CDMSNOTE September 20, 2012

CDMS EVENT FORMAT FOR SOUDAN iZIP and Hybrid iZIP/Mercedes Runs

CDMS DAQ group (in consultation with maintainers of CDMSBATS)

1 Introduction

This note reviews the Super CDMS - Soudan data format. This is the second revision of the data format from the CDMS-II format. The first revision changed the file names, the facility codes, and added a new administrative record (x0000 0002) with a longer series number capability (For the intermediate runs, 130/131, see the SoudanST event format document). This document also includes a new trace record (x0000 0011). The only difference between this and the older trace record is in the details of the detector code.

For the initial iZIP datasets, we will be taking data in a highly mixed mode. For example, run 132, will consist of a mercedes supertower, an old CDMSII tower, and an iZIP tower, which will have two different styles of iZIPs. This document focuses primarily on the data formats that are relevant to the iZIP datasets (R132 and beyond). Much information in this note overlaps previous data format documents. For details specific to the data format from the original 5-tower run at Soudan, please see the document "CDMS Event Format for SUF and Soudan".

2 Design

A DAQ event is viewed as a self-contained unit of information from any trigger source, for example a background-running trigger event, or an experimental-monitoring event. In the background-running event, we would have data from the detectors, veto shields, etc., and from an experimental-monitoring event, we would have data from scalers recording event rates, fridge temperature accessed via the ethernet, etc.

A data file then contains a series (about 1000) of DAQ events. The basic data unit of the DAQ event is the computer word consisting of 4 bytes or 32 bits of binary data. The DAQ event buffer and subevent buffers (also called logical records) were designed to be a linked list of data structures. Specifically, each event or subevent buffer have two header words, the first one identifying the event/subevent type, and the second the size in bytes of the ensuing data. Figure 1 shows the structure of this linked list. The first word in a DAQ event contains the unique header 0xa980 as well as additional information about the event (see next section for complete details), and the second word gives the number of bytes for that event and points to the next DAQ event header word. In each event buffer is contained a linked list of subevent (or logical record) buffers. Each logical record unit, like the event unit, has an identifying header word and a word for the size (again in bytes) of the logical record data buffer.

3 File Header

At SUF, data files were written to disks using Macs whose machine architecture is big-endian so that for multibyte data, the most significant byte goes first. For example, the 4-byte integer 0x01020304 will be stored on file as byte 1 = 0x01, ..., byte 4 = 0x04. The Soudan DAQ uses PCs having Intel CPUs which are little-endian, or least significant byte goes first. Thus the example integer 0x01020304 above will be stored on file as byte 1 = 0x04, ..., byte 4 = 0x01. In light of this possible confusion, we have added an endianness word at the beginning of every data file for Soudan. Furthermore, we will use one more file header word to describe the DAQ/data-format version for that file. The two file header words are then

```
word 1 -- endianness descriptor (= 0x01020304)
word 2 -- DAQ/data-format version descriptor
```

