NSCLDAQ Support for CAEN DPP-PSD digitizers.

ron Fox

2020

Contents

[Introduction 2](#_Toc28798642)

[Building the software 2](#_Toc28798643)

[Creating a Tailored Readout 3](#_Toc28798644)

[Skeleton.cpp 4](#_Toc28798645)

[Using DPP-PSD and DPP-PHA together. 4](#_Toc28798646)

[Event Format 5](#_Toc28798647)

[APPENDIX I – The New, Improved regdump program. 5](#_Toc28798648)

NSCLDAQ Support for CAEN DPP-PSD digitizers.

# Introduction

CAEN DPP-PSD firmware allows selected CAEN digitizers to perform digital signal processing that performs charge integration for a short and long gate. Baseline determination prevents this integration from being biased by a DC offset. Triggers can be based on a digital Leading Edge Discriminator (LED), or Constant Fraction Discriminator. In the event the constant fraction is used, it is possible to get sub-sample signal timing by interpolating the zero crossing of the CFD.

The remainder of this manual describes:

* How to build the libraries that make up the software.
* How to create tailored Readout programs that make use of this library. This section will also describe how to integrate this software with other modules such as CAEN DPP-PHA modules.
* The format of the data in events produced by this software.

# Building the software

The software depends on the following libraries:

* The CAENDigitizer library provides the top level support for the CAEN digitizers.
* CAENComm library provides connection independent methods used by the CAENDigitizer library to communicate with the digitizers.
* CAENVMELib provides VME adaptor specific methods used by CAENComm to communicate with digitizers hosted by VME adaptors such as the CAEN V1718
* The Pugi-XML non validating XML parser, used to parse the XML files crated by the CAEN CoMPASS software.

Note that suitable versions of these libraries are provided with the software. If you use this software in conjunction with the DPP-PHA support software, both libraries must be compiled and linked to the same versions of the CAEN libraries above.

In addition to these libraries:

* The system running this software must have appropriate device drivers present and loaded to communicate with the interfaces to the digitizers.
* You should use the CAEN CoMPASS data acquisition system to configure the digitizers. The XML configuration file produced by that software will be used by the NSCL Support libraries to configure the digitiers.

The top level directory file has a Makefile that must be edited to allow the software to compile properly:

The Makefile symbol CAENCXXFLAGS must be edited to provide the correct header directories for the three CAEN libraries above and the version of NSCLDAQ you will be linking this software to.

Since we are creating static libraries, at this time the CAENCXXLDFLAGS symbol does not need to be edited.

A successful build of the library software will result in:

* libpugi.a - the Pugi-XML XML Parser.
* libCaenPsd.a – the CAEN DPP-PSD support library. This consists of the following modules:
  + PSDParameters – the DPP-PSD digitizer parameter parser and resulting structures. This software does not normally need to be referenced directly by user code.
  + CDPpPsdEventSegment – A class that creates an event segment from a single DPP-PSD digitizer.
  + CPsdCompoundEventSegment – A class that can create an event segment from several DPP-PSD digitizers that may have a synchronized start.
  + CPsdTrigger – an event trigger for a single DPP-PSD module.
  + CCompoundTrigger – A trigger than can poll several arbitrary triggers and return the OR of them all. If you use the DPP-PSD support in conjucnction with the DPP-PHA support, you can insert the triggers for both of those modules into this trigger to obtain triggers from both types of modules.
  + COneOnlyEventSegment – An event segment that contains an arbitrary set of other event segments but only allows one to be read out. The attempts to read are ‘fair’.

# Creating a Tailored Readout

The rodtest directory contains a sample tailored Readout program that runs a single DPP-PSD module. IT also contains a Makefile that, with appropriate editing will build the Readout program and an improved version of regdump, described in the appendix of this document.

The Makefile symbols that must be edited:

* INSTDIR must be edited to point to the top level directory of the version of NSCLDAQ you’re using. Note that while we developed with a relatively old version of NSCLDAQ (11.2-005), any version newer than that *should* work.
* CAENCXXFLAGS – as for the library builds must be edited to provide the proper header directory -I directives for the CAEN support libraries.
* USERCCFLAGS may require editing to allow the directory the support software is installed in to be searched.
* CAENLDFLAGS must be edited to link in the correct set of libraries.
* USERLDFLAGS must be edited to link in libCaenPsd and libpugi. Note that if you are also using the DPP-PHA software you must add those libraries to the build as well.

## Skeleton.cpp

The tailoring to support the DPP-PSD software is confined to Skeleton.cpp, specifically, Skeleton::SetupReadout.

For each digitizer module you must construct a CDPpPsdEventSegment. The constructor for this class requires the following parameters:

* The link type (See PSDParameters.h for legal linktypes).
* The link id
* The link node
* The Digitizer base address if the digitizer is run through a VME controller.
* The event builder source id to be associated with events from this digitizer.
* The path to the CoMPASS configuration file. Note that the constructor will locate the section of the configuration file that corresponds to the board described by the first four connection parameters so there’s no need to separate a multiboard CoMPASS configuration file into per-board pieces.

