GRINCH DAQ Instructions

Scott Barcus
Last Updated: 10/6/16
E-mail: skbarcus@email.wm.edu

Introduction

The following instructions will describe how to utilize the new VETROC based DAQ system for the GRINCH detector. These instructions include:

1. How to operate the high voltage (HV) supply for the PMTs.
2. How to turn on each of the power supplies with their proper settings and the pulser.
3. How to turn on CODA and take a data run.
4. How to analyze the data file from a data run.
5. How to read and understand the ROOT file produced by the decoder.
6. How to use and understand the event display.
7. Some brief comments on the readout list (ROL) for the VETROC, GTP, etc.
8. Remote computer access.
9. Data flow through the VETROC DAQ.
10. Cable Map for the VETROC DAQ Setup.

High Voltage

The high voltage is used to power the PMTs in the prototype or GRINCH detector. It can be operated as follows:

1. Log into the computer currently in the ESB named grinchbb as user ADAQ. The password can be found on a sheet of yellow paper on the table by the computer or on the underside of the keyboard.
2. Change directories to /home/adaq/slowc for all processes associated with the high voltage.
3. In slowc open a new terminal which will be used to log into the HV server.
   (a) From this terminal type *ssh pi@rpi16* to log into the server as user *pi*.
   (b) You will be prompted for a password which can be found in the README file in slowc or on the underside of the keyboard.
   (c) Once you are logged into the HV server type *./start_hv* to start the server. You should see some output in the terminal with the server. You will get an error if another instance of the server is already running.
4. Next open a new terminal in the slowc directory and type *./hvs GRINCH* to cause the HV GUI to display.
   (a) In this GUI to allow tubes to be turned on/off one must click the *ON* button above the *HV ON* button on the left-hand side. Once this is clicked the *HV ON* button should turn yellow.

(b) To power an individual HV channel (which may power multiple tubes) select the tab of the HV supply desired from S11-S15. Currently we are using only S14 (L3.0-L3.6) and S15 (L4.0-L4.6). In the column labelled *Target_v* in the row of the channel you wish to power type the desired value in volts you want the channel to supply to the connected PMTs.

(c) Once the target voltage has been entered in the same row in the column *Ch_En* click the box (a check mark will appear in the box). This channel is now enabled. After a few seconds in the columns *Meas_uA* and *Meas_V* you should see values appear indicating the amperage that channel is using and the voltage it is supplying.

(d) To disable the channel simply click the check marked box again and the check mark should disappear and the measured voltage should return to zero.

5. Note: The HV GUI often crashes or disappears. To bring it back ctrl c in the terminal that you typed *./hvs GRINCH* in and then reenter *./hvs GRINCH*. This should reboot the GUI.

6. Note: If the server is frozen and refuses to update values in the GUI and restarting the GUI fails try turning off the HV server manually on the back of the HV box and then turning it back on. You will then need to repeat all of the steps to start the server after it reboots.

7. Below is a table with the HV values currently being supplied to the PMTs in the prototype detector. These values are also saved in and can be loaded from GRINCH_prototype_2016MMDD in the slowc directory.

Table 1. High Voltage Values for PMTs

HV Box	HV Channel	Voltage (V)
S14	L3.0	1062
S14	L3.1	0
S14	L3.2	1100
S14	L3.3	1140
S14	L3.4	1190
S14	L3.5	1230
S14	L3.6	1383
S15	L4.0	840
S15	L4.1	905
S15	L4.2	930
S15	L4.3	965
S15	L4.4	970
S15	L4.5	978
S15	L4.6	1005
S15	L4.7	1045

8. NOTE: Sometimes the HV crate seems to turn off the power to the tubes so always check the measured voltage to be sure the PMTs are getting power. If everything is on and there is no measured voltage try turning off the power with the button on the left of the GUI and then back on with the *HV ON* button. Otherwise you probably need to restart the HV server or the HV crate.

Power Supplies and Pulser

This next section will describe which units are used to power other components like the NINO cards and how to operate the pulser.

1. NINO Cards Power Supply (Keysight N5741A)

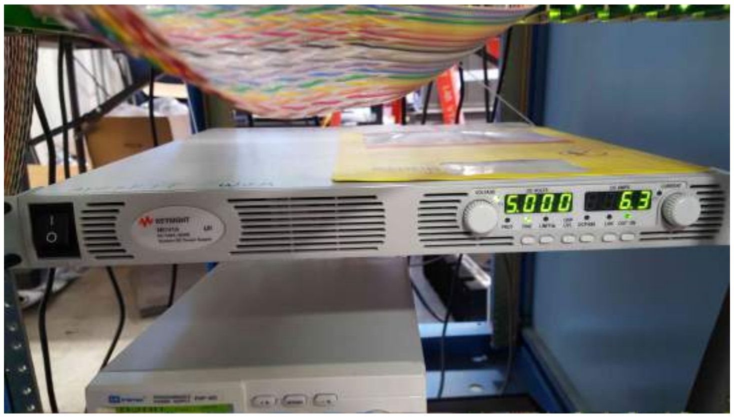


Figure 1. NINO Cards Power Supply. Keysight N5741A.

   (a) Turn on the power supply by flipping the power switch on the left-hand side.
   (b) After switching on the main power you must then press the *OUT ON* button on the lower right of the power supply to enable the output.
   (c) Adjust the *VOLTAGE* knob to 5.000 V while adjusting the *Current* knob upwards as needed to reach 5 volts. You will need to turn on the *FINE* button to reach exactly 5.000 V.

2. NINO Threshold Power Supply (INSTEK PSP-405)

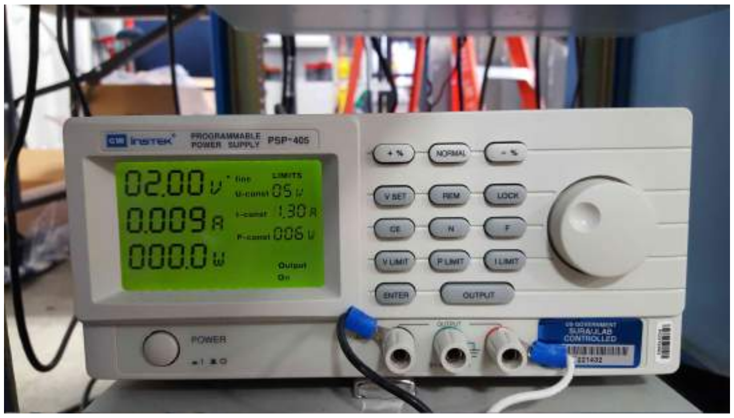


Figure 2. NINO Cards Threshold Power Supply. INSTEK PSP-405.