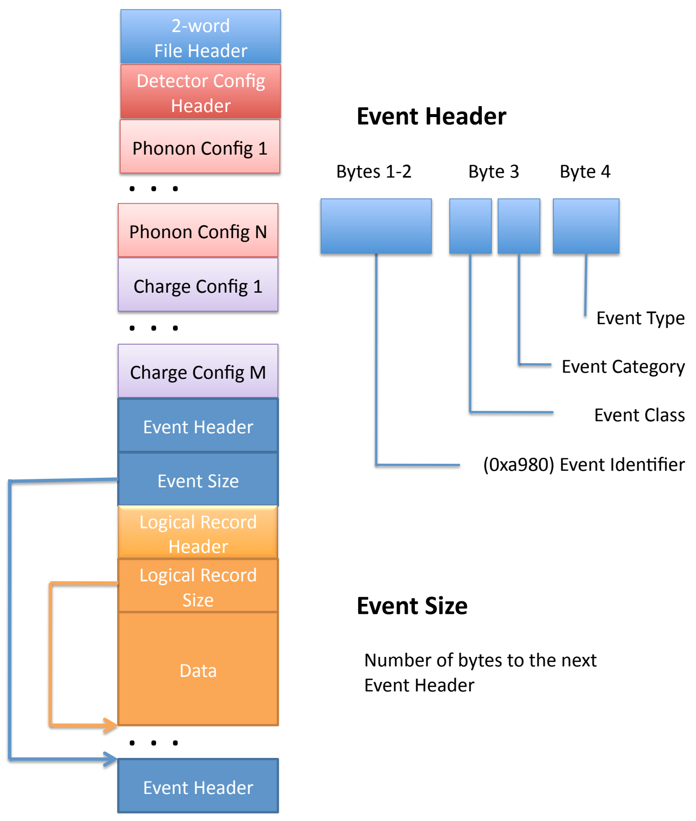

Figure 1: The linked list data structure of a CDMS raw data file.

The four bytes of word 2 is formatted as

```
word 2 = DAQ major | DAQ minor | data-format major | data-format minor
```

where "DAQ major" means the major field in the version number of the daq. etc. This current Soudan data version is 2.0 so that data-format major = 2, and data-format minor = 0. For obsolete reasons SUF data files contained a 3-word header, first 2 identically 0, and the third was the online software version. We will supersede that with the above described 2-word file header.

4 Detector Configuration

Starting Fall 2010, we expect the DAQ to start incorporating more ZIP detector configuration data directly into the raw data. At the beginning of each file, after the file header, will be a detector configuration which specifies the settings on each channel of each detector. The storage of gains and biases in this file will take the place of the information that was formerly obtained from the ISR file.

Note that some of these values can be signed, so unlike the rest of the raw data file, signed integers must be specified for reading back the values from the detector configuration record. This is ok for the potentally largest number, the detector code, because its maximum value of 999999999 can still fit within a signed 32-bit integer. The ×100 factors are needed for storing floating values as integers. They may be converted back to more familiar units once they are read from the file.

```
Detector config record format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                  Config Header = 0x0001 0000
   2                  Configuration record length (len)
   3                     Phonon channel config header = 0x0001 0001
   4                     Phonon channel config record length (len = 11*4 = 44 bytes)
   5                        Detector code (see trace record) (signed 32 bit)
   6                        Tower number (signed 32 bit)
   7                        Phonon post-amp (driver) gain * 100 (signed 32 bit)
   8                        Phonon QET bias * 100 [pA] (signed 32 bit)
   9                        Phonon SQUID bias * 100 [pA] (signed 32 bit)
   10                       Phonon SQUID lockpoint * 100 [uV] (signed 32 bit)
   11                       Phonon RTF offset  [uV] (signed 32 bit)
   12                       Phonon variable gain (tunes SQUID bandwidth) (signed 32 bit)
   13                       delta t [ns] (signed 32 bit)
   14                       T0 (time of trigger in trace) [ns] (signed 32 bit)
   15                       Trace length [samples] (signed 32 bit)
   ....
   N+1                   Charge channel config header = 0x0001 0002
   N+2                   Charge channel config record length (len = 8*4 = 32 bytes)
   N+3                      Detector code (see trace record) (signed 32 bit)
   N+4                      Tower number (signed 32 bit)
   N+5                      Charge channel post-amp (driver) gain * 100 (signed 32 bit)
   N+6                      Charge channel bias [uV] (signed 32 bit)
   N+7                      Charge RTF offset [uV] (signed 32 bit)
   N+8                      delta t [ns] (signed 32 bit)
   N+9                      t0 (time of trigger in trace) [ns] (signed 32 bit)
   N+10                     Trace length [samples] (signed 32 bit)
```

5 Event Header Word

The first 2 bytes of the Event Header word is assigned the unique event identifier 0xa980, see Figure 1. The next byte allocates 4 bits for the Event Class and 4 bits for the Event Category. The least significant byte describes the Event Type.

5.1 Event Class (4 bits)

```
0x0   - Raw
0x1   - Processed
0x2   - Monte Carlo
```

For SUF, only the raw data event classes were implemented. During the Soudan period, the DMC is now coming online. We will also make use of the 0x2 flag in this field. Monte Carlo files will also be further identified because they begin with a “5” in the location number.

