Hall C User Howto

Experiment: SANE

Introduction to raw data structure

M. K. Jones

January 27, 2008

Abstract

Introduction to raw CODA data structure and how it is read into the Hall C engine.

1 Raw CODA data structure

The raw CODA data structure has the basic structure shown in Fig. 1. For each event the CODA raw data event is read into the array *craw* using a standard CODA subroutine in the Analyzer subroutine *ENGINE/g_get_next_event.f* called from *engine.f*. The first *craw* array element is the total array length and the second contains the event type information. Next is an event ID

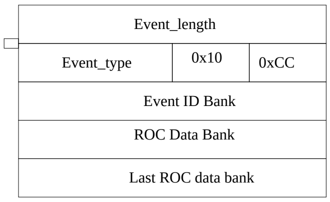

Figure 1: General CODA raw event structure

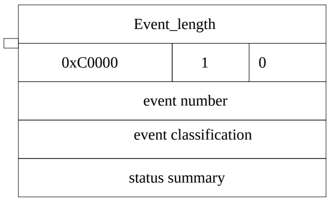

Figure 2: Event ID bank structure

roc_bank_length		
(code & status) ROC#	1	counter
data word		
data word		
last data word		

Figure 3: Generic ROC bank structure

bank with the details shown in Fig. 2. Following that are the ROC (Read Out Controller) data banks whose generic structure is shown in Fig. 3.

In the *engine.f* , the code determines the event type and if it is a physics event type then calls *g_reconstruction.f* where the array *craw* is passed into the array *event*. In *g_reconstruction.f*, *event* is passed to the subroutine *g_decode_event_by_banks.f*. This subroutine checks that *event(2)* contains 10CC which double checks that it is standard CODA data event. It gets the event length from *event(1)* and checks that *event(4)* equals C0000100 which indicates that*event* contains an event bank as expected. Then the array index *bankpointer* is set to 3+event(3)+1 which is the index of first word in the ROC bank which contains the length of the first ROC bank. There is a *while* loop which calls the subroutine *g_decode_fb_bank.f* and goes through the rest of the *event* array for each ROC bank.

The *event* array is passed to the subroutine *g_decode_fb_bank.f* to fill the array *bank*. *event* is passed in such a way that *bank(1:evlength)* is equal to *event(bankpointer:evlength)* where *evlength* is the total length in array *event* and *bankpointer* points to the location in *event* which the ROC bank starts. Therefore *bank(1)* is the present ROC bank length and in *bank(2)* contains the ROC identification number. A table of the ROC ID numbers is in Table 3. For a physics event, the scaler ROCs contain only header words and are skipped over. For ROC 21, the helicity information for the event (whether it is plus, minus or indeterminate helicity state) is determined and the input trigger pattern to the Trigger Supervisor is read. For the other fastbus ROC , the code reads through the bank in a loop. First it gets the slot and chan number and determines the detector ID number from the detector map. Depending on the detector ID, the function *g_decode_fb_detector* is called with different arguments.

The array *bank* is passed into the array *evfrag* in *g_decode_fb_detector*. The other arguments that are passed to *g_decode_fb_detector* are the present slot number (*oslot*), the present ROC ID (*roc*), maximum number of words *maxwords*, the present detector *did* and the flag *signalcount*. *signalcount* can have the values of 1, 2, 3 or 4 which refers to the number of raw detector arrays that could be filled. In Table 1, examples of the HMS detector IDs are given. So for example , the HMS scintillator detector has four possible raw arrays to be filled. The detector map indicates which raw array is filled for that particular ROC, slot and channel. The function loops through the array *evfrag* until it reach a data word for a new detector ID and then it returns to *g_decode_fb_bank.f*. Before returning to *g_decode_fb_bank.f*, the function *g_decode_fb_detector* is set to the total number of words processed. Then in *g_decode_fb_bank.f*, the *pointer* is increased by*pointer* + *g_decode_fb_detector* and it reads then next data word and gets the new detector ID. This loop continues until the ROC bank is read. Then it returns to *g_decode_event_by_banks.f* and gets the *pointer* for the next ROC bank and goes through the same process with that ROC bank until the entire CODA data bank is read.

2 Event types

There are standard CODA event types for Prestart (event type = 17). GO (18), PAUSE (19) and END (20). Event types 0-15 are reserved for user trigger event types. Event types larger than 20 are used to identify data such EPICS data, HV data, start of run info, prescales. A description of the event types is given in Table 2.

3 ROC identification numbers

Detector	ID	Arrays
Drift chamber planes	1	HDC_RAW_TDC
Scintillator planes	2	HSCIN_ALL_ADC_POS,HSCIN_ALL_ADC_NEG HSCIN_ALL_TDC_POS,HSCIN_ALL_TDC_NEG
Cerenkov	3	HCER_RAW_ADC
Calorimeter	4	HCAL_ADC_POS, HCAL_ADC_NEG
HMISC	5	HMISC_RAW_DATA
GMISC	6	GMISC_RAW_DATA
Aerogel	7	HAERO_ADC_POS, HAERO_ADC_NEG HAERO_TDC_POS, HAERO_TDC_NEG
BigCal Protvino blocks	8	BIGCAL_PROT_ADC_RAW
BigCal RCS blocks	9	BIGCAL_RCS_ADC_RAW
BigCal Group of 8 TDC	10	BIGCAL_TDC_RAW_IGROUP
BigCal Trigger ADC	18	BIGCAL_ATRIG_ADC_RAW
BigCal Trigger TDC	19	BIGCAL_TTRIG_TDC_RAW
BETA tracker Y-planes	21	
BETA tracker X-planes	22	
BETA Cerenkov	23	
BETA Lucite	24	

Table 1: HMS Detector IDS

Event type	Description
0	Scaler or sync event
1	HMS
2	SOS
3	HMS and SOS
4	Pedestal
5	BETA
6	BETA and HMS
8	LASER

Table 2: Description of event types

ROC ID number	Description
1	HMS ADCs and TDCs
2	HMS drift chambers TDCs
5	BigCal scalers
11	BigCal Protvino part ADC
12	BigCal RCS part ADC, BETA TDCs and ADCs
20	HMS scalers
21	Trigger Supervisor data (trigger pattern, helicity info)

Table 3: Description of ROC identification numbers