   (a) Turn on the spower supply by pressing the power button in the lower left.
   (b) Press the *OUTPUT* button on the lower right to enable the output. On the screen the in the lower right under OUTPUT it should now say ON.
   (c) Turn the knob on the right to increase the voltage to the desired level. Currently we are using 2.00 V and drawing about 0.009 A.

(d) It is best to set voltage and current limits on this power supply to be certain to not accidentally send too much power to the NINOs. Current limits are set at 5.0 V and 1.3 A.

(e) For further information on how the external NINO threshold power changes the minimum threshold the NINOs apply to the signal from the PMTs consult page 5 of `http://hallaweb.jlab.org/12GeV/SuperBigBite/SBS-minutes/2015/Montgomery_NINO_300915.pdf`.

3. OPTIONAL LVDS-ECL Translator Power Supply (Topward 6306D-10)

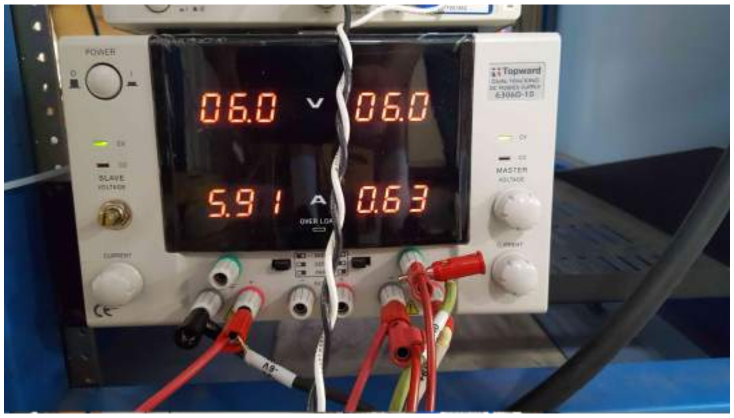


Figure 3. LVDS-ECL Translator Power Supply. Translates LVDS signal from NINO cards into ECL signal for viewing on an oscilloscope.

(a) This component is only needed if you want to translate the LVDS output from the NINO cards into an ECL signal so it can be viewed on an oscilloscope.

(b) Turn on the power supply by pressing the power button in the top left.

(c) Place the power supply in *SER* (series mode) by moving the center-left black switch to the rightmost position and the center-right switch to the leftmost position.

(d) By turning the *Master Voltage* knob on the right-hand side, which should now control both output channels, adjust the output such that each channel is outputting 6.0 V. You may need to increase the *Master Current* knob to accommodate this voltage.

(e) With the current output configuration the left channel should say 6.0 V and ∼5.90 A while the right channel reads 6.0 V and ∼0.63 A.

4. Pulser (HP 8160A)

(a) Turn on the pulser by pressing the *LINE* button on the lower left side of the pulser.

(b) NOTE: When the pulser turns on the output channels will automatically be disabled. To enable the outputs press the *DISABLE* button on the right-hand side of the pulser. The LEDs indicating output channels A and B should now be lit.

(c) To set the period for the pulser first under *Parameter* press *PERIOD*, then under *CHANNEL/DATA* enter the numerical value for the period desired, and finally under *ENTRY* press the button corresponding to the units you wish to apply to the value you entered.

(d) To set the pulse width first under *Parameter* press *WIDTH*, then under *CHANNEL/DATA* press which channel's (A, B, or C) width you want to set (this is an extra step from setting the period). Next after the channel is has been selected also under *CHANNEL/DATA* enter the value for the width desired, and finally under *ENTRY* press the button corresponding to the units you wish to apply to the value you entered.

(e) Other parameters such as the output voltages (*HIL* and *LOL*) and the leading and trailing edge slopes (*LEE* and *TRE*) are set in the same manner as the pulse width, but instead of pressing *WIDTH* first one of the other parameters is selected instead.

(f) NOTE: The pulser should save the settings it was last using when it is turned back on.

(g) This pulser has many more features more advanced than single square pulses such as bursts of pulses. How to operate these other modes can be found in the user's manual for the pulse at `http://literature.cdn.keysight.com/litweb/pdf/08160-90002.pdf?id=806426`. Note: that in the user's manual sections 3-1 to 3-12 appear to be missing. They are simply out of order and can be found on page 3-1 after section 4-14 on page 4-14.

Starting CODA and Taking a Data Run

This section will describe how to start CODA as well as how to use CODA to take a data run. Opening and running CODA currently requires six different terminals (a macro could be created to simplify this).

1. Opening CODA GUI before taking data.

   (a) In /home/adaq/Desktop (other directories are fine too) open a new terminal and type msqld to load the msql database. Once you have done this the terminal should say *ACL load done*. As long as you don't kill this terminal or end the process you will only have to repeat this step if you restart the computer.

   (b) Open another new terminal in the same directory and type *startcoda*. This should pop up several terminals for running CODA. Alternatively you can do each of these steps individually by following the steps outlined below in (c), (d), and (e). NOTE: If you do step (b) do not do optional steps (c), (d), and (e) and vice versa.

   (c) OPTIONAL: Open another new terminal in the same directory and type *platform*. The terminal will print some text.

   (d) OPTIONAL: Open another new terminal in the same directory to run the event recorder and type *coda_er_rc3 -i -n ER10 -t ER*. There will be some output in this terminal and the terminal running the platform will say *ER10 Host registered*.

   (e) OPTIONAL: Open another new terminal in the same directory to run the event builder and type *coda_eb_rc3 -i -n EB10 -t CDEB*. There will be some output in this terminal and the terminal running the platform will say *EB10 Host registered*.

   (f) Open another new terminal to run the readout controller (ROC) in. The ROC is run on the intelvmeha6 computer next to the scaler and the VETROC in the rack with the user adev (i.e. adev@intelvmeha6). This can be accessed more easily from the grinchbb computer by typing the alias *govme* into the new terminal and then entering the password for intelvmeha6 which can be found on the yellow piece of paper with the other password or underneath the keyboard.

   (g) Once logged into intelvmeha6 in the directory /home/adev type *./startroc9* to turn the ROC on. There will be some output in this terminal and the terminal running the platform will say *ROC9 Host registered*.

(h) Open another new terminal in the same directory to run the CODA GUI. In this new terminal type *rcgui* and the CODA GUI should pop up.

(i) At this point you may want a 7th terminal to kill coda from if it starts having issues (and it will). In this 7th new terminal you can type *kcoda* to kill the CODA GUI and the processes associated with it (i.e. the platform, the ER, and the EB). Anytime you kill CODA you will need to restart the platform, ER, EB, ROC9 and the CODA GUI (rcgui). ROC9 doesn't stop automatically when you *kcoda* so you will have to ctrl c the terminal running ROC9 and then restart it before you can take data again. The msql database does not need to be restarted when you *kcoda*.

2. Taking data once the CODA GUI is open.

(a) Now that the CODA GUI is open click *Platform->Connect* in the top left.