5.2 Event Category (4 bits)

For raw events the following categories are possible:

```
0x0   - Per Trigger
0x1   - Occasional
0x2   - Begin File Series
0x3   - Begin File
0x4   - End File
0x5   - End File Series
0x6   - Per Trigger w/selective readout of detectors that cross threshold
```

For SUF, only the ”per trigger” category had been used. It was thought that the ”Begin File” and ”End File” categories may be implemented but were not. This is the same situation at Soudan so that this field will be identically 0. Note that the Per Trigger above is meant to be due to real physical triggers such as ZIP triggers, and Veto Multiplicity triggers via the Spare Trigger channel in the TLB (Trigger Logic Board) as was configured in Soudan run 119. As a software crosscheck, we use the Occasional tag for DAQ generated triggers (via the Random trigger channel in the TLB).

5.3 Event Type (8 bits)

For raw events, the following types are possible:

```
0x0  - WIMP search
0x1  - 60Co calibration
0x2  - 60Co low energy calibration
0x3  - Neutron calibration
0x4  - Random triggers
0x5  - Pulser triggers
0x6  - Test
0x7  - Data monitoring event
0x8  - 137Cs calibration
0x9  - 133Ba calibration
0xa  - Veto OR multiplicity trigger
```

Be advised that the last type (0x7) for data monitoring was added for Soudan. We have decided to write the same auxiliary files (such as .info, .isr, .dmm, .hv, etc.) as were used at SUF. Thus for Soudan running we will log the asynchronous data-monitoring DAQ events in parallel with the main synchronous detector-trigger DAQ events.

Comment: in CDMS note 0004003, the above was followed by a series of types for these raw occasional (or monitoring) events. It makes more sense to reassign those to the logical records, and we have done so, see later in this note.

Comment 2: in CDMS note 0004003, there were also some comments here about begin and end file events, which were never implemented. To avoid extraneous confusing information, we have not included those comments here.

6 Logical Records – All Event Types Except Data Monitoring

In every logical record, the first word gives a unique code that describes the type of data in the buffer. The possible logical record types are

```
0x0000 0002    - Administrative, long series number (Event Number, Timestamp, Livetime,...)
0x0000 0011    - Trace, long detector code (replaces older 0x0000 0010 type)
```

```
  0x0000 0021    - Soudan History Buffer
  0x0000 0060    - GPS Date/Time
  0x0000 0080    - Trigger
  0x0000 0081    - TLB Trigger Mask
```

All these types were used at SUF with the exceptions of the Soudan History Buffer type and the TLB Trigger Mask. See below for the Soudan extension for which we have added the distinct type code 0x0000 0021. See also the added TLB Trigger Mask format which will be implemented for the Soudan 5-tower data run of 2005.

The Adminstrative record contains general information about the event. Each data file will be uniquely identified using a Series Number which has date and time information for the first event in that file as well as relative times of events within that same series.

```
Adminstrative record format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                Admin Header = 0x0000 0002
   2                Admin length (len = 6*4 = 24 bytes)
   3                   Series Number LLYYMMDD (U32)
   4                   Series Number HHMM (U32)
   5                      Event Number Within This Series (U32)
   6                      Event Time (seconds from Epoch) (U32)
   7                      Time From Last Event (milliseconds) (U32)
   8                      Live Time Since Last Event (milliseconds) (U32)
```

The Series Number is in the form:

```
LLYYMMDD_HHMM
```

where each YY, MM, ... represent two decimal digits. The digits LL are formed by

```
    LL =  0 (SUF)
          1 (Soudan)
          2 (UCB)
          3 (CWRU)
          6 (Queens)
          7 (U of Minn)
```

So for example, a series started at 4:30 pm on January 15, 2010 at Soudan would have Series Number:

```
01100115_1630.
```

The Event Time at SUF was referenced from 12 am, Jan 1, 1904, an artifact of using Macs. For Soudan, we will reference the Event Time from the Unix Epoch time of Jan 1, 1970. In addition, we have designated that Monte Carlo output utilize "5" as the first digit. For example, DMC output corresponding to Soudan data for example will have location code 51.

