Table of Contents

SWGuideL1TRawToDigi

This page describes the SWGuideL1TRawToDigi software package, which includes a framework for unpacking data readout by MP7 boards as well as customisations for the stage 1 calo trigger upgrade, and stage 2 calo and GT upgrades.

Code

The code is all contained in EventFilter/L1TRawToDigi. For stage 1 calo, use a recent CMSSW version and use cms-addpkg, ie :

```
cmsrel CMSSW_7_6_0_pre7
cd CMSSW_7_6_0_pre6/src
cmsenv
git cms-addpkg EventFilter/L1TRawToDigi
scram b -j 4
```

For stage 2 calo trigger, the code is under development on branch l1t-stage2calo-CMSSW_7_6_0_pre7. Check it out using :

```
cmsrel CMSSW_7_6_0_pre7
cd CMSSW_7_6_0_pre7/src
cmsenv
git cms-merge-topic --unsafe cms-l1t-offline:l1t-calostage2-$CMSSW_VERSION
git cms-checkdeps -a
scram b -j 4
```

To compile the unpacker with debug messages (which you probably want if you are debugging problems) do :

```
cd EventFilter/L1TRawToDigi
scram b clean
scram b -j 4 USER_CXXFLAGS="-DEDM_ML_DEBUG"
```

Instructions for developers are available here

Utilities

These are located in EventFilter/L1TRawToDigi/utils. Usage examples provided below.

Stage 1 : (mini)DAQ data

Utility to unpack data stored in streamer file or EDM file format. It will dump both raw data and digis, as well as produce histograms and output an EDM file including unpacked objects.

```
cmsRun EventFilter/L1TRawToDigi/utils/unpackData-CaloStage1.py \
inputFiles=file:run247425_ls0021_streamA_StorageManager.dat \  put multiple files in a comma sepa
streamer=True \
skipEvents=10 \
maxEvents=10 \
valEvents=False \
debug=True \
dump=True \
dumpDigis=True \
histos=True \
edm=True
```

Note that the 'debug' option, which enables debug messages, requires the symbol EDM_ML_DEBUG to be defined at compilation time, ie.

```
cd EventFilter/L1TRawToDigi
scram b clean
scram b -j4 USER_CXXFLAGS="-DEDM_ML_DEBUG"
```

TDR Upgrade : (mini)DAQ data

As above, but covering the stage 2 unpackers (calo layer 2, demux, uGT).

```
cmsRun EventFilter/L1TRawToDigi/utils/unpackData-CaloStage2.py \
inputFiles=file:run247425_ls0021_streamA_StorageManager.dat \  put multiple files in a comma sepa
streamer=True \
skipEvents=10 \
maxEvents=10 \
valEvents=False \
debug=True \
dumpRaw=True \
dumpDigis=True \
histos=True \
edm=True
```

TDR Upgrade : Buffer Capture (OBSOLETE!)

To run unpacking from MP7 buffer capture use :

```
cmsRun EventFilter/L1TRawToDigi/test/unpackBuffers-CaloStage2.py
```

General command line options :

```
skipEvents=N       - skip N events (default = 0)
maxEvents=N        - run over N events (default = 1)
dump=True          - dump DAQ raw data payload (default = False)
debug=True        - enable debug printout (default = False), see note about compiling with debug
```

MP unpacking options :

```
doMP=False         - enable/disable MP crate (default = True)
mpFramesPerEvent=N - number of buffer dump frames per MP event (default = 40)
mpLatency=N        - latency in frames of MP (default = 0)
mpOffset=N         - offset in packets of MP (default = 0)
```

Demux unpacking options :

```
doDemux=False      - enable/disableDemux (default = True)
dmLatency=N        - latency in frames of Demux (default = 0)
dmOffset=N         - offset from zero, in frames, of start of Demux data (default = 0)
```

GT unpacking options :

```
doGT=False      - enable/disable GT (default = True)
gtFramesPerEvent=N - number of buffer dump frames per GT event (default = 6)
gtLatency=N        - latency in frames of GT (default = 47)
gtOffset=N         - offset in packets of GT (default = 0)
```

The job reads input files :

```
mp_rx_summary.txt - input data from all MPs concatenated
mp_tx_summary.txt - output data from all MPs concatenated
```

```
demux_rx_summary.txt - demux input data
demux_tx_summary.txt - demux output data
```

Developing New Unpacker Code

The framework is designed to unpack RAW data produced by a crate of MP7 cards, using an AMC13 to send data to DAQ. The raw data from a single MP7 is arranged in "Blocks", each of which corresponds to data captured from one input or output link (or a dummy output link that is only used for data acquisition). The framework allows each Block to be assigned to an Unpacker class, which will convert the Block contents to a C++ object. The user has control over how many Unpacker classes are required and which Blocks should be assigned to which Unpackers.

A new MP7 user needs to provide the following classes, in a directory under /src (with a name to be agreed with Matthias Wolf!)

- At least one UserUnpacker class. This should inherit from the Unpacker class, and override the `unpack()` method. This method is called whenever a Block that the class is expected to unpacked is encountered, with the block contents as one argument, and a pointer to a UserCollections object as the second, into which it should insert the results of unpacking. Example:

```
namespace l1t {
  namespace stage1 {
    class UserUnpacker : public Unpacker {
      public:
        virtual bool unpack(const Block& block, UnpackerCollections *coll) override {
          SomethingBxCollection res = static_cast<UserCollections*>(coll)->getSomething();

          int nwords; // every link transmits 2 words per event
          int nBX, firstBX, lastBX;
          nBX = int(ceil(block.header().getSize() / nwords));
          getBXRange(nBX, firstBX, lastBX);

          res->SetBXRange(firstBX, lastBX);

          // now process block.payload() and put result into the collection res
        };
    };
  }
}
```

- A UserSetup class. This inherits from `PackingSetup` and overrides the following methods:
  - `registerProducts()`, which tells the CMSSW which output collections to expect.
  - `getCollections()`, which returns a subclass of `UnpackerCollections` (is called once per event.)
  - `getUnpackers()`, which returns an `UnpackerMap`, to instruct the framework which unpacker to use for each block of the payload. The map contains link IDs as keys and unpacker objects as values.

- A UserCollections class which inherits from `UnpackerCollections`. This is merely a container for the unpacking products (or collections). To save the output collections, use `event_.put(myCollection_)` in the destructor of the subclass for every collection.

-- JimBrooke - 2015-06-08

This topic: CMSPublic > SWGuideL1TRawToDigi
Topic revision: r16 - 2015-10-27 - JimBrooke