(b) Press the *Configure* button below *Platform* (it looks like a wrench crossing a screwdriver). Wait until in the CODA GUI under *State* ER10, EB10, and ROC9 all say *configured*.

(c) After all the states read configured wait several seconds because CODA gets locked up if you give it commands too quickly and you'll have to restart. After the short wait press the *Download* button next to the *Configure* button. After a few seconds the states for ER10, EB10, and ROC9 should all read *downloaded*.

(d) You are now ready to start a data taking run. Press the *Start* button and the data run will begin. You can tell is the system is seeing triggers by looking at the trigger interface (TI) in the rightmost slot in the VME crate. If it has two lights lit up it is running but not seeing any triggers, but if there are three lights then the system is seeing triggers and recording data (see figure 4). You can also see the data rate increase in the CODA GUI if there is data being recorded.

(e) After the desired number of events are recorded you can stop the data run by pressing the *END* button. The run state will now say ended.

(f) NOTE: the number of events currently recorded is shown on the CODA GUI under *Total Events*. These events increase in increments of 50 since the system buffers 50 events before dumping them to the file. There are also an additional three events per run for the prestart, start, and end processes. Lastly, as will be described when discussing the readout list for the DAQ system, each event listed in the CODA GUI is currently equal to 100 real data events (100 triggers) because the system is set to read out data in blocks of 100 triggers seen. Thus one event in the CODA GUI is really 100 triggered events from the VETROC. One block equalling 100 events can be changed in the readout list and can probably be further optimized to increase the maximum data rate.

(g) One can preset the number of events for CODA to record by going to *options* then *scheduler* then *program*. This will provide several options where preset limits can be created for *time*, *number of events*, or *number of runs*. Generally it is most useful to select a maximum number of events.

(h) The run number is given on the CODA GUI where is says *Run Number*. For example if the run number was 1410 the output data file would currently be vertoc_1410.dat.

(i) To take another run simply press the *Start* button again.

(j) If there are errors during this process first try using the *reset* button and then re-downloading before running again. If this still doesn't fix the issue kill coda with *kcoda* and repeat the steps after loading the msql database remembering to kill the ROC9 process as well. If this still doesn't fix the problem power cycle the crate with the VETROC and intelvmeha6 in it and

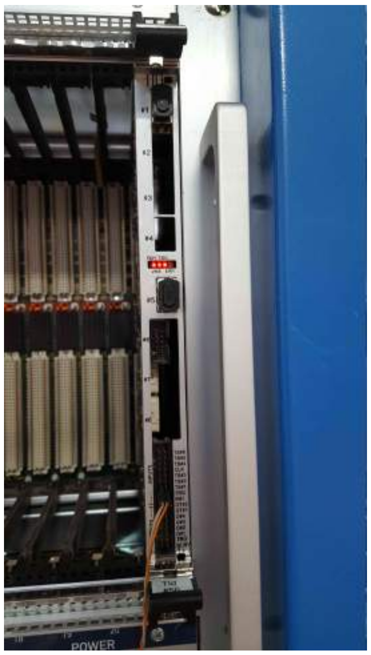


Figure 4. TI taking Data. This image shows the TI while it is taking data. Three lights are lit indicating data is being seen and recorded.

then repeat all the steps after loading the msql database (you should not need to reload the msql database from power cycling the crate). Note that when you switch the crate off and on intelvmeha6 will be unavailable for a few minutes while it reboots. Finally after power cycling the crate the first data run you take almost always runs but sees no data (two lights only). I'm not sure why this happens but it usually does. This is easily fixed by ending the first run and then pressing the *Reset* button and then re-downloading and starting a new run.

Analyzing a Data Run

After data has been recorded and CODA produces a raw data file this file must be decoded into a useful readable format. The raw data file, labelled vetroc_run#.dat and stored in directory /home/adaq/data, contains the data words from the VETROC that represent different parts of the data run such as the beginning and end of events as well as the actual TDC timings of those events. The decoder takes these words and translates each of them to their corresponding meanings and then organizes them in a ROOT file in the form of TTrees, histograms, etc. These ROOT files, labelled vetroc_run#.root and stored in /home/adaq/vetroc_decode, can then be used for physics analyses.

1. Change directories to /home/adaq/vetroc_decode.

2. The decoder is called VETROC_main1.C.

3. The decoder is run by calling the *newrun* macro using the form: *./newrun [run#] [# events to analyze]*. I strongly advise piping the output to a file since there is a lot of output and it will take an extremely long time to print to the terminal. For example say we want to analyze the first 700

events in the raw data file vetroc_1410.dat and then pipe the output to the text file called NULL we would type *./newrun 1410 700 >> NULL.*

4. It will take several minutes for the decoder to analyze the file. Once it is finished it will have created a ROOT file of the same name as the .dat file in the vetroc_decode directory.

5. To examine the raw data file (vetroc_runnumber.dat) type *xcefdmp* into a terminal. This will pop up a GUI that will allow for the examination of the individual data words recorded by CODA. To examine this raw data click *View File* then go to *Data Source* and select *Open* then in the window that opens up in the box under *Filter* change the source to */home/adaq/data*. Now in the box under *Files* select the name of the data file you wish to examine. This will take you back to the main Xcefdmp window with the file you selected ready for examination. To navigate through the events in the data file use the *View Next* and *View Previous* buttons or enter an event number in the *Event Number* box. The individual data words can be seen by clicking the *0x3* box on the right of the screen. NOTE: Events one and two are just prestart and go events and as such contain no data. Also each of the 'events' in the Xcefdmp is equal to the number of triggers in one event block set in the readout list. For example if the block level is 100 (as it currently is) one event in Xcefdmp will contain 100 triggers and their associated data.

Reading/Understanding the ROOT File

After the decoder has been run on the raw data file a ROOT file with various histograms and TTrees is produced in the vetroc_decode directory.

1. To open the ROOT file and examine the histograms in the same directory as the ROOT file type *root -l vetroc_1410.root*. This will open ROOT while attaching the file vetroc_1410.root so it's contents can be viewed.

2. Next open a new TBrowser in ROOT by typing *TBrowser b* in the ROOT command line.

3. A new browser should now pop up. On the left-hand side of the browser select the name of the ROOT file attached to the current ROOT session. This file should expand to show various histograms and TTrees containing the decoded data. Many of these are used mostly for debugging code and checking data quality so I will only describe a few below.

   (a) The items labelled eventtree are used for the event display. They contain data on each event that occurred during the data run. An event is all of the data recorded due to a single trigger being seen. Each event contains a 2D array which stores which PMTs fired and the TDC times associated with those firings.

