Using NCSL DAQ Software to Readout a CAEN V785 and V775

Timothy Hoagland

December 2, 2004

Abstract

The paper's purpose is to assist the reader in setting up and reading out data from CAEN V775 and CAEN V785 at the same time. This paper assumes that you are familiar with both of these modules and that you have setup and ran them both individually. Help with setting them up individually is available athttp://docs.nscl.msu.edu/daq/samples This paper will expand on the code written code in those documents for the CAENcard class of modules.

All the code is available at:
http://docs/daq/samples/CAEN V775/CAEN V775.zip

Contents

List of Figures

1 A Brief Description of the CAEN V775 and V785

The CAEN V775 is an 12-bit time digitizer. The output is a value proportional to the time between two logic pulses. It has 32 channels on a one unit wide VME module. The board can be operated in either common start or common stop mode. The CAEN V785 is a 12 bit, 32-channel analog to digital converter that returns a value related to the maximum voltage that occurs during the time a gate is present. The input signal must be positive and less than 4V. Both are VME based modules. For a complete understanding of the modules I suggest that you obtain a copy of the product manuals, available as a PDF at http://www.caen.it/

2 A Minimal Electronics Setup

The following items will be needed to build our simple setup:

- CAEN V775
- CAEN V785
- VME Crate
- VME controller
- NIM Crate
- NIM Discriminator
- NIM Amplifier
- NIM Splitter
- Lemo to Ribbon cable converter
- NIM Gate and Delay generator(3 channels)
- NIM-ECL Converter
- VME CAEN V262 - I/O controller

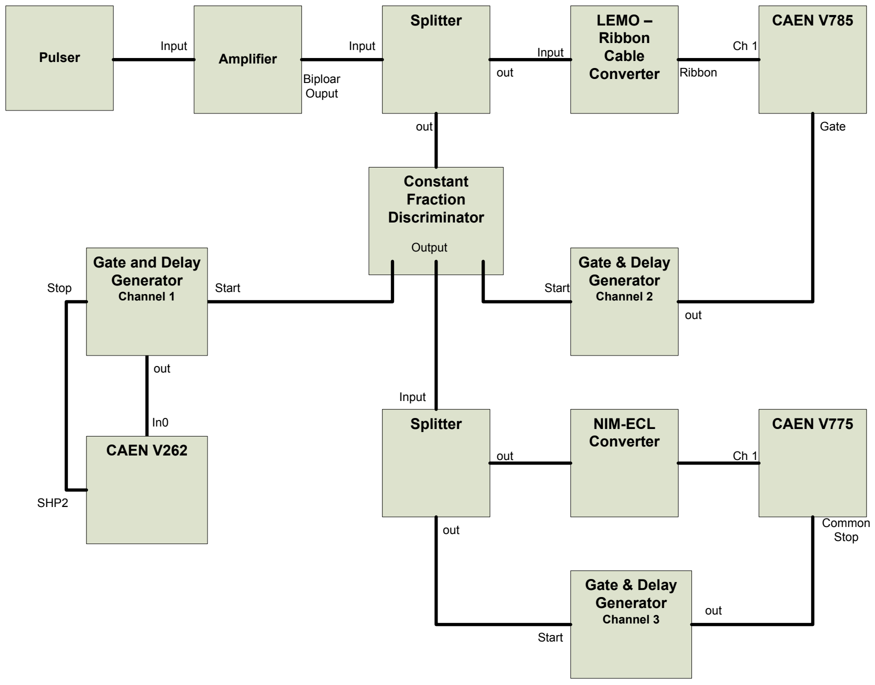

Figure 1: A simple electronics setup to readout a CAEN V775

- Pulser
- 50 Ohm LEMO terminator
- Various length LEMO cables
- Oscilloscope

Before starting make sure that you can secure all of these items, many are available through the NSCL electronics pool. Figure 1 shows the complete setup of the system. More information about some of the modules we are going to use is available at http://docs/daq/samples/

We will first look at the signal from the pulser. A pocket pulser will work well for this. The signal directly from the pulser will look like figure 2 when viewed on a scope. The pulser only works when terminated with 50 Ohm.

