E835 Event Header and Unpacking

Keith Gollwitzer and George Zioulas
UC Irvine

February 29, 1996

The Event Header will be formed in the E835 online filter code PRUDE. The Event Header will consist of 17 words which are listed below and described. Some of the words are pointers to the DYC header words within the data words for the event. This subset of the Event Header will allow unpacking of the E835 data. In this document are the Event Header words and code, both C and Fortran, for unpacking the data. The code is generalized with Event Header and data offsets.

Below is a table showing the Event Header with the array element number for both C and Fortran as applicable to PRUDE; when using the DART daft library routines for I/O, the array elements are offset by one (see below for further explanation). The specific bits used by each part of the Event Header are indicated. Below assumes 32 bit words: type `int` for C and `integer*4` for Fortran (`integer*2` is treated on page 4).

Element #			
C	Fortran	Bits	Description
0	1	0-31	Number of inclusive bytes for event
1	2	16-31	Gateway buffer number
1	2	14-15	Number of streams
1	2	0-13	Run number
2	3	30-31	Processor number
2	3	0-29	Software event number
3	4	24-31	Error flag
3	4	16-23	Software trigger ID
3	4	0-15	Gatemaster bits
4	5	0-31	Software trigger bits 32-63
5	6	0-31	Software trigger bits 0-31
6	7	16-31	Pointer to DYC 1
6	7	0-15	Pointer to DYC 2
7	8	16-31	Pointer to DYC 3
7	8	0-15	Pointer to DYC 4
8	9	16-31	Pointer to DYC 5
8	9	0-15	Pointer to DYC 6
9	10	16-31	Pointer to DYC 7
9	10	0-15	Pointer to DYC 8

10	11	16-31	Pointer to DYC 9
10	11	0-15	Pointer to DYC 10
11	12	16-31	Pointer to DYC 11
11	12	0-15	Pointer to DYC 12
12	13	16-31	Pointer to DYC 13
12	13	0-15	Pointer to DYC 14
13	14	16-31	Pointer to ROC
13	14	0-15	Pointer to event summary
14	15	0-31	DDD Stream 0 DYC order
15	16	0-31	DDD Stream 1 DYC order
16	17	0-31	Time stamp

More detailed descriptions of some of the Event Header are the following:

Inclusive byte count--daft expects the first word to be the number of bytes for the event's data;

Number of streams -- the number of DDD/ROC data streams in the data (maximum of 3 streams, 2 DDDs plus ROC);

Gateway buffer number -- the gateway numbers the buffers that it sends to the PRUDE filters;

Software event and processor numbers -- combined with the run number form an unique identification for an event;

Gatemaster bits -- the decoded hit channels for the appropriate TDC which serves as a latch for the input signals to the gatemaster;

Software trigger ID -- a number which identifies which software trigger bit conditions were met and which software trigger ID the event was assigned;

Error flag -- bits will signify error conditions encountered while doing the PRUDE analyses for the event;

Software trigger bits -- 64 bits can be possibly set by PRUDE analyses and are checked against software trigger masks for required ON/OFF set of bits;

Pointer to DYC *n* -- which word of the event is DYC *n* header word;

Pointer to ROC -- which word of the event is ROC header word;

Pointer to event summary -- which word begins the event summary written by the PRUDE analyses;

Stream *N* DYC order -- the order that the DYCs are connected to DDD *N* where each DYC requires 4 bits (bits 0-3 have the DYC number that is first connected to DDD *N*; bits 4-7 are the second DYC number connected; etc.);

Time stamp -- the time that the event was received from the gateway by PRUDE with respect to January 1, 1992 (bits 4-31 are the number of seconds and bits 0-3 is tenth of seconds).