   (b) The items labelled TDC Data Channel # each hold the TDC data for a single PMT channel over the whole run. For channels observing large amounts of LED light there will be two peaks in the channel (one for the leading and one for the trailing edge of the pulse). The first peak should be a bit sharper than the second as the second peak represents a time over threshold measurement making it useful as a rough ADC as well.

   (c) The histogram hhits shows a primitive event display indicating which channels saw hits during the data run. Note the main event display is a separate piece of code which uses the .root files.

   (d) The histogram hchnlhits shows how many hits each PMT channel saw during the data run.

Using/Understanding the Event Display

The event display is used to see which tubes fired, and how many times they fired, during a single event (one trigger). This will help us to identify if a cluster of tubes firing was a Cherenkov ring or not. It can also create a heat map of every PMT firing over all events to show the relative distribution of light on the PMT array. The current version of the event display being used is called Grinch_Prototype_Event_Display_Development.C and it can be found in the vetroc_decode directory. It is currently not fully functional on the grinchbb computer and this issue is being investigated. The event display is still a work in progress and has numerous issues.

1. To run the event display open ROOT in the vetroc_decode directory by typing *root -l*. Then in the ROOT terminal type *.x Grinch_Prototype_Event_Display_Development.C* to run the event display.
2. Now a small window which says *Event Displayer* should pop up. This window allows you to navigate through the events.
3. By pressing the *First* button after a few seconds a new window showing a visualization of the GRINCH prototype PMT array (a version of the event displayer with the full GRINCH geometry will have to be made). You are now looking at event zero. PMTs that fired will be shaded red and above the PMT number is a number that indicates how many hits that PMT saw during this event.
4. The user can step forward and backward through events by using the *First*, *Next*, *Prev*, and *Last* buttons.
5. The buttons *Next with Hit* and *Prev with Hit* will step forward and backward between only events which saw hits. Currently there will be hits in every event as the trigger condition is only met if there is a hit in the reference channel (currently 112). Once the cluster finding is being used this will allow the user to step between only events with valid clusters. (This will require some modifications in the event display code before it works.)
6. The *Event* button allows the user to enter in the terminal the number of the event they wish to view.
7. The *All Events* button will create a new pop up that is a heat map of every PMT firing over all events to show the relative distribution of light on the PMT array. This function can take upwards of 10 minutes to execute depending on the length of the data run.

DAQ Readout List

The readout list (ROL) for the DAQ is located on the intelvmeha6 computer. It is called vme_block1.c and can be found in the directory /home/adev/linuxvme/vetroc/rol. To login as adev@intelvmeha6 type the alias *govme* into a terminal on the grinchbb computer and then enter the password for intelvmeha6 which can be found on a yellow piece of paper next to the grinchbb computer or on the underside of the keyboard. The ROL is responsible for controlling and setting up the various components of the DAQ such as the VETROC, TI, and GTP.

1. BLOCKLEVEL defines the number of events (single triggers) that will be buffered before the data is read out from the VETROC. Currently it is set to 100 events per block, but it may be possible to optimize the data output rate (and thus the maximum data rate the system can run at) by trying different events per block.
2. BUFFERLEVEL defines the number of blocks that can be held in the buffer at once. It is currently set to 10 but could also be optimized.

3. VETROC_SLOT defines the VME crate slot that the VETROC is in. It is currently in slot 3.

4. The function vetrocGSetProcMode(2750,2000) determines the location of the TDC data in the timing window and the size of the timing window. The second parameter determines the size of the TDC data window in nanoseconds. Currently it is set to 2000 ns so all TDC times must fall between 0 ns and 2000 ns to be recorded. The first parameter gives an offset for the TDC data in the timing window. It is currently set to 2750 ns. If it was set to 3250 ns it would shift the TDC data 500 ns to the right. Note: if the offset is too large or too small, i.e. outside of the timing window, the TDC data will not be recorded. When a data run is taken but the histograms are empty it is possible that these parameters need to be adjusted to bring the TDC data back into the data window.

How to Remotely Access the grinchbb Computer

Often it can be helpful and convenient to access the grinchbb computer remotely (including off site) to modify the code or access data. This is done as follows:

1. First open a new terminal on your own machine. Then log into the JLab computer system by typing *ssh -X yourusername@login.jlab.org* and then entering your personal JLab password. You should now be in the JLab system.

2. Next you need to login to the actual grinchbb computer currently in the ESB. Do this by typing *ssh -X adaq@grinchbb* and then entering the password for the grinchbb computer. This password can be found either on a yellow sheet of paper near the computer or on the underside of the keyboard.

3. You should now be logged into the grinchbb computer as the user adaq. You can now access and modify files as usual.

4. Note: Be cautious when modifying files remotely if that file is open on the actual grinchbb monitor at the same time since the open file you did not modify will not update until it is closed and reopened.

Data Flow Through the VETROC DAQ

This section will briefly explain how the data flows through the VETROC DAQ from the data source all the way to being collected by CODA. The diagram in Figure 5 shows how the data flows through the DAQ. First the data is collected from the data source (e.g. a detector, pulser, etc.) and sent into the VETROC through the front panel. The VETROC accepts differential inputs (e.g. ECL and LVDS signals). Once the data enters the VETROC it is held in a pipeline. The VETROC streams this data to the GTP through the back-plane and the GTP scans the data looking for trigger conditions.

When the GTP finds a trigger it sends a trigger signal to the VETROC on an external cable. In this case the VETROC is only being used as a level translator to convert the GTP trigger signal from LVDS to ECL. The translated ECL trigger signal is then taken out of the VETROC and sent to the TI. Once the TI receives the trigger signal it stops (only if in "ROC LOCK" mode; otherwise triggers are allowed to be accepted up to the set buffer limit) and then reads out the data from the VETROC over the back-plane which is then dumped to CODA. NOTE: I'm not entirely sure what role the signal distributor (SD) plays. It distributes the trigger from the TI, but I don't know if it does this when only one VETROC is used or if it is only needed when multiple VETROC boards are in operation.