The Trace record format is as shown below, and the explanations follow:

```
Trace record format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                    trace Header = 0x0000 0011
```

```
2                     trace length (len)
3                        trace bookkeeping header = 0x0000 0011
4                        bookkeeping length (bk = 3*4 = 12 bytes)
5                           Digitizer base address (U32)
6                           Digitizer channel (U32)
7                           Detector code (U32)
8                        timebase header = 0x0000 0012
9                        timebase length (tb = 3*4 = 12 bytes)
10                          t0 in nanoseconds (I32)
11                          delta t in nanoseconds (U32)
12                          number of points (U32)
13                       trace header = 0x0000 0013
14                       number of samples (ns)
14+1                        trace data, samples 1 (lower U16) & 2 (upper U16)
14+2                        trace data, samples 3 & 4
...                                ...
14+ns/2                     trace data, samples (ns-1) & ns
```

The fact that we assign two trace samples to each word means we should always have an even number of samples. Furthermore, due to the manner in which a PSD is unfolded into an FFT in the data processing pipeline, the expectation is that the number of bins is a power of 2 (i.e. 1024, 2048, 4096, etc.).

Note also that because the trace data are stored in units of U16 within the same 32-bit word, care should be taken to extract the right adc samples, e.g. the lower 16 bits of word 15 store ADC values for sample 1; the upper 16 bits for sample 2. More specifically, suppose adc1=0x0102, and adc2=0x0304, then word 15 would be 0x0304 0102. The PC's used for the DAQ system at Soudan have intel architecture, so small-endian. Thus, the raw data of the above word 15 would be stored in the byte sequence 0x02 0x01 0x04 0x03.

The digitizer base address is simply the VME base address of the physical module. For SUF and Soudan, we have adopted the convention of uniquely correlating the VME base address of the Joerger digitizers (VTR812) to the corresponding data address as shown below:

```
------------------------------------------------------------
 Joerger digitizer base/data address convention
------------------------------------------------------------
 VME base address        VME memory address
     0xa000                0xf000 0000
     0xa100                0xf100 0000
     0xa200                0xf200 0000
       ...                      ...
```

The Struck digitizers (SIS3301) have only the VME base address specification (A32) with a fixed offset to get to the data address. At Soudan, we have chosen to start with Struck addresses 0xf000 0000, 0xf100 0000, etc. Note that when Joerger/Struck digitizers are used in the same crate, care must be taken to not overlap the VTR812 memory address space with the SIS3301 base address space. The digitizer channel is as the name implies: the channel into which the analog traces are connected (1-8 for both the VTR812 and SIS3301).

The detector code (32-bit word), when written in decimal, takes the form: xxxyyyzzz where xxx, yyy, zzz correspond to the detector type (from 0 - 999), the detector number (0-999) and the channel number (0-999). The values of each of these quantities can vary depending on the detector type as follows:

```
xxx = 1  BLIP
      2  FLIP
      3  Veto
```

```
       4  ZIP
       5  mercedes ZIP
       6  endcap (class I) detectors read out as 1 detector (as in Soudan R130/131)
       7  endcap (class II) detectors read out as 2 detectors (as in Soudan R132 and beyond)
      10  iZIP (class I)  - semicircular inner phonon
      11  iZIP (class II) - mercedes inner phonon


yyy = Detector #


zzz = 0 (all veto traces)

       1 BLIP QI
       2 BLIP QO
       3 BLIP PS1
       4 BLIP PS2

       0 FLIP/ZIP/mZIP/ENDCAP (classI) QI
       1 FLIP/ZIP/mZIP/ENDCAP (classI) QO
       2 FLIP/ZIP/mZIP/ENDCAP (classI) PA
       3 FLIP/ZIP/mZIP/ENDCAP (classI) PB
       4 FLIP/ZIP/mZIP/ENDCAP (classI) PC
       5 FLIP/ZIP/mZIP/ENDCAP (classI) PD

       0 ENDCAP (classII) Q
       1 ENDCAP (classII) PA
       2 ENDCAP (classII) PB

       0 iZIP (classI) QIS1
       1 iZIP (classI) QOS1
       2 iZIP (classI) PAS1 (A1)
       3 iZIP (classI) PBS2 (B1)
       4 iZIP (classI) PCS1 (C1)
       5 iZIP (classI) PDS1 (D1)
       6 iZIP (classI) QIS2
       7 iZIP (classI) QOS2
       8 iZIP (classI) PAS2 (A2)
       9 iZIP (classI) PBS1 (B2)
      10 iZIP (classI) PCS2 (C2)
      11 iZIP (classI) PDS2 (D2)

       0 iZIP (classII) QIS1
       1 iZIP (classII) QOS1
       2 iZIP (classII) PAS2 (A1)
       3 iZIP (classII) PBS1 (B1)
       4 iZIP (classII) PCS2 (C1)
       5 iZIP (classII) PDS1 (D1)
       6 iZIP (classII) QIS2
       7 iZIP (classII) QOS2
       8 iZIP (classII) PAS1 (A2)
```

```
 9 iZIP (classII) PBS2 (B2)
10 iZIP (classII) PCS1 (C2)
11 iZIP (classII) PDS2 (D2)
```

