ff902  rback=ff902    OK
 98  rsent= a481a  rback=a481a    OK
 99  rsent= 478fe  rback=478fe    OK
 100  rsent= abb43  rback=abb43    OK

========== INT-DAC ============
dac=400   adc=1960  7a8
dac=600   adc=3916  f4c
dac=800   adc=5876  16f4
dac=1000   adc=7827  1e93
dac=1200   adc=9788  263c
dac=1400   adc=11740  2ddc
dac=1600   adc=13696  3580
dac=1800   adc=15649  3d21
dac=2000   adc=17595  44bb
dac=2200   adc=19549  4c5d
dac=2400   adc=21501  53fd
dac=2600   adc=23464  5ba8
dac=2800   adc=25407  633f
dac=3000   adc=27379  6af3
dac=3200   adc=29340  729c
dac=3400   adc=31288  7a38
dac=3600   adc=33244  81dc
dac=3800   adc=35208  8988
dac=4000   adc=37147  911b
end of event=0 error=0  elapsed time = hrs=0  min=0  sec=8
dac=400   adc=1961  7a9
dac=600   adc=3917  f4d
dac=800   adc=5876  16f4
dac=1000   adc=7827  1e93
```

The program ramps the offset DAC via SPI and reads back the results over i2c from the integrator ADC.

A linear fit flags errors if the slope, intercept,rms or chi-sqr is bad.

Same reset and resume run if things go bad.

```
79   rsent= ad36b  rback=ad36b    OK
80   rsent= 17796  rback=17796    OK
81   rsent= bd78f  rback=bd78f    OK
82   rsent= ea438  rback=ea438    OK
83   rsent= 5585c  rback=5585c    OK
84   rsent= 64e2a  rback=64e2a    OK
85   rsent= 342ec  rback=342ec    OK
86   rsent= 87cb0  rback=87cb0    OK
87   rsent= ed43b  rback=ed43b    OK
88   rsent= a06fb  rback=a06fb    OK
89   rsent= 89a32  rback=89a32    OK
90   rsent= 4ccaf  rback=4ccaf    OK
91   rsent= d8d3c  rback=d8d3c    OK
92   rsent= 88f54  rback=88f54    OK
93   rsent= 289ec  rback=289ec    OK
94   rsent= 91b18  rback=91b18    OK
95   rsent= 37fdb  rback=37fdb    OK
96   rsent= 4a45c  rback=4a45c    OK
97   rsent= ff902  rback=ff902    OK
98   rsent= a481a  rback=a481a    OK
99   rsent= 478fe  rback=478fe    OK
100  rsent= abb43  rback=abb43
```

> fpga
> starts with the usual 100 20bit random number test of the USB interface.

Through a shift register chain, 20bit random numbers are transferred from the ctrl-fpga to the radiated-fpga. The data in the radiated fpga is held for several seconds and then it is clocked back to the crl-fpga where is in compared to what was sent. If they do not match an error is flagged. In tests at Los Alamos we did see some errors where a 1 flipped to a zero.
Some will probably be seen at the lower flux for the lot testing at PSI. They recovered after 1 or 2 rewrites.

If errors persist use the same resets to resume the run.