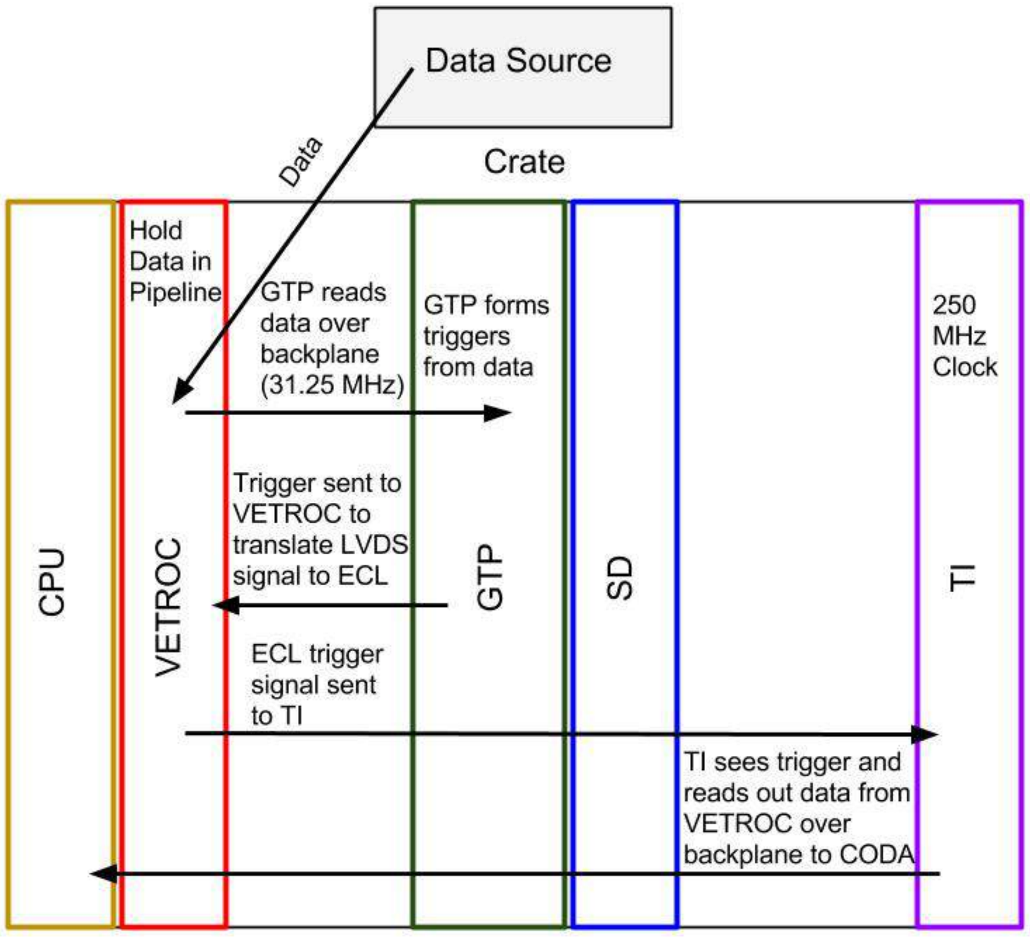

Data Source
Data
Crate
Hold Data in Pipeline
GTP reads data over backplane (31.25 MHz)
GTP forms triggers from data
250 MHz Clock
Trigger sent to VETROC to translate LVDS signal to ECL
CPU
VETROC
GTP
SD
TI
ECL trigger signal sent to TI
TI sees trigger and reads out data from VETROC over backplane to CODA

Figure 5. VETROC DAQ Data Flow Diagram. This figure shows how data flows from some data source like a detector through the components in the VXS crate and ultimately is collected by CODA.

Cable Map for the VETROC DAQ Setup

This section will describe how to properly cable all of the VETROC DAQ components together to perform tests using the prototype GRINCH detector. Figure 6 shows a diagram of which components need to be wired to which other components (some details are omitted for space and are explained further in this section). NOTE: Some components use the VXS back-plane to communicate either instead of or in addition to using external wires. For example the data in the VETROC is read through the back-plane by the GTP.

First the CPU running CODA (currently grinchbb in the ESB) needs to be connected via Ethernet cable to the CPU controlling the VXS crate components (currently intelvmeha6) so that they may communicate. Next the CPU running CODA must also be connected via Ethernet cable to the GTP so that the firmware which determines the trigger conditions can be burned to the GTP.

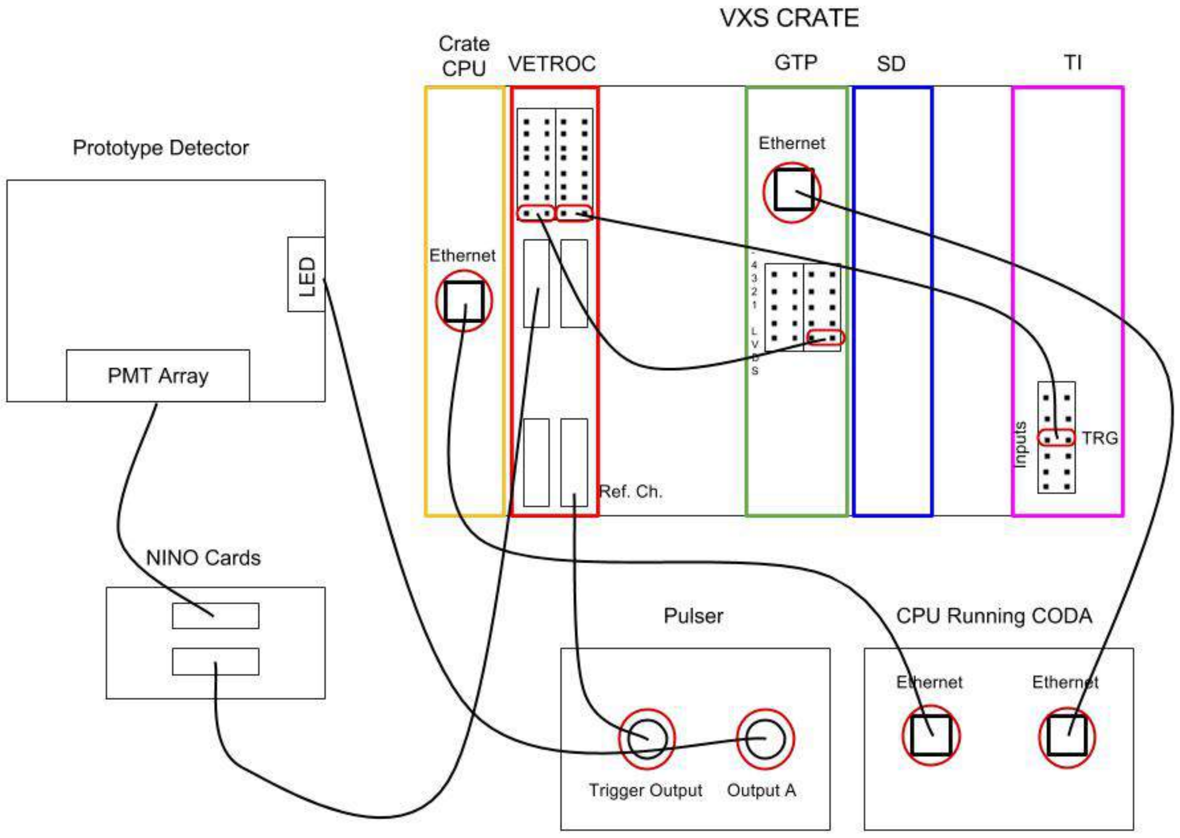


Figure 6. VETROC DAQ Cable Map. This figure shows how each component should be connected for using the VETROC TDC with the prototype GRINCH detector. NOTE: Some of these connections have subtleties such as needing to translate the signal to a different type (NIM, ECL, LVDS) or polarity.