For example, a trace from charge channel QI2 (channel 6) on an iZIP of type class II (detector type 11), which is assigned to be detector number 17 will be stored in the raw data file with detector code: 0x00a8 1b2e (in decimal it is 11017006).

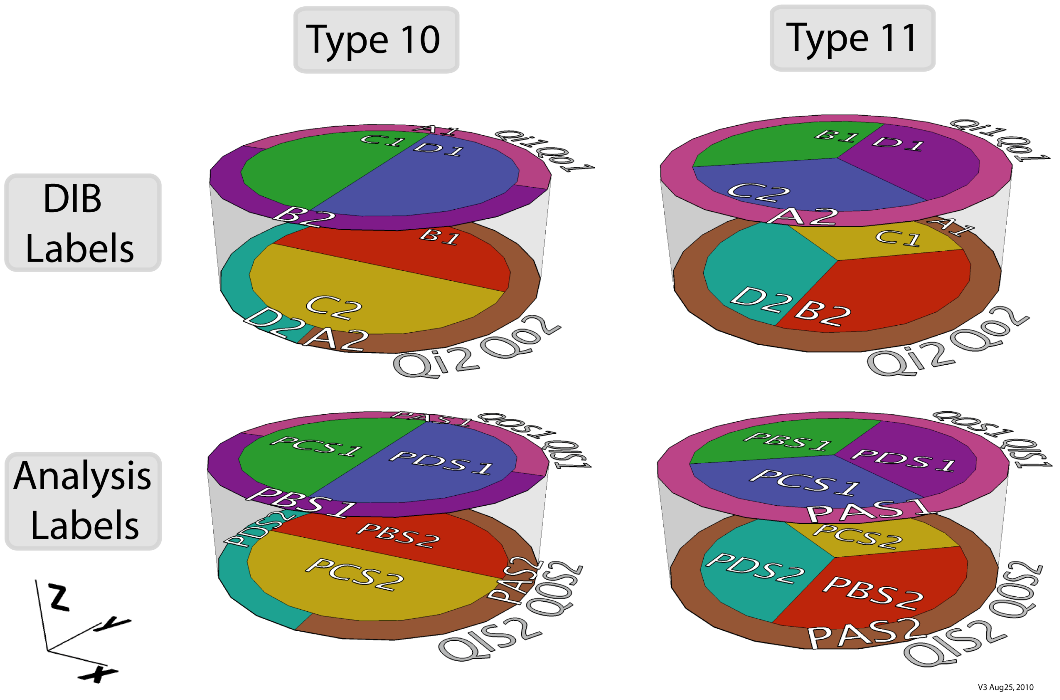

Figure 2: dib and analysis mapping for the first (left) and second (right) generation of iZIP installed at Soudan. At the time of this document update, it is unclear whether we will have a mixed run of class I and II or just iZIPs of classII.