```
==== hold pattern 2 sec in radiated FPG ====
event=0   error=0   sent=240fb   back=240fb   hrs=0   min=0   sec=8    OK
event=1   error=0   sent=26fa   back=26fa   hrs=0   min=0   sec=10    OK
event=2   error=0   sent=1deaa   back=1deaa   hrs=0   min=0   sec=12    OK
event=3   error=0   sent=6c33a   back=6c33a   hrs=0   min=0   sec=14    OK
event=4   error=0   sent=685fb   back=685fb   hrs=0   min=0   sec=16    OK
event=5   error=0   sent=6a529   back=6a529   hrs=0   min=0   sec=18    OK
event=6   error=0   sent=eedd1   back=eedd1   hrs=0   min=0   sec=20    OK
event=7   error=0   sent=a3fe6   back=a3fe6   hrs=0   min=0   sec=22    OK
event=8   error=0   sent=ef005   back=ef005   hrs=0   min=0   sec=24    OK
event=9   error=0   sent=9c13c   back=9c13c   hrs=0   min=0   sec=27    OK
event=10   error=0   sent=bb77c   back=bb77c   hrs=0   min=0   sec=29    OK
event=11   error=0   sent=ac794   back=ac794   hrs=0   min=0   sec=31    OK
```

The final exposure is to the EPCQ4A. No programs runs during exposure.
This device programs the FPGA at power up.
After the exposure ends, cycle the power and then
Run >fpga for a few minutes.

Back at Chicago, we will test that the EPCQ4a reprograms OK.
We will also do further tests on the FADC beyond the frame clock counting.

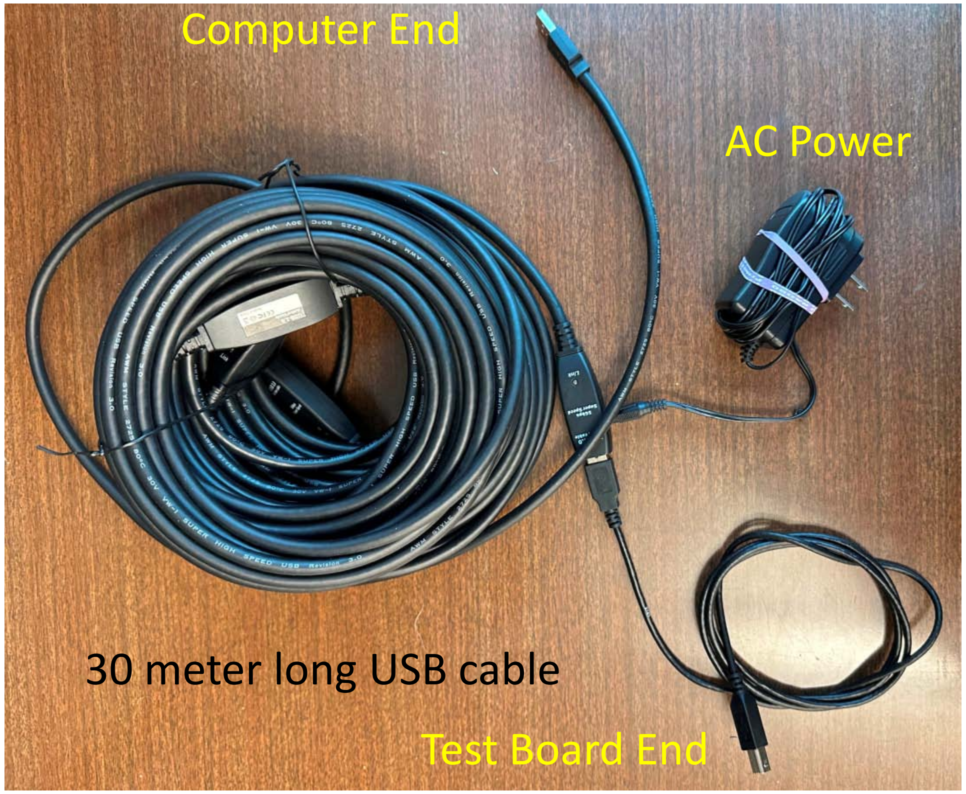
Please put the log files on a stick memory separate from the computer.

I will need like the computer and boards sent back to Chicago.

I think the log files could be emailed to me.

Thanks everyone.

PS: the mounting holds of the 2 boards are grounds. One should be connected to local ground. You should check everything out at CERN before heading for the PSI.



I will send 3 of these US to Swiss adaptors.

TestBoard Power
(caution yellow is gnd
black is +10VDC)

Computer Power

You will need to come up with a long AC power cord.