You must create one CPsdCompoundEventSegment and use AddModule to add each CDPpPsdEventSegment you created to it. This will handle synchronized starts for the module set. More about this later. This CPsdCompoundEventSegment should be added to the experiment’s event segment.

You should also create a CPsdTrigger instance and use its addModule method to add each CDPpPsdEventegment module to that trigger. Then use the CExperiment’s EstablishTrigger method to set this as the trigger forevents.

## Using DPP-PSD and DPP-PHA together.

In addition to the obvious Makefile modifications to ensure you link both support libraries, you need to take a bit of care about how to setup the event segment. Presumably all digitizer modules will run with a common synchronization. That means either one of the DPP-PSD or DPP-PHA modules is the master. CoMPASS sets the master digitizer to be software started. Synchronized starts means that:

* First all slave digitizers are setup and armed. Slave digitizers are programmed to use a front panel input to actually start as well as a clock daisy chain to provide a common clock to all modules.
* Finaly the master digitizer is given a software start and that will initiate the hardware starts in the slaves.

This means that;

The DPP-PSD and DPP-PHA compound event segments (CDPpPsdCompoundEventSegment and CompassMultiModuleEventSegment (DPP-PHA) ask each digitizer module if it is a slave or a master. The event segment that does not have a master will therefore initialize all modules as slaves. Furthermore, we want to ensure that each trigger will only give one hit as the event. The class COneOnlyEventSegment is an event segment that does this (part of the DPP-PSD library). Add your CDPpPsdCompoundEventSegment and CompassMultiModuleEventSegment to an instance of a COneOnlyEventSegment, with the segment containing the master last. Register the resulting COneOnlyEventSegment with the experiment as an event segment:

* Since you registered the segment containing the master module last, it will get initialized last as it should be, starting/synchronizing all slave modules.
* The COneOnlyEventSegment responds to triggers by “fairly” cycling through its member segments reading them. Once any of those segments returns data it returns without attempting to read any more member segments.
* “fairly cycling” means that it keeps track of the last event segment that returned data and starts trying to read with the “next” segment the next trigger. This should prevent readout starvation in much the same way that readout starvation is prevented by the DPP-PSD and DPP-PHA compound event segments.

The triggers from both the DPP-PHA and DPP-PSD modules should be added to a single CompoundTrigger. Note that fairness is not required when polling for triggers. The key thing to know is that at least one module has data and then to let the fair cycling readout take it from there.

# Event Format

The data for an event from the CDPpPsdEventSegment has the following format:

|  |  |
| --- | --- |
| **Contents** | **Size** |
| Number of 16 bit words in the event | uint32\_t |
| Size of the remainder of the event in bytes | uint32\_t |
| Raw digitizer timestamp(\*) | uint64\_t |
| Channel number | uint16\_t |
| short gate charge value | uint32\_t |
| Long gate charge value | uint32\_t |
| Baseline value | unt16\_t |
| Pile-up rejection indicator | uint16\_t |
| Wave form data length(\*\*) | uint32\_t |
| Wave form data – see below. | Varies. |

(\*) While the meaning of the raw digitizer timestamp is model dependent, the timestamp assigned to the event header is calibrated in nanoseconds so that mixed modules can be used.

(\*\*) This is self-inclusive so if traces are not being taken, this will have the value sizeof(uint32\_t).

If waveforms are acquired, the data after the wave form data length will look like this:

|  |  |
| --- | --- |
| **Contents** | **Size** |
| Number of samples/trace | uint32\_t |
| Non zero if dual trace mode is enabled. | uint8\_t |
| If dual trace is enabled, identifies the analog probe contained in the second trace | uint32\_t |
| Trace 1 | nsamples\*uint16\_t |
| Trace 2 (if dual trace enabled). | nsamples\*uint16\_t |

# APPENDIX I – The New, Improved regdump program.

One of the most useful trouble-shooting programs that helps understand differences between what CoMPASS is doing and what Readout does – when there are differences is a register dump program. The DPP-PHA register dumper had a hardcoded set of registers configured. The DPP-PSD software ships with an improved regdump which takes the description of the registers to dump from a file: registers.txt in the current working directory. Thus the set of registers dumped can be modified without recompiling the program. A registers.txt file that describes the DPP-PSD registers is also supplied. This is a text file with one definition per line Each line has three fields:

* The name of the register (no whitespace allowed here, sory
* The base address of the register(s) to dump
* A single register flag. This is 0 to indicate the address provided is really of the form xnyz where n is a channel number. 1 indicates only the supplied address is dumped.

The command parameters are the same for the original regdump – a description of the connection parameters to the digitizer.