Note that for the iZIP channel names, the physical channel number which corresponds to the cold hardware (the name in parenthesis) does not correspond in an intuitive way with the physical location of the channel on the detector. Specifically, channels on one side of the detector are not necessarily hooked up to the same stripline. This is a constraint related to where the wire bonds must be placed and the fact that we must use two striplines (designed for CDMSII detectors which have 4 phonon channels) to read out one iZIP (which has 8 phonon channels) at Soudan. To preserve the sanity of those doing data analysis, we have chosen a set of "analysis" side names that correspond to the physical location of the channel on the detector rather than which cold hardware channel it is hooked up to. Those doing analysis should not have to know about the "dib" naming scheme. The "analysis" names are distinguished by having the letter "S" for "side" in the channel name. For the benefit of DAQ and pipeline experts, we document the dib-user mapping in the list above and in fig. 2. Note that the mapping to convert from dib name to analysis names is done by the DAQ. The mapping that is found in this document (channel code to analysis name) must match the mapping that is stored in:

`cdmsbats/BatCommon/datareader/BatRootTypes.h`

The time field t0 is the beginning of the trace relative to time of trigger for that digitizer module, and (delta t) is 1/(digitization rate). Words 12 and 14 should always be exactly equal. This is different from the formatting in CDMS note 0004003. It was allowed in the older format to have more than one digitization rate for the same trace. Thus each trace could have more than 1 group of (t0, delta t, number of points), and the sum of all the number of points would add up to the number of samples. This was never implemented in SUF and will not be for Soudan. Note that each of the data words holds two ADC samples so that in software a maximum dynamic range of 16 bits is allowed.

The Soudan history buffer record format is shown below:

```
Soudan History Buffer record format
-----------------------------------------------------------
 Word No.                          Description
-----------------------------------------------------------
   1                           History Buffer Header = 0x0000 0021
   2                           History Buffer length (len)
   3                              Number of veto times (I32) nvt
   3+1                               time 1 (microseconds) (I32)
   3+2                               time 2 (microseconds) (I32)
   ...                                 ...
   3+nvt                             time nvt (microseconds) (I32)
   3+nvt+1                        Number of veto mask words (nvw) per time (I32)
   3+nvt+1+1                         time 1 - veto mask 1 (U32)
   ...                                 ...
   3+nvt+1+nvw                       time 1 - veto mast nvw (U32)
   3+nvt+1+nvw+1                     time 2 - veto mask 1 (U32)
   ...                                 ...
   3+nvt+1+nvw+nvw                   time 2 - veto mast nvw (U32)
   ...                                 ...
   ...                                 ...
   ...                               time nvt - veto mask 1 (U32)
   ...                                 ...
   3+nvt+1+nvt*nvw                   time nvt - veto mask nvw (U32)
   3+nvt+1+nvt*nvw+1              Number of trigger times (I32) ntt
   3+nvt+1+nvt*nvw+1+1               time 1 (microseconds) (I32)
   3+nvt+1+nvt*nvw+1+2               time 2 (microseconds) (I32)
   ...                                 ...
   3+nvt+1+nvt*nvw+1+ntt             time ntt (microseconds) (I32)
   3+nvt+1+nvt*nvw+1+ntt+1        Number of trigger mask words (ntw) per time (I32)
   3+nvt+1+nvt*nvw+1+ntt+1+1         time 1 - trigger mask 1 (U32)
   ...                                 ...
   3+nvt+1+nvt*nvw+1+ntt+1+ntw       time 1 - trigger mask ntw (U32)
   ...                                 ...
   ...                                 ...
   ...                                 ...
   3+nvt+1+nvt*nvw+1+ntt+1+ntt*ntw   time ntt - trigger mask ntw (U32)
```

Note that we have removed the global trigger mask since those are redundant – they are already contained in the trigger masks coming from the history buffer data. For scalability, we have included the nvw (number of veto mask words per time stamp) and ntw (number of trigger mask words per time stamp) so that the format remains the same as the data buffer increases when we go to more detector towers.

If we are missing the veto history buffer data (i.e. reading out only detectors), then the veto section will have exactly two zero words: nvt = nvw = 0x0 (words 3 and 4). Similarly if no detector history buffer records are readout.

For the case when the number of detector VME crates becomes larger than 1, we have no information in the data record to discriminate for example whether the 4 masks come from a (tower 1-2)-(tower3-4) set or from a (tower 1-2)-(tower 5) set. To remove this ambiguity, we will insist that the number of trigger masks always be 6 (ntw == 6) so that the tower information is set by the mask location in this 6-grouping.