The pulser will serve as the input to the NIM based amplifier. The amplifier will be used to give us the positive signal that is required by the

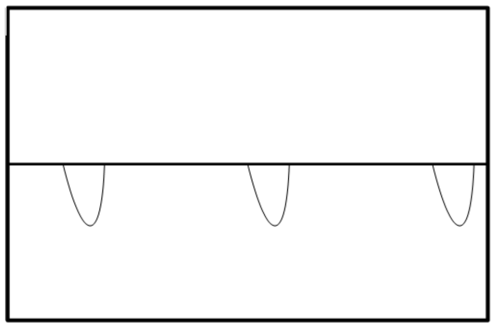

Figure 2: Pulser output signal

V785. It will also improve the signal to noise ratio making our peak stand out more later. The bipolar output of the amplifier will go into a splitter.

A splitter does just what its name suggests and splits the signal, however unlike a "T" it also maintains the 50 Ohm termination we need for the pulser. One of the outputs from the splitter should be connected to Ch0 of the V785 via the LEMO to Ribbon Cable Converter. The other output will go to the input of the discriminator.

NOTE: Ribbon cable can difficult to work with because it is easy to get it twisted and lose track, of which end is which. To avoid this look carefully at the coloring on the cable you are using and be sure it is plugged in the correct channel of the TDC

Discriminators are used to determine when a signal has occurred. It does this by monitoring the input voltage and emits a logic pulse when the signal reaches a user-defined threshold. Obviously you should set the threshold above the noise level in the input signal to prevent the discriminator from identifying noise as a legitimate signal. You will need four outputs from the discriminator. If the one you are using does not have four outputs you can split one of them using a splitter, figure 1 show the setup using a splitter since most discriminators only have three outputs.

One of the discriminator outputs will be used to trigger a gate for the V785. To produce the gate connect the discriminator output to the start of a gate and delay generator. Adjust the output of the of the gate and delay generator until the so that the output signal produces a temporal window around the signal going into Ch0 of the ADC. Connect the gate to the gate input of the V785 and terminate the other connector with 50 Ohm

One of the discriminator outputs will provide the start signal for the V775. This should be connected to Ch0 of the V775 through the NIM-ECL converter. Use another discriminator output to provide the stop. The stop signal will have to be delayed this can be done using cable delay or a delay generator. The delay should be about 400ns. When you have set the delay connect it to one of the common connectors on the V775, terminate the other with 50 Ohms.

The last output of the discriminator will go to a gate and delay channel; this combined with the CAEN V262 will trigger the computer when there is data present. This channel will be run in latched mode meaning that it will be given both a start and a stop signal for the gate. The start signal is the signal from the discriminator. The discriminator output will go to the IN0 of the CAEN V262 I/O module. A cable running from the SHP2 output of the V262 to the stop of the gate and delay generator will deliver the computer generated stop.

At this point your setup should be complete. Now is good time to make sure that your setup is the same as Figure 1. If everything is setup correctly the BUSY and DRDY lights on the V775 should be lit up.

NOTE: The setup shown in figure 1 does not have a dead time lockout, that is it could try to process a second event while the computer is still busy. This will not be a problem as long as source is a predictable as a pulser but would be problematic if we replaced the pulser with a detector signal.

3 Sofware Modifications

Since this paper is written with assumption that you have already written or obtained code to readout a CAENcard objects the discussion here will be limited to the modifications that will need to be made to that code

3.1 Readout Modification

Only one change needs to be made to the MyEventSegment class. There are some card dependent initialization details we need to take care of for the

TDC. Therefore modify the Initialize function to look like:

```
void MyEventSegment::Initialize()
{
  module->reset();  //reset defaults
  system("sleep .1s"); //pause while reset happens
  module->clearData(); //clear data buffer
  if(cardType()==775)
  {
    module->discardOverflowData();
    module->discardUnderThresholdData();
    module->commonStop();  //common stop mode
    module->setRange(0x1E); // Set full range
  }
}
```