Next the VETROC, GTP, and TI need to be connected. Figure 7 shows a wide view of this configuration and Figures 8, 9, 10, and 11 show close up views of each components' wiring. Each of these components are connected with ECL cable (the signal from the GTP to the VETROC is LVDS and from the VETROC to the TI is ECL). At the top of the VETROC there will be two ports with eight pairs of pins each. To connect the VETROC (to be used as a level translator for LVDS signal from the GTP to ECL for the TI) and the GTP place the ECL cable on the bottom set of pins on the upper-leftmost port of the VETROC. Next locate the middle two sets of ports next to one another on the GTP each containing five sets of pins by the label *-1234 LVDS*. Connect the other end of the cable attached to the VETROC to the lowest set of pins in the rightmost middle port on the GTP. To connect the VETROC to the TI place the ECL cable on the bottom set of pins on the upper-rightmost port of the VETROC then connect the other end of the cable to the TI set of pins labelled *TRG* located on the bottom part of the TI next to *INPUTS*. The TI and VETROC are now connected.

Now that the CPUs and VXS crate components are connected we must connect the prototype detector and the data it produces to the system. In the current setup we are using a pulser to power an LED in the prototype detector whose light is then detected by an array of 81 PMTs in this detector. The PMT output is transferred to NINO cards which amplify and discriminate the signal (although an analog

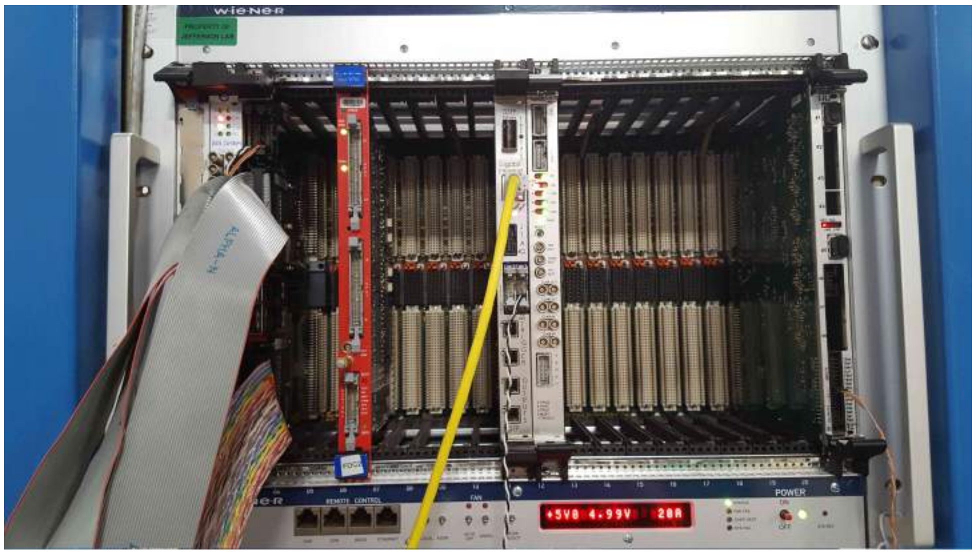


Figure 7. VXS Crate Configuration. This figure shows how the VXS crate looks when properly configured. NOTE: The red unit is an ADC and it is not ustilized in the setup described here.

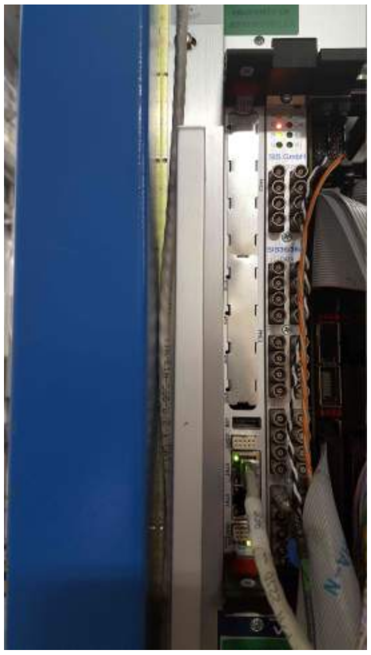
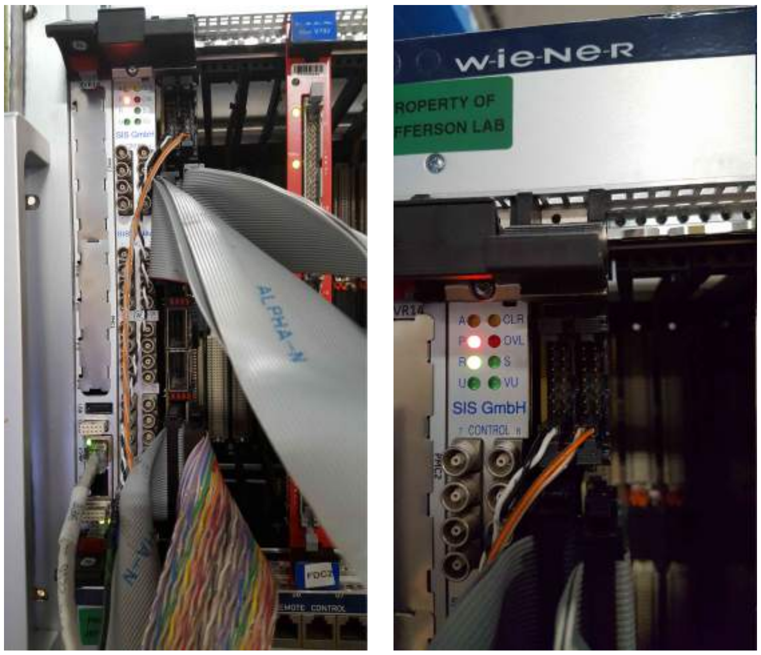


Figure 8. Close Up of VXS CPU. This figure shows a close up of the VSX CPU (left-most module) and its connections.

output can be chosen instead) and this output is then sent into the VETROC to be read out.