The Trigger logical record format is shown below. The SUF standard had fixed the size of this buffer to 5 words total since we only had exactly one Individual Trigger Mask word and one Global Trigger Mask word. For Soudan, going to more towers would give additional Trigger Mask words. We have chosen to keep the record format identical to that at SUF and have the Individual Trigger Mask words consecutively recorded; thus ntw will be deduced from the len word. Note that this ntw is exactly the same as that in the Soudan History Buffer record. Also, we have deleted the Global Trigger Mask word since as previously mentioned, this information is already contained inside the Individual Mask Words. Trigger time is identically 0, being the time of the global trigger relative to itself. Note that with use of the Struck 64 channel history buffer electronic modules, the trigger masks come in groupings of 2 towers each. For Soudan runs 118 and 119 in which we had operated 1 and 2 towers, respectively, only trigger masks 1 and 2 were present. For the anticipated 5-towers Soudan run in 2005, and for convenience in the data reconstruction pipeline, we will fix the number of trigger masks below, ntw, to always be six.

```
Trigger record format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                      Trigger Header = 0x0000 0080
   2                      Trigger length (len)
   3                         Trigger Time (always 0)
   3+1                          Individual Trigger Mask 1 (U32)
   3+2                          Individual Trigger Mask 2 (U32)
   ...                            ...
   3+ntw                        Individual Trigger Mask ntw (U32)
```

The TLB Trigger Mask logical record is shown below. This logical record is added with the 5-towers, 2005 run at Soudan. Like the Soudan history logical record above, we will fix the number of TLB trigger masks to be six even though we will operate only five towers. Thus ntw == 6 below as well.

```
TLB Trigger Mask record format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                      TLB Mask Header = 0x0000 0081
   2                      length (len)
   2+1                       Tower 1 TLB Mask (U32)
   2+2                       Tower 2 TLB Mask (U32)
   ...                         ...
   2+ntw                     Tower ntw TLB Mask (U32)
```

For each tower, the U32 mask word will take the format (in hex)

```
      tower TLB mask = 0xttmm mmmm
```

in which tt=tower number, and the 24 remaining mask bits are either 1 (ZIP trigger) or 0 (no trigger), with the bit number corresponding to the ZIP number within that tower. This information comes directly from the TLB electronics into the VMIC Digital In VME boards. As an example, if the event was caused by a single trigger from ZIP 3 of tower 1, then the TLB Trigger record would read

```
0x0000 0081
0x0000 0018 (24 in decimal)
0x0100 0004
0x0200 0000
0x0300 0000
0x0400 0000
0x0500 0000
0x0000 0000
```

Note that the sixth TLB mask has its tower ID field identically zero since we only have 5 towers.

With the NUMI beam coming online, we have added a VME GPS-decoder module so that we can stamp the GPS time of our events and veto the window of the NUMI beam. The NUMI beam will come in bunches about 1.6 us wide, roughly a few bunches occuring in a 10 us wide window that is repeated once every 2-4 seconds. Our GPS-decoder will stamp the GPS time to be compared offline to the NUMI beam spill times for software vetoing. The format is shown below. The 3 GPS words are in BCD (binary-coded decimal) format of the form:

```
GPS year/day                      0xyyyy dddd
GPS Status/hour/minute/second     0xS0hh mmss
GPS 0.1us's from GPS second       0xuuuu uuuu
```

For example, consider an event occurring on the 320th day of the year 2005, at the time of 11:15:26 AM, and at exactly 0.2 seconds after the turn of the GPS second. Then the 3 GPS words above would be printed out in hex as

```
0x2005 0320
0x0011 1526
0x0200 0000 (2,000,000 counts of 0.1 micro second units)
```

Note in this case that the Status bits are all zeros, meaning that the GPS unit is synced with the IRIG-B input signal, and that the internal timing is kept at a precision of $\pm 5\mu$s with respect to the input signal.

```
GPS Date/Time format
------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------
   1                      TLB Mask Header = 0x0000 0060
   2                      length (len)
   3                         GPS year and day (U32)
   4                         GPS Status/hour/minute/second (U32)
   5                         GPS 0.1us's from GPS second (U32)
```