There is one small modification that must be done to the Skeleton.cpp. The previous version of the code only defined one module we now need to create the second. After opening Skeleton.cpp add locate the block of code that reads:

```
void
CMyExperiment::SetupReadout(Cexperiment& rExperiment)
{
   CReadoutMain::SetupReadout(rExperiment);

   // Insert your code below this comment.

   rExperiment.AddEventSegment(new MyEventSegment(8,12));
}
```

Replace that block of code with:

```
void
```

```
CMyExperiment::SetupReadout(Cexperiment& rExperiment)
{
   CReadoutMain::SetupReadout(rExperiment);

   // Insert your code below this comment.

   rExperiment.AddEventSegment(new MyEventSegment(8,12));
   rExperiment.AddEventSegment(new MyEventSegment(9,13));
}
```

The two numbers in the MyEventSegment call are the VME slot number and the packet ID, respectively. You will need to replace these numbers with values that reflect your own setup. Be sure to note what ID you use, that will be needed later.

We are now ready to compile our code, simply type ”make” at the command prompt. This will compile and link all the needed files and create a file called Readout. Once compilation errors have been fixed we are ready to test Readout.

3.2 Modifying SpecTcl

We are now ready to make some changes to the MySpecTclApp.cpp file. Locate the CreateAnalysisPipeline function. Edit this function to read:

```
MyEventProcessor MyProcessor1(101, 12);
MyEventProcessor MyProcessor2(101, 12);
void
CMySpecTclApp::CreateAnalysisPipeline(CAnalyzer& rAnalyzer)
{
  RegisterEventProcessor(MyProcessor1);
  RegisterEventProcessor(MyProcessor2);

}
```

The parameters passed after MyProcessor* are the base index and packet ID we used earlier. You can now run ”make” to compile your code to make a file called SpecTcl. Fix all compilation errors before continuing.

3.3 Modifying the SpecTcl Script

The setup.tcl script is the last thing that needs modified. We have to tell it that there are 32 more channels and where to find them. The finished script should look like

```
set slot 101;    #TDC Base index used earlier earlier
#Define 32 parameters named tdc0...tdc31 starting in slot 101
for {set i 0} {$i <= 31} {incr i} {
   parameter tdc$i $slot  12;
   incr slot
}
set slot 201;    #ADC Base index used earlier earlier
#Define 32 parameters named adc0...adc31 starting in slot 101
for {set i 0} {$i <= 31} {incr i} {
   parameter adc$i $slot  12;
   incr slot

#define a 1-d spectrum for each parameter:
for {set i 0} {$i <= 31}  {incr i} {
    spectrum tdc$i 1 tdc$i {{0 4095 2048}; # 12 bit
    spectrum adc$i 1 adc$i {{0 4095 2048}; # 12 bit

}

sbind -all;   #make all spectrum displayable
```

3.4 Running SpecTcl

At this point SpecTcl and Readout should both be ready to run. Start Readout and SpecTcl just like you would is you using only one module. To obtain a spectrum of the ADC and TDC channels your using at the same time Click on the Geometry button and choose 1 row and 2 columns. Then click in each frame individually and choose which spectrum display in that frame.

4 More information

The DAQ system we just put together is a fully functional and useful system. A system like this could be used in real experiments by simply replacing the pulser with a couple of detectors. An ideal use of this system would be to obtain time of flight and energy information from a particle. This system could easily be expanded and combined with other modules such as the CAEN V792 QDC.

More information is available at: **http://docs.nscl.msu.edu/** and should be your first source for help. If that doesn't help contact **daqdocs@nscl.msu.edu**

Please help with the accuracy of the paper. If you find any kind of error please report it at **daqdocs@nscl.msu.edu** .

5 Complete Sample Code

The complete code used in this paper can be found at http://docs.nscl.msu.edu/daq/samples/. Be sure to specify your own slot numbers and Ids. Also you will have to modify ”abutton” so that it references the computer that you are using. Finally don’t forget that you need to compile the code after any changes.