To begin making these connections take a NIM cable from the back of the pulser labelled *Output A*

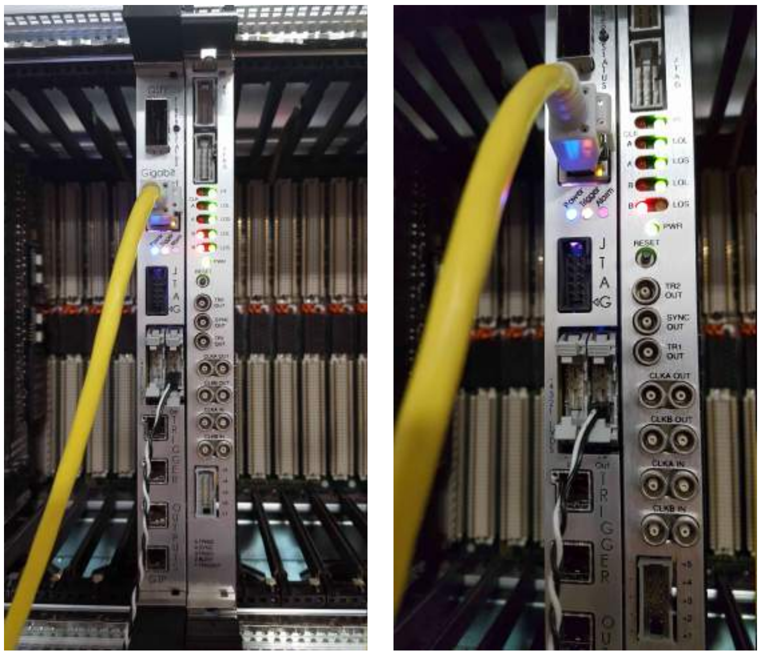


(a) VETROC Wiring (b) Zoomed VETROC Wiring

Figure 9. Close Up of VETROC. This figure shows a close up of the VETROC and its connections. The VETROC is the third module from the left with the ribbon cables going into it.

and attach the other end to the LED input on the side of the prototype detector. Figure 12 shows the backside of the pulser and the LED connector on the prototype detector. The light from this LED is seen by the PMTs inside the detector and their outputs should be connected to the NINO card inputs using NIM cables with special adapters to connect to the PMT bases. Figure 13 shows the connections between the PMTs and the NINO cards. Then to get the PMT signals from the NINO cards to the VETROC a ribbon cable should be inserted in the analog output of the NINO cards and the other end should be connected to one of the two (or four if the mezzanines giving additional channel input to the VETROC are attached as they currently are) ribbon cable ports in the upper or lower section of the VETROC. Figure 14 shows the ribbon cable coming out of the digital (amplified and discriminated) output of a NINO card. The VETROC ribbon cable connections can be seen in Figure 9. NOTE: The ribbon cables currently in use have 32 channels split into two different ribbons with each 16 channel single ribbon going to the output of different NINO cards which only have 16 channels each.

The final part of the DAQ which needs to be connected is an input to a given VETROC channel to be used as a reference channel for the other channels with 'real' PMT data. To do this take a NIM cable and plug it into the *Trigger Output* on the back of the pulser (Figure 12). We want to connect this output to a certain VETROC channel to be used as a reference, but first we must make the NIM signal compatible with the VETROC. The VETROC reads differential inputs signals like ECL or LVDS, but cannot read a NIM signal directly. Therefore, we must convert the NIM signal to ECL using a NIM-ECL converter (blue box LeCroy 4616). Unfortunately the converter only takes negative inputs so we must first invert the trigger output signal using the inverted output from a linear fan-in/out (in this case a Phelps Scientific model 778). So the trigger output from the pulser must be sent to a fan-in/out module and then the inverted signal must be sent to the NIM input of the NIM-ECL converter corresponding to the desired reference channel. Then from the ECL outputs on the converter a ribbon cable must be

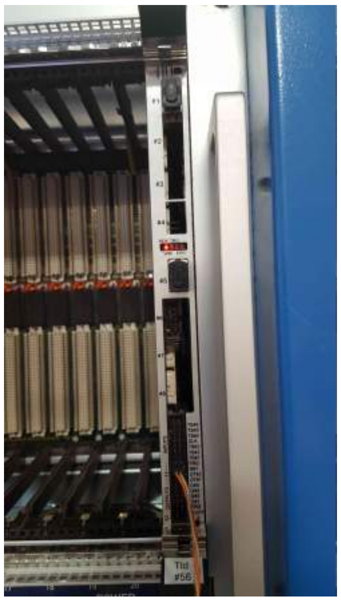


(a) GTP Wiring (b) Zoomed GTP Wiring

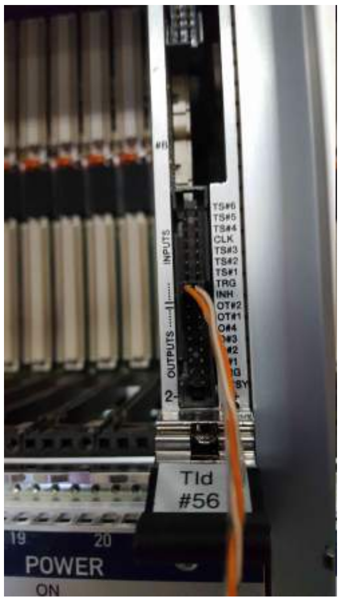
Figure 10. Close Up of GTP. This figure shows a close up of the GTP and its connections.

connected to the VETROC ribbon cable input corresponding to the desired reference channel. With this final addition to create triggers the DAQ should be ready to use. Figure 15 shows the wiring for the NIM modules and the ribbon cable plugging into the VETROC board can be seen in the close up in Figure 9.

Currently the DAQ is configured to trigger whenever VETROC channel 112 sees a signal. Channel 112 gets its signal from the *Trigger Output* of the pulser. This means that whenever the pulser sends the LED a pulse causing it to fire and be detected by the PMTs a simultaneous pulse is sent to the reference channel (112) telling the DAQ to record data (i.e. the light seen from the LED). This is the same idea as having the DAQ trigger only when a coincidence between several scintillators or other detectors occurs during an experimental run. NOTE: When we begin trying to trigger on clusters this method of using a single reference channel will need to be modified.

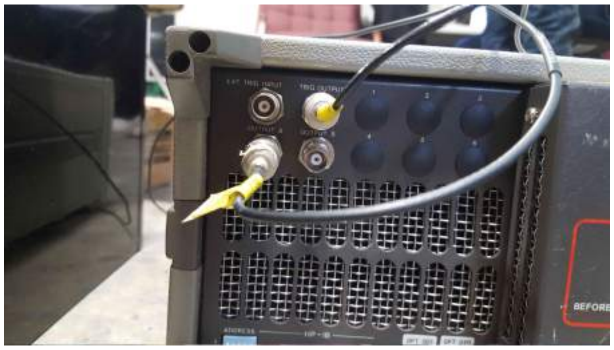


(a) TI Wiring

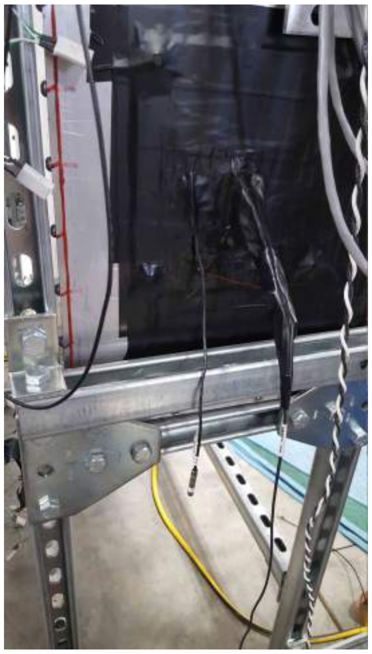


(b) Zoomed TI Wiring

Figure 11. Close Up of TI. This figure shows a close up of the TI and its connections.