7 Logical Records – Data Monitoring Event Types

We have used the older definitions from CDMS note 0004003 to define the logical header words below for the monitoring event types. Note that these were originally used to specify the event type, but we have modified that scheme and have used 0x7 for data monitoring event and have deferred these to the logical headers as listed below:

```
0x0000 0000    - Dark Monitor
0x0000 0010    - Fridge Monitor
0x0000 0020    - Detector Temperatures
0x0000 0021    - Detector Trigger Thresholds
0x0000 0022    - Detector Trigger Rates
0x0000 0030    - Veto HV
0x0000 0031    - Veto Rates
0x0000 0032    - Veto Spectra
0x0000 0040    - Electronics Crate Voltages
0x0000 0040    - Crate HV
0x0000 0050    - Enviroment
0x0000 00F0    - MINOS Spill
```

The RTF warm electronics for monitoring is organized in modules of towers. Thus it requires that the raw data format be organized similarly. For reference, the RTF electronics modules output for each detector a total of 9 ADC threshold lines (5 for Phigh/Plow/Qhigh/Qlow/Whisper; 4 for the phonon offsets A-D) and a total of 5 scaler lines from each of the triggering channels.

Physically, the trigger thresholds and phonon offsets for each tower (6*9=54 total) are read out by a 64 channel scanning ADC module (VMIVME3128). This structrue guides the Detector Trigger Thresholds data format below:

```
Detector Trigger Thresholds data format
------------------------------------------------------------------
 Word No.                      Description
------------------------------------------------------------------
   1                     Header = 0x0000 0021
   2                     length to next Header (in bytes)
   3                        minimum voltage level (in V)
   4                        maximum voltage level (in V)
   5                        dynamic range (14 bits = 0x3fff)
   6                     tower number
   7-12                  6 detector codes
   13-21                 9 operations codes
   22                       adc value: det code 1, operation code 1
   23                       adc value: det code 1, operation code 2
   ...
   75                       adc value: det code 6, operation code 9
```

Recall that in the Trace Logical data records, the detector code takes the form xyyz. For our purposes here, the detector code will take only the form xyy with x and yy as defined previously and here repeated for easy reference:

```
x = 1  BLIP    yy = Detector #
    2  FLIP         (01-99)
    3  Veto
    4  ZIP
    5  mercedes ZIP
    6  endcap detector
```

The operations code takes the form p00j where

```
p = 1  phonon offset      j = 1-4 corresponding to phonon A-D
p = 2  trigger theshold   j = 1 Qhigh
```

```
                                          = 2 Qlow
                                          = 3 Phigh
                                          = 4 Plow
                                          = 5 Whisper
```

Note: For SUF only Plow/Qlow were recorded and written out using a different j convention (j=1 for Qlow; j=2 for Plow). We have changed this convention to match the grouping in the actual ordering of the physical cable lines.

Similarly, the Detector Trigger Rates data format has also been organized according to tower groups:

```
Detector Trigger Rates data format
------------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------------
   1                        Header = 0x0000 0022
   2                        length to next Header (in bytes)
   3                           clocking interval (in us)
   4                        tower number
   5-10                     6 detector codes
   11-15                    5 j-codes
   16                          counter value: det code 1, j-code 1
   17                          counter value: det code 1, j-code 2
   ...                         ...
   16                          counter value: det code 6, j-code 5
```

The detector code is as in the Detector Trigger Thresholds data format, and j-code is also as described there in the operations code, 1-5 corresponding to Qhigh/Qlow/Phigh/Plow/Whisper.

For the veto panels, all are included in the same raw data structure:

```
Veto Trigger Rates data format
------------------------------------------------------------------
 Word No.                       Description
------------------------------------------------------------------
   1                        Header = 0x0000 0031
   2                        length to next Header (in bytes)
   3                           clocking interval (in us)
   4                        number of entries (np)
   4+1                         detector code 1
   ...
   4+np                        detector code np
   4+np+1                      counter value: det code 1
   ...
   4+np+np                     counter value: det code np
```

The detector code is as described above of the form xyy. For the veto panels, x=3, and we will begin the Soudan run with yy=01-40. The veto OR channel will have the detector code xyy=300. Thus, at the beginning of Soudan running, we will have np=41.