The following code is for unpacking the pointer, dyc_point , to DYC *n,* dyc_n , and involves offsets in the event header, off_head , and to the DYC header word, off_data . All variables are integers.This is how to unpack in C:

```
if ( dyc_n & 0x1 )  {
  dyc_point = ( ( array[ ( dyc_n/2 ) + off_head ] & 0xFFFF0000 ) >> 16 ) - off_data; }
else {
  dyc_point = ( array[ ( dyc_n/2 ) + off_head - 1 ] & 0xFFFF ) - off_data; }
```

Here is how to do the same in Fortran:

```
if ( ( dyc_n.and.1 ) = 1 ) then
  dyc_point = ( ( array(( dyc_n/2) + off_head ).and.'FFFF0000'X ) / '10000'X ) - off_data
else
  dyc_point = ( array((dyc_n/2) + off_head -1 ).and.'FFFF'X ) - off_data
endif
```

Neither code shown above is an unique way to unpack.The offsets depend upon which program, language and I/O routine that you are using. Here are three foreseeable unpacking conditions with the appropriate offsets list for the Fortran language {C offset}:

`PRUDE` -- The online filter uses three arrays for en event's information: header, data and summary. When unpacking for filter analyses, the header has already been built and is in a separate array from the raw data array. The offset values are 7{6} for off_head and 17{18} for off_data.

`Routine where I/O is done by daft library` -- daft does not return the first word of the event (the first header word), the inclusive byte count, within the passed array; however, the byte count is returned as a separate variable. The offset values are 6{5} for off_head and 1{2} for off_data .

`Routine` *`not`* `using daft library` -- the DYC pointer is meant for this case when unpacking the raw data in Fortran. The offset values are 7{6} for off_head and 0{1} for off_data .

Below is a treatment when using integer*2 in Fortran assuming that an equivalence statement is used to convert integer*4 to integer*2 . The Event Header, now 34 words, is then the following (as appropriate for PRUDE):

Element #	Bits	Description
1	0-15	Number of inclusive bytes for event (upper bits)
2	0-15	Number of inclusive bytes for event (lower bits)
3	0-15	Gateway buffer number
4	14-15	Number of streams
4	0-13	Run number
5	14-15	Processor number
5	0-13	Software event number (upper bits)
6	0-15	Software event number (lower bits)
7	8-15	Error flag
7	0- 7	Software trigger ID
8	0-15	Gatemaster bits
9	0-15	Software trigger bits 48-63
10	0-15	Software trigger bits 32-47
11	0-15	Software trigger bits 16-31
12	0-15	Software trigger bits 0-15
13	0-15	Pointer to DYC 1
14	0-15	Pointer to DYC 2
15	0-15	Pointer to DYC 3
16	0-15	Pointer to DYC 4
17	0-15	Pointer to DYC 5
18	0-15	Pointer to DYC 6
19	0-15	Pointer to DYC 7
20	0-15	Pointer to DYC 8
21	0-15	Pointer to DYC 9
22	0-15	Pointer to DYC 10
23	0-15	Pointer to DYC 11
24	0-15	Pointer to DYC 12
25	0-15	Pointer to DYC 13
26	0-15	Pointer to DYC 14
27	0-15	Pointer to ROC
28	0-15	Pointer to event summary
29	0-15	DDD Stream 0 DYC order (fifth to eighth)
30	0-15	DDD Stream 0 DYC order (first to fourth)
31	0-15	DDD Stream 1 DYC order (fifth to eight)
32	0-15	DDD Stream 1 DYC order (first to fourth)
33	0-15	Time stamp (upper bits of seconds)
34	0-15	Time stamp (lower bit of seconds and tenths)

For better description of any of the above see second page.

The unpacking code is straight forward in using an integer*2 array in Fortran:

```
equivalence ( array_2(1), array(1) )
dyc_point = ( 2 * array_2( dyc_n + off_head ) ) - off_data
```

The pointer now points to the DYC marker. The next integer*2 contains information about number of words in the DYC. Again the offsets are dependent upon program and I/O routine used:

`PRUDE` -- The offsets are12 for off_head and 33 for off_data

`Routine where I/O is done by daft library` -- The equivalence statement of the integer*4 array passed by the daft routine will start with the third integer*2 word listed above. The offsets are 10 for off_head and 3 for off_data .

`Routine not using daft library` -- The offsets are 12 for off_head and 1 for off_data .