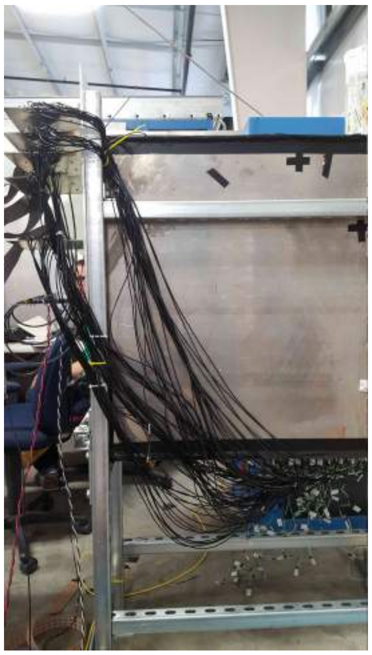


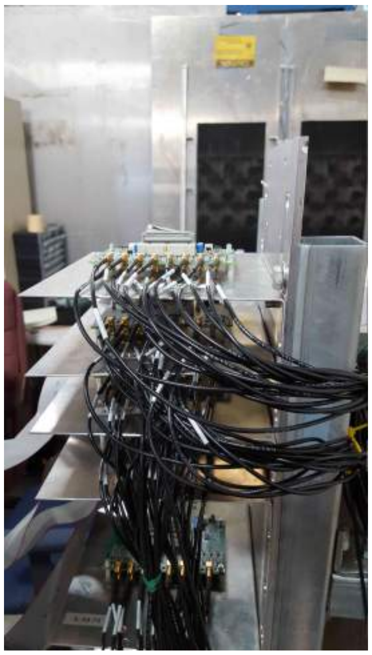
(a) Pulser Wiring



(b) LED Location

Figure 12. Close Up of Pulser Backside and LED Location. This figure shows a close up of the pulser backside and its wiring in (a) and the wiring connections to the LED in (b).


(a) PMT to NINO Wiring


(b) Zoom of NINO Wiring

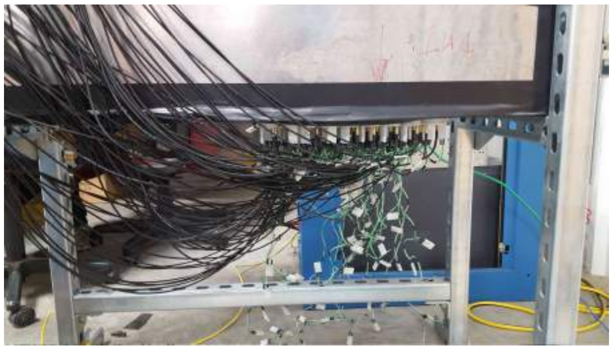
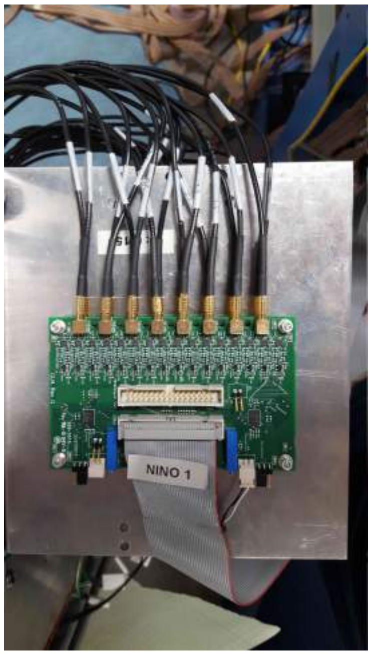

(c) Zoom of PMT Wiring

Figure 13. Connection of PMTs and NINO Cards. This figure shows the wiring connecting the PMTs in the prototype detector to the NINO cards in (a), and (b) and (c) show zoomed views of the NINO side of the wiring and the PMT side of the wiring respectively.



(a) Digital NINO Output

(b) NINO to VETROC Wiring

Figure 14. Digital NINO Output to VETROC. This figure shows a ribbon cable taking the PMT signals from the amplified and discriminated digital output of a NINO card in (a). The open ribbon cable port above the filled one in (a) is the NINO analog output. In (b) the ribbon cable can be seen going to the VETROC.

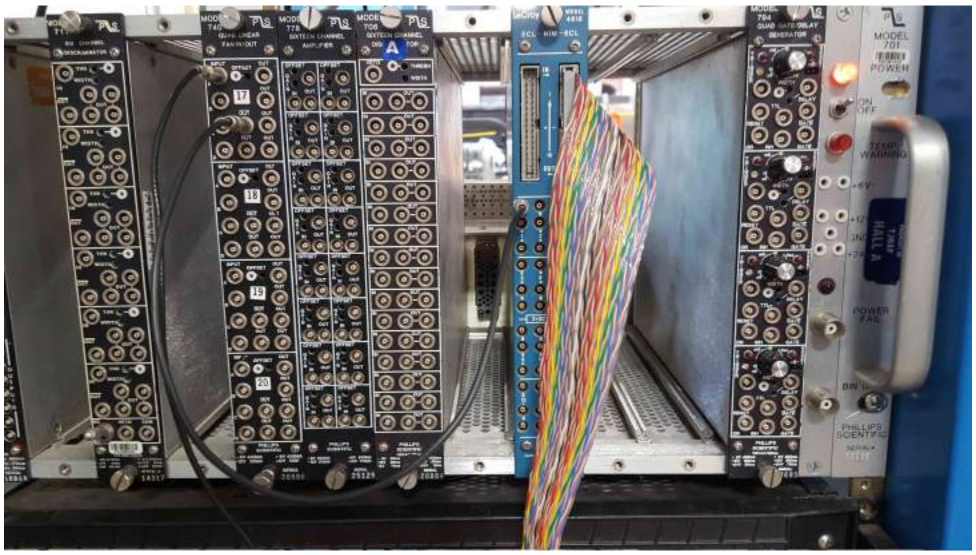


Figure 15. NIM Module Trigger Wiring. This figure shows the wiring needed to invert the trigger out from the pulser used to create triggers and a reference channel and then convert it to an ECL signal that is sent to the VETROC via a ribbon cable. The trigger out from the pulser is the top-left cable.